core WEB programming
SECOND EDITION
The worldwide best seller for serious Web developers—now 100% updated!
In-depth HTML 4/CSS, Java 2, Servlets, JavaServer Pages, XML, and more!
Industrial-strength code examples throughout
THE SUN MICROSYSTEMS PRESS JAVA SERIES
Sun
MARTY HALL · LARRY BROWN

Java:

A Software Revolution?

http://courses.coreservlets.com

Agenda

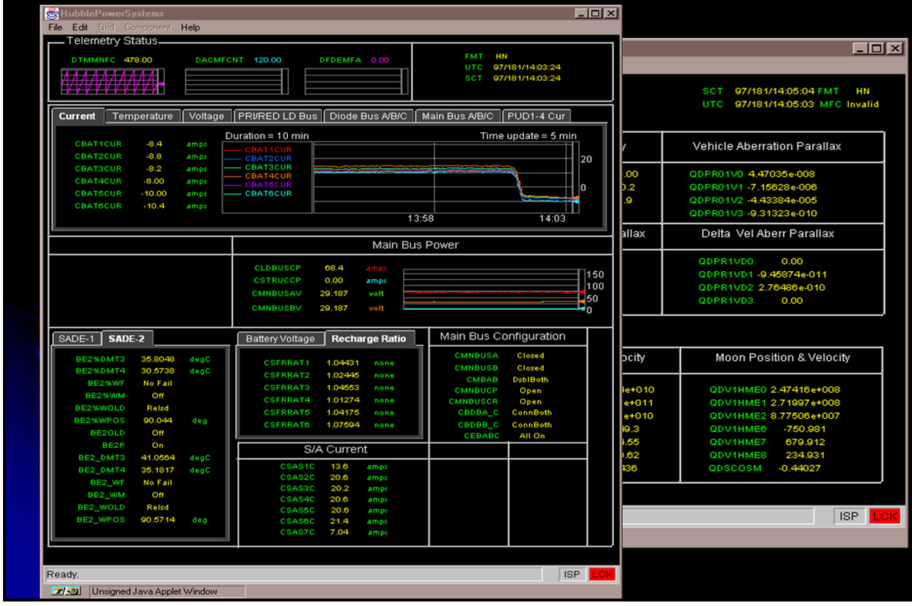
- Truths / Myths About Java
  - Java is Web-Enabled?
  - Java is Safe?
  - Java is Cross-Platform?
  - Java is Simple?
  - Java is Powerful?
- Common Java Protocols and Packages
- The Future of Java
- Getting Started
- Questions and (Hopefully) Answers

Java is Web-Enabled?

- Truth: Web browsers can run Java “applets”
  - The Web can be used for *software* delivery and *execution*, not just *document* delivery and *display*
  - No more installation or updates; just a bookmark
  - Large, complex applets best suited for intranets. Fits the APL model better than the WWW at large.
- Truth: Java’s network library is easy to use
  - Ordinary mortals can do socket programming
  - Standard distributed object protocol and DBMS API

Hubble Space Telescope Monitoring:

“NASA Goddard’s Most Successful SW Project *Ever*.”



DTMMNFC	478.00
DACMFCNT	120.00
DFDEMFA	0.00

FMT	HN
UTC	97/181/14:03:24
SCT	97/181/14:03:24

CBAT1CUR	-8.4	amps
CBAT2CUR	-8.8	amps
CBAT3CUR	-8.2	amps
CBAT4CUR	-8.00	amps
CBAT5CUR	-10.00	amps
CBAT6CUR	-10.4	amps

Main Bus Power

CLDBUSCP	68.4	amps
CSTRUCCP	0.00	amps
CMNBUSAV	29.187	volt
CMNBUSBV	29.187	volt

BE2%DMT3	35.8048	degC
BE2%DMT4	30.5738	degC
BE2%WF	No Fail	
BE2%WM	Off	
BE2%WOLD	Relsd	
BE2%WPOS	90.044	deg
BE2OLD	Off	
BE2P	On	
BE2_DMT3	41.0564	degC
BE2_DMT4	35.1817	degC
BE2_WF	No Fail	
BE2_WM	Off	
BE2_WOLD	Relsd	
BE2_WPOS	90.5714	deg

CSFRRAT1	1.04431	none
CSFRRAT2	1.02445	none
CSFRRAT3	1.04553	none
CSFRRAT4	1.01274	none
CSFRRAT5	1.04175	none
CSFRRAT6	1.07594	none

S/A Current

CSAS1C	13.6	amps
CSAS2C	20.6	amps
CSAS3C	20.2	amps
CSAS4C	20.6	amps
CSAS5C	20.6	amps
CSAS6C	21.4	amps
CSAS7C	7.04	amps

Main Bus Configuration

CMNBUSA	Closed
CMNBUSB	Closed
CMBAB	DsblBoth
CMNBUCP	Open
CMNBUSCR	Open
CBDBA_C	ConnBoth
CBDBB_C	ConnBoth
CEBABC	All On

Ready.

SCT	97/181/14:05:04	FMT	HN
UTC	97/181/14:05:03	MFC	Invalid

Vehicle Aberration Parallax

QDPR01V0	4.47035e-008
QDPR01V1	-7.15628e-006
QDPR01V2	-4.43384e-005
QDPR01V3	-9.31323e-010

Delta Vel Aberr Parallax

QDPR1VD0	0.00
QDPR1VD1	-9.45874e-011
QDPR1VD2	2.76486e-010
QDPR1VD3	0.00

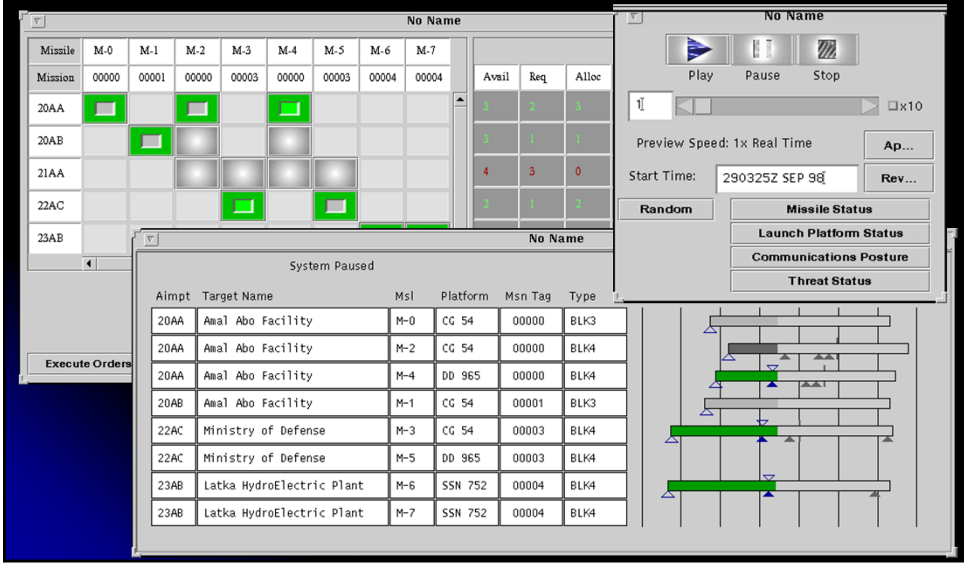
Moon Position & Velocity

QDV1HME0	2.47416e+008
QDV1HME1	2.71997e+008
QDV1HME2	8.77506e+007
QDV1HME6	-750.981
QDV1HME7	679.912
QDV1HME8	234.931
QDSCOSM	-0.44027

Java is Web-Enabled?

- Myth: Java is *only* for the Web
  - Java “applets” run in Web pages
  - Java “applications” run stand-alone
  - Current usage (roughly)
    - Client (applet): 5%
    - Desktop (application): 45%
    - Server (servlets/JSP/EJB): 50%

Tomahawk Strike Coordination Planner (APL/PPSD)



No Name

Missile	M-0	M-1	M-2	M-3	M-4	M-5	M-6	M-7
Mission	00000	00001	00000	00003	00000	00003	00004	00004
20AA	🟩		🟩		🟩			
20AB		🟩	●		●			
21AA			●	●	●	●		
22AC				🟩		🟩		
23AB								

Avail	Req	Alloc
3	2	3
3	1	1
4	3	0
2	1	2

No Name

System Paused

Aimpt	Target Name	Msl	Platform	Msn Tag	Type
20AA	Amal Abo Facility	M-0	CG 54	00000	BLK3
20AA	Amal Abo Facility	M-2	CG 54	00000	BLK4
20AA	Amal Abo Facility	M-4	DD 965	00000	BLK4
20AB	Amal Abo Facility	M-1	CG 54	00001	BLK3
22AC	Ministry of Defense	M-3	CG 54	00003	BLK4
22AC	Ministry of Defense	M-5	DD 965	00003	BLK4
23AB	Latka HydroElectric Plant	M-6	SSN 752	00004	BLK4
23AB	Latka HydroElectric Plant	M-7	SSN 752	00004	BLK4

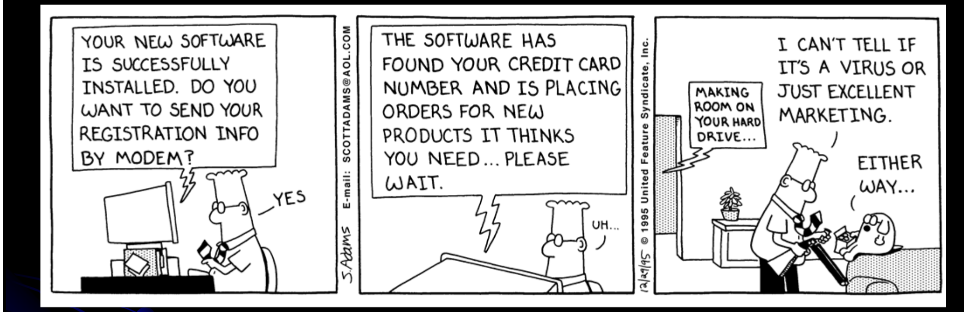
No Name

1 □x10

Preview Speed: 1x Real Time

Start Time: 290325Z SEP 98

Java is Safe?


YOUR NEW SOFTWARE IS SUCCESSFULLY INSTALLED. DO YOU WANT TO SEND YOUR REGISTRATION INFO BY MODEM?
YES
E-mail: SCOTTADAMS@AOL.COM
S.Adams
THE SOFTWARE HAS FOUND YOUR CREDIT CARD NUMBER AND IS PLACING ORDERS FOR NEW PRODUCTS IT THINKS YOU NEED... PLEASE WAIT.
UH...
12/29/95 © 1995 United Feature Syndicate, Inc.
MAKING ROOM ON YOUR HARD DRIVE...
I CAN'T TELL IF IT'S A VIRUS OR JUST EXCELLENT MARKETING.
EITHER WAY...

Dilbert copyright United Media. Used with permission.

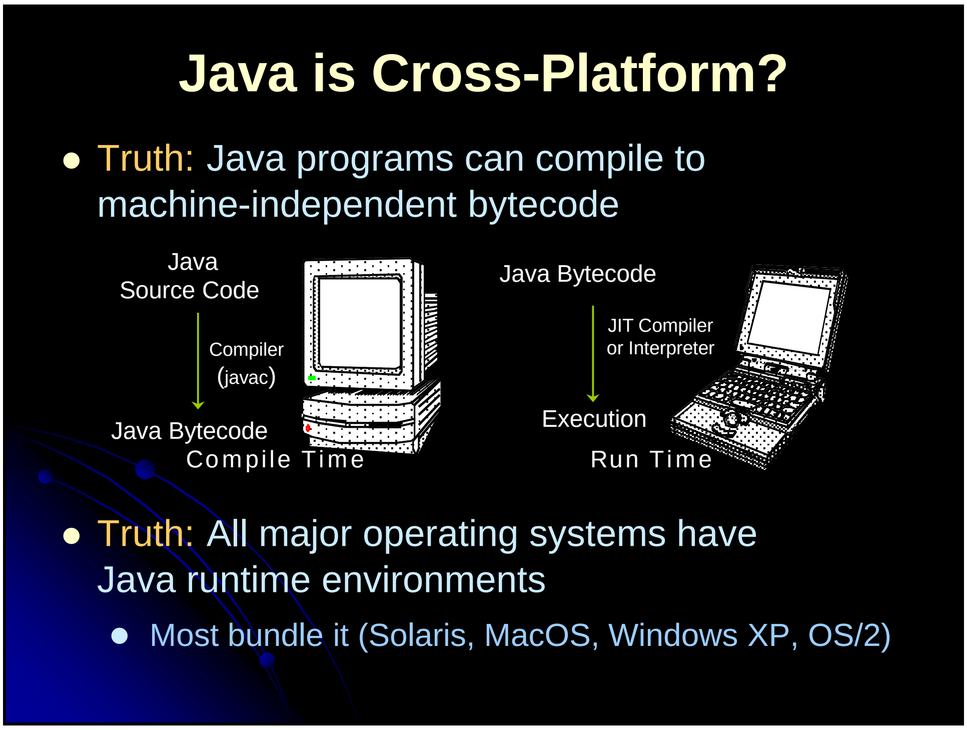
- **JAVA: Just Another Virus Architecture?**

Java is Safe?

- Truth: Restrictions on permissible operations can be enforced
  - No “raw” memory manipulation (directly or indirectly).
    - Thus, it is easy to identify prohibited operations.
  - Applets, by default, prohibited from:
    - Reading from the local disk
    - Writing to the local disk
    - Executing local programs
    - Opening network connections other than to HTTP server
    - Discovering private info about user (username, directories, OS patch level, applications installed, etc.).

Java is Safe?

- Myth: Applets cannot harm your computer
  - Denial of service
  - Browser misconfiguration
  - Implementation bugs
- Myth: Java is too restricted to be useful
  - Restrictions apply only to applets, not regular Java programs
  - Digital signatures support relaxed restrictions
- Myth: Applets with digital signatures are no more or less safe than ActiveX
  - Relaxed security in applets not “all or nothing” as in ActiveX



Java is Cross-Platform?

- Truth: Java programs can compile to machine-independent bytecode

Java Source Code

Compiler (javac)

Java Bytecode

Compile Time

Java Bytecode

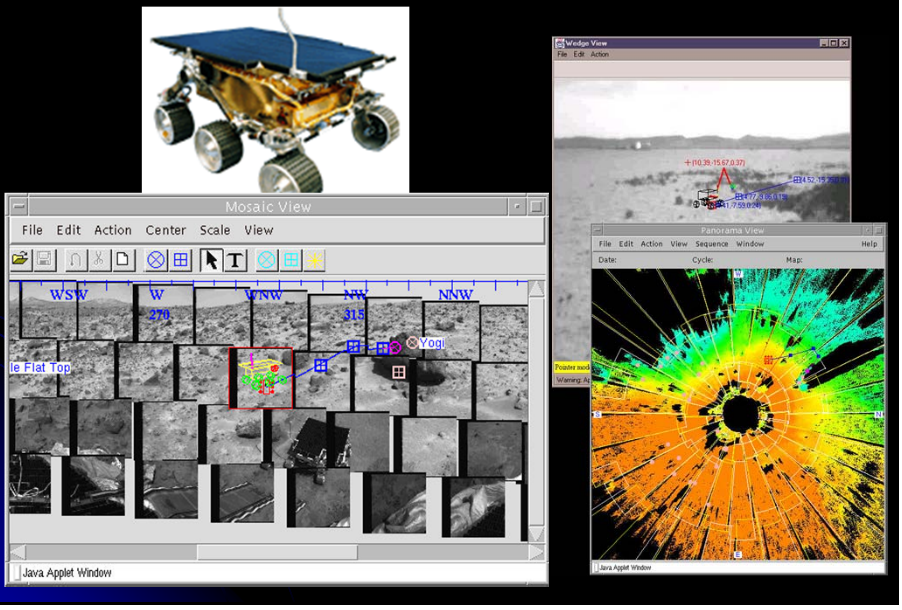
JIT Compiler or Interpreter

Execution

Run Time

- Truth: All major operating systems have Java runtime environments
  - Most bundle it (Solaris, MacOS, Windows XP, OS/2)

Mars Rover Controller and Simulator



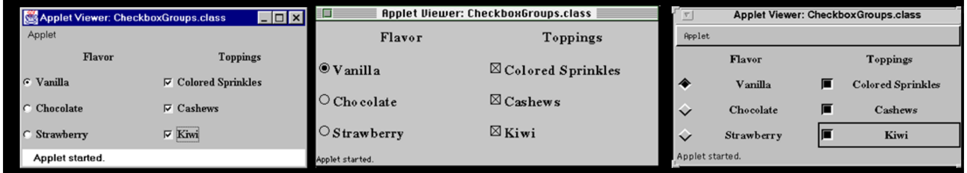
Java is Cross-Platform?

- Myth: Safety and machine independence can be achieved with no performance penalty
  - Current systems are about 20% slower than C++
  - Upcoming releases claim to lower or eliminate that gap
  - I expect the gap to stay at 10% or more
  - Commercial compilers are sometimes faster than free ones
- Myth: Java is interpreted
  - Early releases were interpreted
  - Many major “Just in Time” (JIT) compilers
  - HotSpot and “native” compilers even faster (IBM, Symantec, TowerJ, etc.)

Java is Cross-Platform?

- Truth: Java has a portable graphics library

"Native look & feel" -- Java 1.1 UI controls adapt to OS



Applet Viewer: CheckboxGroups.class
Applet

Flavor	Toppings
◉ Vanilla	☑ Colored Sprinkles
○ Chocolate	☑ Cashews
○ Strawberry	☑ Kiwi

Applet started.

Applet Viewer: CheckboxGroups.class

Flavor	Toppings
◉ Vanilla	☒ Colored Sprinkles
○ Chocolate	☒ Cashews
○ Strawberry	☒ Kiwi

Applet started.

Applet Viewer: CheckboxGroups.class
Applet

Flavor	Toppings
◆ Vanilla	■ Colored Sprinkles
◇ Chocolate	■ Cashews
◇ Strawberry	■ Kiwi

Applet started.

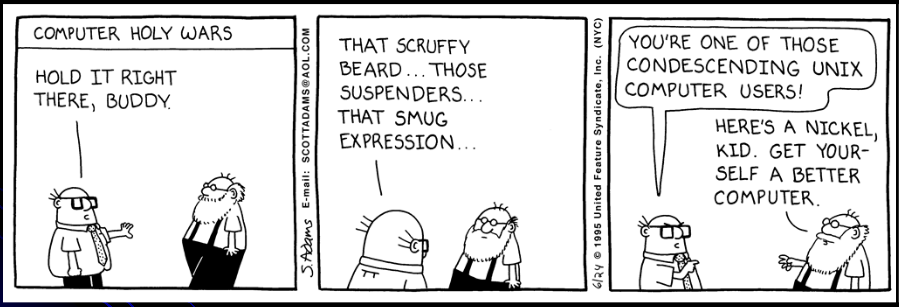
"Pluggable look & feel" -- Java 2 controls can change looks

- Myth: The graphics library has everything most applications need.

AWT (Java 1.0 and 1.1) was weak. JFC/Swing (Java 2) much more complete and powerful.

Java is Cross-Platform?

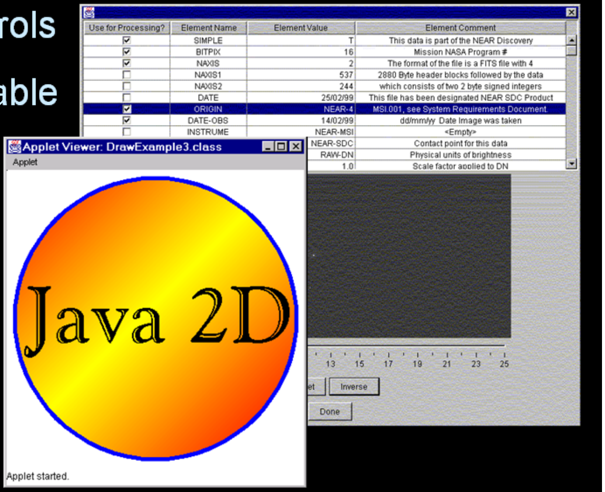
- Truth: (Opinion) Native look and feel was the right choice


COMPUTER HOLY WARS
HOLD IT RIGHT THERE, BUDDY.
E-mail: SCOTTADAMS@AOL.COM
S. Adams
THAT SCRUFFY BEARD... THOSE SUSPENDERS... THAT SMUG EXPRESSION...
6/24 © 1995 United Feature Syndicate, Inc. (NYC)
YOU'RE ONE OF THOSE CONDESCENDING UNIX COMPUTER USERS!
HERE'S A NICKEL, KID. GET YOURSELF A BETTER COMPUTER.

Dilbert copyright United Media. Used with permission.

Java Foundation Classes (JFC) Improve Graphics Considerably

- More GUI Controls
- More customizable
- Pluggable Look and Feel
- Native Fonts
- Richer Drawing Operations

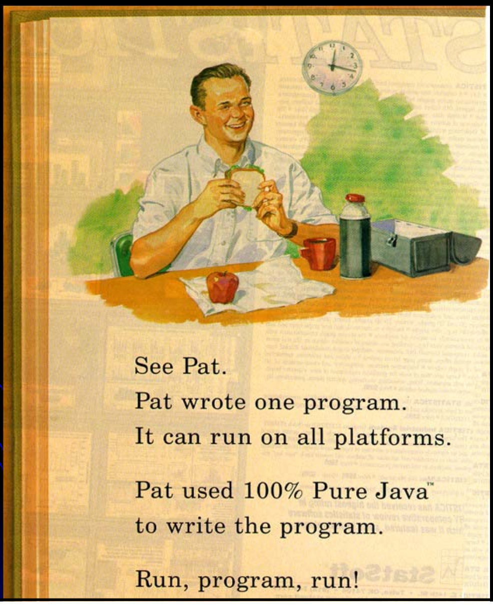


Use for Processing?	Element Name	Element Value	Element Comment
☑	SIMPLE	T	This data is part of the NEAR Discovery
☑	BITPIX	16	Mission NASA Program #
☑	NAXIS	2	The format of the file is a FITS file with 4
□	NAXIS1	537	2880 Byte header blocks followed by the data
□	NAXIS2	244	which consists of two 2 byte signed integers
□	DATE	25/02/99	This file has been designated NEAR SDC Product
☑	ORIGIN	NEAR-4	MSI.001, see System Requirements Document.
☑	DATE-OBS	14/02/99	dd/mm/yy Date Image was taken
□	INSTRUME	NEAR-MSI	<Empty>
		NEAR-SDC	Contact point for this data
		RAW-DN	Physical units of brightness
		1.0	Scale factor applied to DN

Java is Cross-Platform?

- Myth: Write Once Run Anywhere
  - Cross-platform code can be achieved, but you must test on all platforms you will deliver on.
    - Java applications can execute local code
    - The graphics library behaves slightly differently on different platforms
    - The behavior of the thread scheduler is only loosely defined
- Myth: Java will kill Microsoft
  - There is also no longer immediate danger of the reverse (Microsoft killing Java)
  - Microsoft wavered between trying to fight Java and joining it and making money by dominating the market. With .NET, they are back to fighting it again.

Sun Mantra:
“100% Pure Java”



See Pat.
Pat wrote one program.
It can run on all platforms.

Pat used 100% Pure Java™
to write the program.

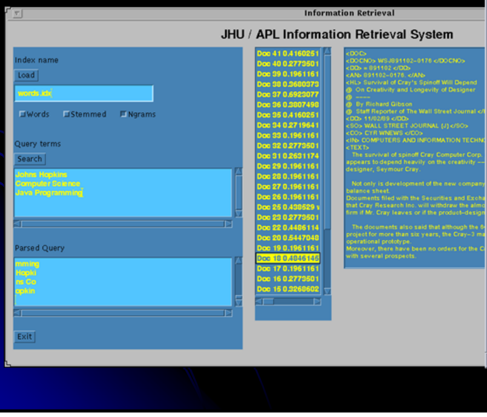
Run, program, run!

Java is Simple?

- Truth: Java greatly simplifies several language features
  - Java has automatic memory management
    - Does Windows and takes out the garbage
    - No dangling pointers. No memory leaks.
  - Java simplifies pointer handling
    - No explicit reference/dereference operations
  - No makefiles
  - No header files
  - C++ syntax streamlined
  - C# is comparable to Java, at least as far as the core language goes

Rapid Application Development in Java

- Information Retrieval for multi-gigabyte text corpus (APL RTDC)
- Geoplot for distributed simulation (APL STD)



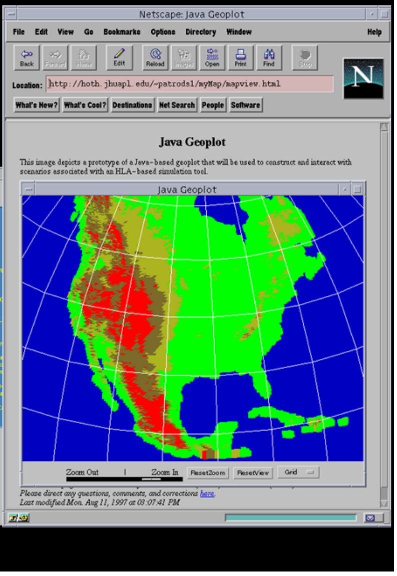
Index name: words.idx

Query terms:
Johns Hopkins
Computer Science
Java Programming

Parsed Query:
mming
Hopki
ns Co
opkin

```
Doc 41 0.4160251
Doc 40 0.2773501
Doc 39 0.1961161
Doc 38 0.3680373
Doc 37 0.6923077
Doc 36 0.3807498
Doc 35 0.4160251
Doc 34 0.2719641
Doc 33 0.1961161
Doc 32 0.2773501
Doc 31 0.2631174
Doc 29 0.1961161
Doc 28 0.1961161
Doc 27 0.1961161
Doc 26 0.1961161
Doc 25 0.438529 w
Doc 23 0.2773501
Doc 22 0.4486114
Doc 20 0.5447048
Doc 19 0.1961161
Doc 18 0.4846145
Doc 17 0.1961161
Doc 16 0.2773501
Doc 15 0.3268602
```

```
<DOC>
<DOCNO> WSJ891102-0176 </DOCNO>
<DD> = 891102 </DD>
<AN> 891102-0176. </AN>
<HL> Survival of Cray's Spinoff Will Depend
@ On Creativity and Longevity of Designer
@ ----
@ By Richard Gibson
@ Staff Reporter of The Wall Street Journal </
<DD> 11/02/89 </DD>
<SO> WALL STREET JOURNAL (J) </SO>
<CO> CYR WNEWS </CO>
<IN> COMPUTERS AND INFORMATION TECHN
<TEXT>
  The survival of spinoff Cray Computer Corp.
appears to depend heavily on the creativity --
designer, Seymour Cray.

  Not only is development of the new company
balance sheet.
Documents filed with the Securities and Excha
that Cray Research Inc. will withdraw the almo
firm if Mr. Cray leaves or if the product-design

  The documents also said that although the 6
project for more than six years, the Cray-3 ma
operational prototype.
Moreover, there have been no orders for the C
with several prospects.
```



Location: http://hoth.jhuapl.edu/~patrods1/myMap/mapview.html

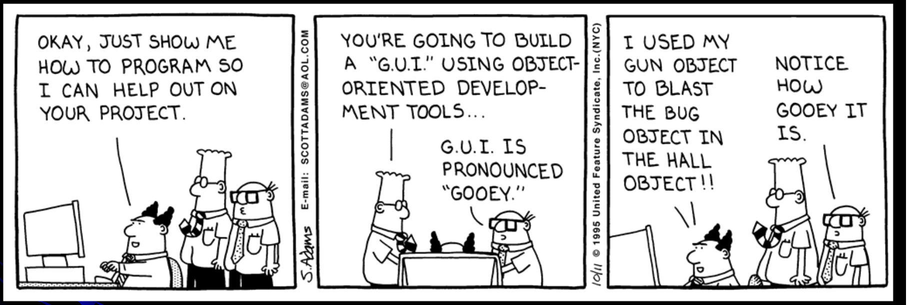
Java Geoplot

This image depicts a prototype of a Java-based geoplot that will be used to construct and interact with scenarios associated with an HLA-based simulation tool.

Please direct any questions, comments, and corrections here.
Last modified Mon. Aug 11, 1997 at 03:07:41 PM

Java is Simple?

- Myth: Java *programming* is simple



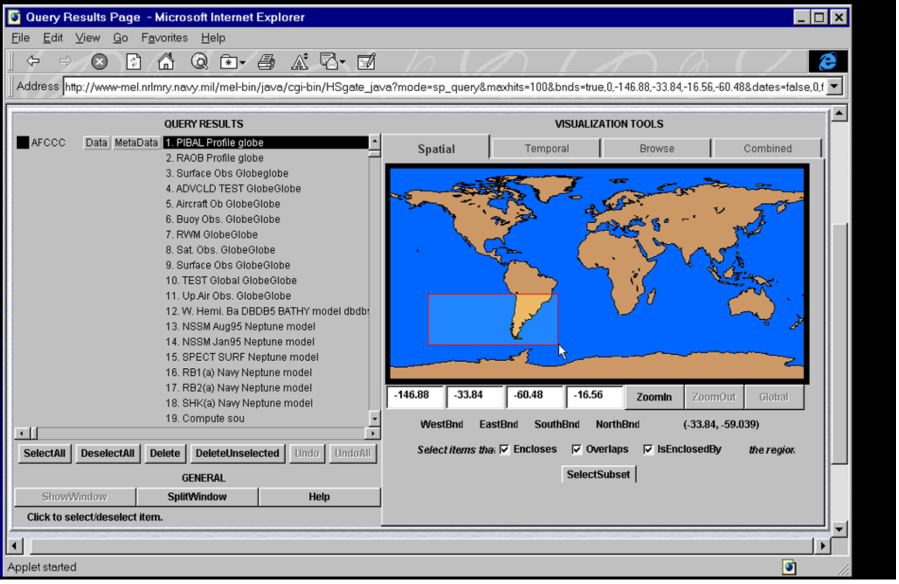
OKAY, JUST SHOW ME HOW TO PROGRAM SO I CAN HELP OUT ON YOUR PROJECT.
E-mail: SCOTTADAMS@AOL.COM
S. Adams
YOU'RE GOING TO BUILD A "G.U.I." USING OBJECT-ORIENTED DEVELOPMENT TOOLS...
G.U.I. IS PRONOUNCED "GOOEY."
© 1995 United Feature Syndicate, Inc.(NYC)
10/11
I USED MY GUN OBJECT TO BLAST THE BUG OBJECT IN THE HALL OBJECT!!
NOTICE HOW GOOEY IT IS.

- Programming is always hard
  - Java is nothing like HTML;   only a little bit like JavaScript
- Programmers typically push complexity envelope
  - Multithreaded and network programming

Java is Powerful?

- Truth: Java has a rich set of standard libraries
  - Networking
  - Threads (lightweight processes)
  - Distributed objects
  - Database access
  - Graphics: GUI controls and drawing
  - Data structure library
  - Arbitrary precision integral and fixed-point arithmetic
  - Digital signatures
  - Serialization (transmitting/reassembling data structures)
  - File and stream compression

MEL - Master Environmental Library (DMSO)



Query Results Page - Microsoft Internet Explorer

Address http://www-mel.nrlmry.navy.mil/mel-bin/java/cgi-bin/HSgate_java?mode=sp_query&maxhits=100&bnds=true,0,-146.88,-33.84,-16.56,-60.48&dates=false,0,f

QUERY RESULTS

AFCCC

1. PIBAL Profile globe
2. RAOB Profile globe
3. Surface Obs Globeglobe
4. ADVCLD TEST GlobeGlobe
5. Aircraft Ob GlobeGlobe
6. Buoy Obs. GlobeGlobe
7. RWM GlobeGlobe
8. Sat. Obs. GlobeGlobe
9. Surface Obs GlobeGlobe
10. TEST Global GlobeGlobe
11. Up.Air Obs. GlobeGlobe
12. W. Hemi. Ba DBDB5 BATHY model dbdb
13. NSSM Aug95 Neptune model
14. NSSM Jan95 Neptune model
15. SPECT SURF Neptune model
16. RB1(a) Navy Neptune model
17. RB2(a) Navy Neptune model
18. SHK(a) Navy Neptune model
19. Compute sou

GENERAL

Click to select/deselect item.

VISUALIZATION TOOLS

WestBnd	EastBnd	SouthBnd	NorthBnd
-146.88	-33.84	-60.48	-16.56

(-33.84, -59.039)

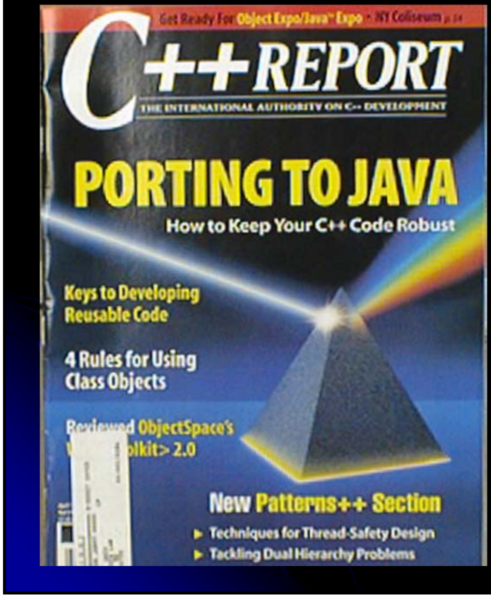
Select items tha ☑ Encloses ☑ Overlaps ☑ IsEnclosedBy the regior

Applet started

Java is Powerful?

- Myth: Java will increase programmer productivity for all applications by XXX%.
- Myth: Java will kill C++
- Myth: All software should be written in Java
  - Unix utilities: C
  - Small/medium Windows-only programs: Visual Basic
  - String parsing: Perl
  - High-performance, single-platform OO systems: C++
  - Air traffic control, aircraft flight software: Ada
  - Knowledge-based systems: Lisp/CLOS
  - Java also a good alternative for many of these

Java and C++


Get Ready For Object Expo/Java™ Expo · NY Coliseum
C++REPORT
THE INTERNATIONAL AUTHORITY ON C++ DEVELOPMENT
PORTING TO JAVA
How to Keep Your C++ Code Robust
Keys to Developing Reusable Code
4 Rules for Using Class Objects
Reviewed ObjectSpace's
olkit> 2.0
New Patterns++ Section
Techniques for Thread-Safety Design
Tackling Dual Hierarchy Problems

Although Java will certainly not kill off C++, Java and C++ do compete for some of the same territory.

Hmm, does *The C++ Report* think that the way to keep your C++ code robust is to port it to Java?

Key Java Packages and Protocols

- Core Technologies
  - JDBC
  - RMI (and Jini)
  - JavaBeans
  - Swing
  - Java 2D
- Standard Extensions
  - Servlets (and JavaServer Pages)
  - Enterprise Java Beans (and JNDI)
  - Java 3D

Java Packages and Protocols: JDBC (Java DataBase Connectivity)

- Standardizes mechanism for making connection to database server
  - Requires server-specific driver on client. No change to server.
- Standardizes mechanism for sending queries
  - Either regular or parameterized queries (stored procedures)
- Standardizes data structure of query result
  - Assumes relational data, so data structure is a table
- Does *not* standardize SQL syntax
  - Queries are simply strings
  - Server extensions and enhancements supported

Java Packages and Protocols: Remote Method Invocation (RMI)

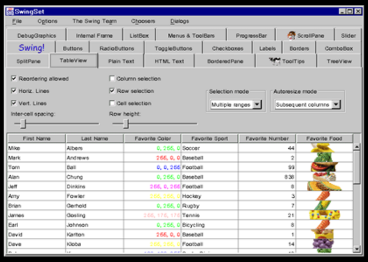
- Built-in Distributed Object Protocol
  - RMI lets a developer access a Java object and manipulate it in the normal manner. Behind the scenes, each function call really goes over the network to a remote object.
  - Arbitrary Java data structures can be sent over the network with little or no special packaging, due to Java's "serialization" mechanism
  - Similar to a simplified CORBA, but restricted to Java-to-Java communication
- Jini
  - RMI-based protocol for self-documenting services.
  - Allows real "plug and play" -- no separate drivers
  - Jury is out on eventual success. Security and industry adoption are open questions.

Java Packages and Protocols: JavaBeans

- JavaBeans is to Java as ActiveX is to Visual C++.
  - Lets you package a Java program as a software "component"
  - Visual tools can modify/manipulate it without knowing anything about it in advance
    - For example, you can drop a Bean into Visual Café, IBM VisualAge for Java, Inprise (Borland) JBuilder, Sybase PowerJ, Metrowerks CodeWarrior, Sun JavaWorkshop, etc., and it is automatically available from their tool palette for drag-and-drop development
  - Better security and portability than ActiveX
  - More ActiveX components available

Java Packages and Protocols: Swing

- Standard GUI-control (widget) library in Java 2
- Many more built-in controls
- Much more flexible and customizable
- Includes many small features aimed at commercial applications
  - Tooltips, tabbed panes, on-line help, HTML support, dockable toolbars, multi-document interface, etc.
- Look and feel can be changed at run time

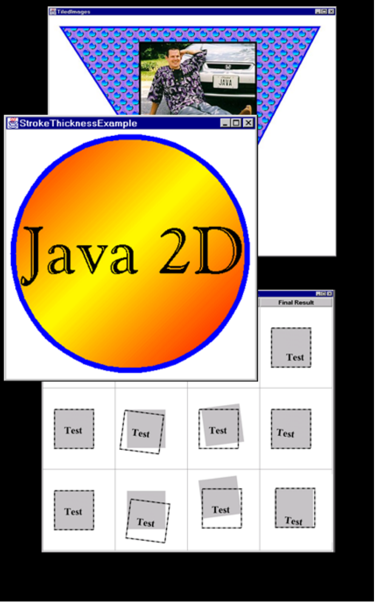


☑ Reordering allowed
☑ Horiz. Lines
☑ Vert. Lines
Inter-cell spacing:
☐ Column selection
☑ Row selection
☐ Cell selection
Row height:
Selection mode: Multiple ranges
Autoresize mode: Subsequent columns

First Name	Last Name	Favorite Color	Favorite Sport	Favorite Number	Favorite Food
Mike	Albers	0, 255, 0	Soccer	44	
Mark	Andrews	255, 0, 0	Baseball	2	
Tom	Ball	0, 0, 255	Football	99	
Alan	Chung	0, 255, 0	Baseball	838	
Jeff	Dinkins	255, 0, 255	Football	8	
Amy	Fowler	255, 255, 0	Hockey	3	
Brian	Gerhold	0, 255, 0	Rugby	7	
James	Gosling	255, 175, 175	Tennis	21	
Earl	Johnson	0, 255, 0	Bicycling	8	
David	Karlton	255, 0, 0	Baseball	1	
Dave	Kloba	255, 255, 0	Football	14	

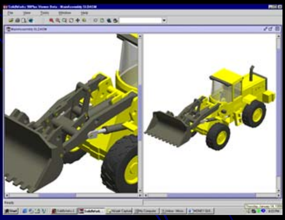
Java Packages and Protocols: Java 2D

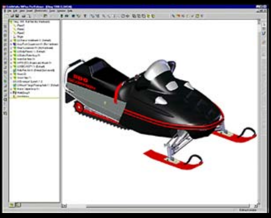
- Standard drawing library in Java 2
- Many new drawing attributes
  - Fill patterns and images
  - Arbitrary fonts
  - Pen thicknesses and dashing patterns
  - Color mixing rules and transparency
- Coordinate transformations
  - Floating-point coordinate system
  - Mapping from memory coords to screen or printer coords
  - Affine transforms: translate, scale, rotate, and shear



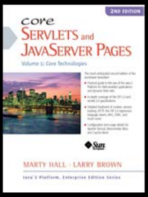
Java Packages and Protocols: Java 3D

- Standard extension to Java
  - Not part of “core” Java language like Java 2D
- Built on top of Direct3D or OpenGL, depending on platform
- Scene-graph based model, not primarily immediate-mode rendering





Java Packages and Protocols: Servlets and JavaServer Pages (JSP)



- Servlets: Java’s answer to CGI
  - Efficient: thread, not process, per request
  - Convenient: HTTP headers, cookies, etc.
  - Powerful: persistence, session tracking, etc.
  - Secure: no buffer overflows or shell escapes
- Supported by virtually all Web servers:
  - Native support: Netscape/iPlanet, IBM WebSphere, Oracle 8i/9i and Oracle Application Server, BEA WebLogic, Silverstream, Sapphire/Web, etc.
  - Via add-on engine: Apache, Microsoft IIS and Personal WebServer, Netscape FastTrack, O’Reilly WebSite, StarNine WebSTAR for MacOS, etc.
- JavaServer Pages (JSP)
  - Convenient and efficient way to combine servlets and HTML. Portable alternative to ASP & ColdFusion.

JAVASERVER PAGES™
DYNAMICALLY GENERATED WEB CONTENT

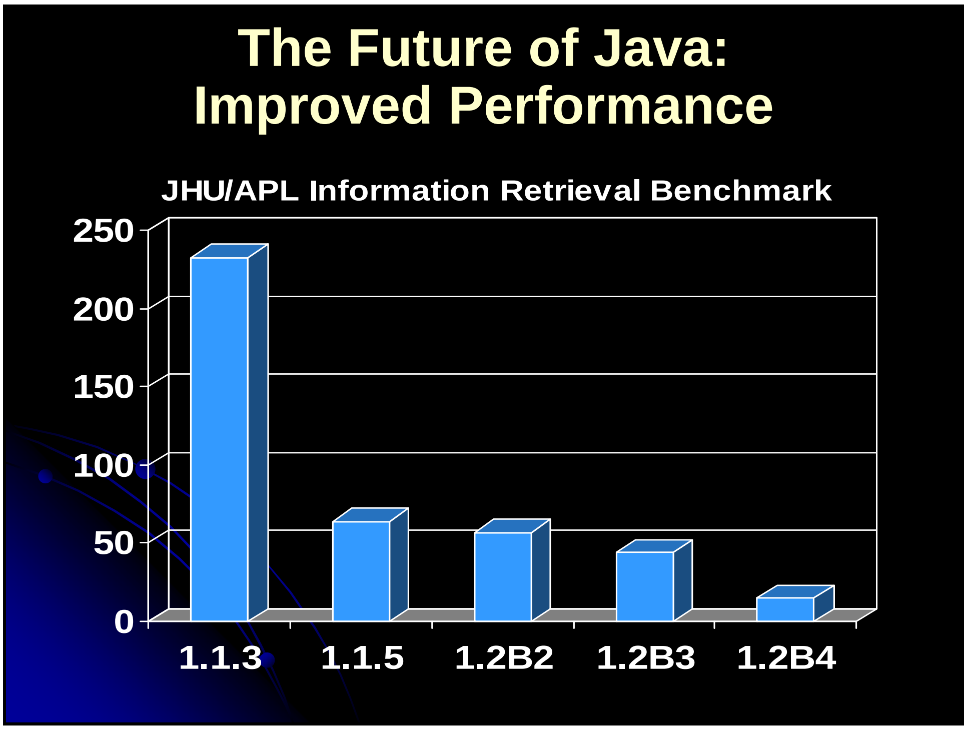
Java Packages and Protocols: Enterprise JavaBeans (EJB)

- EJBs are to server components what regular JavaBeans are to application components
- Standardizes access to services like load balancing, persistence, failover, etc.

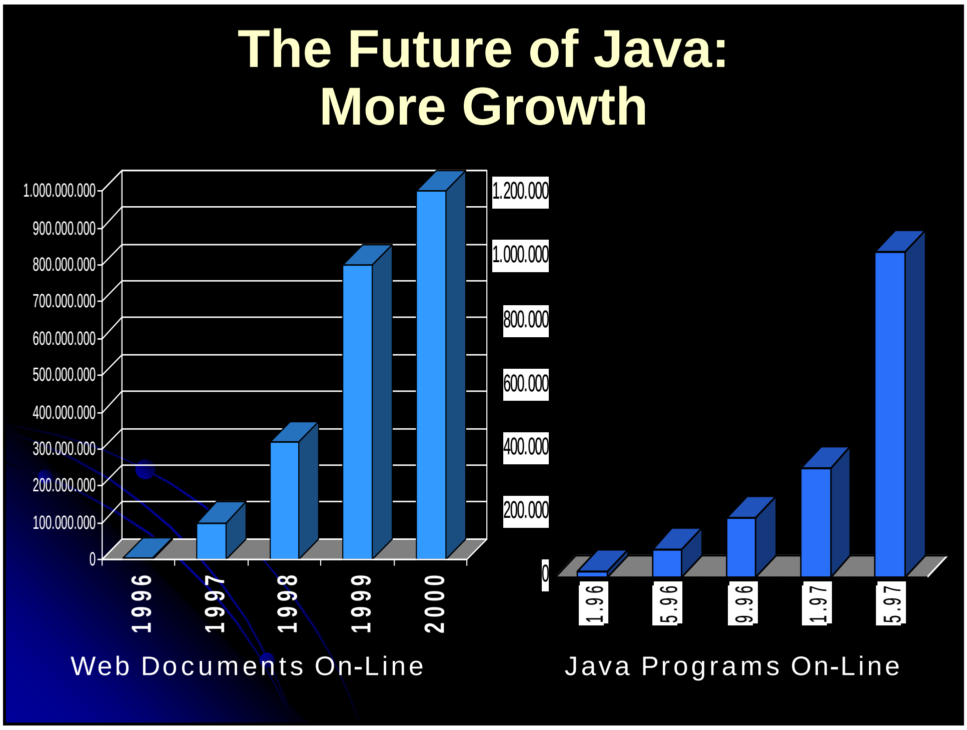
- Builds on JavaBeans, CORBA, and RMI "under the hood"
- Potentially accessible via non-Java programs
- Application Servers Supporting EJB
  - BEA WebLogic, IBM WebSphere, Netscape/iPlanet, Oracle, Progress SW Apptivity, NetDynamics, Sybase, Bluestone, Saphire/Web etc.

The Future of Java

- Core language
  - Java 2 (aka JDK 1.2-1.5) released for Windows in Dec '98. Richer set of GUI controls, better drawing model, extensive data structure library ("collections"), better audio support, standard CORBA interface, better performance. Core language evolution slowed.
- Standard extensions
  - Servlets, JSP, Jini, JAXP, etc. Continue to evolve rapidly.
- Java on the server: current growth is here
- Java for small devices and embedded apps
  - Java 2 Micro Edition (PDAs, cell phones, etc.), JavaCard
  - Future of Real-Time Java is still unknown (www.rtj.org)
- Legal battles over?



The Future of Java: Improved Performance



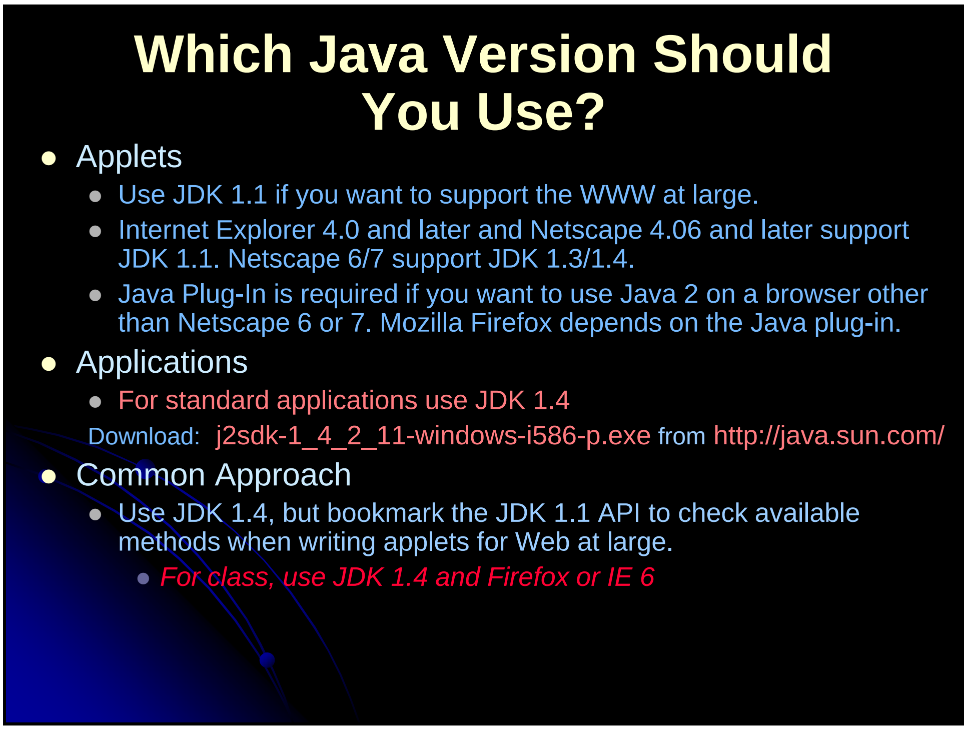
The Future of Java: More Growth

The Future of Java: More Jobs

- Even in economic downturn, most companies that do large amounts of software development have shortages of good Java developers
- IBM has over 2,500 professionals involved with Java product development
- Seen on a blackboard in the background of a video clip at the JavaOne conference:

```
if (you.canRead(this))
  you.canGet(new Job(!problem));
```

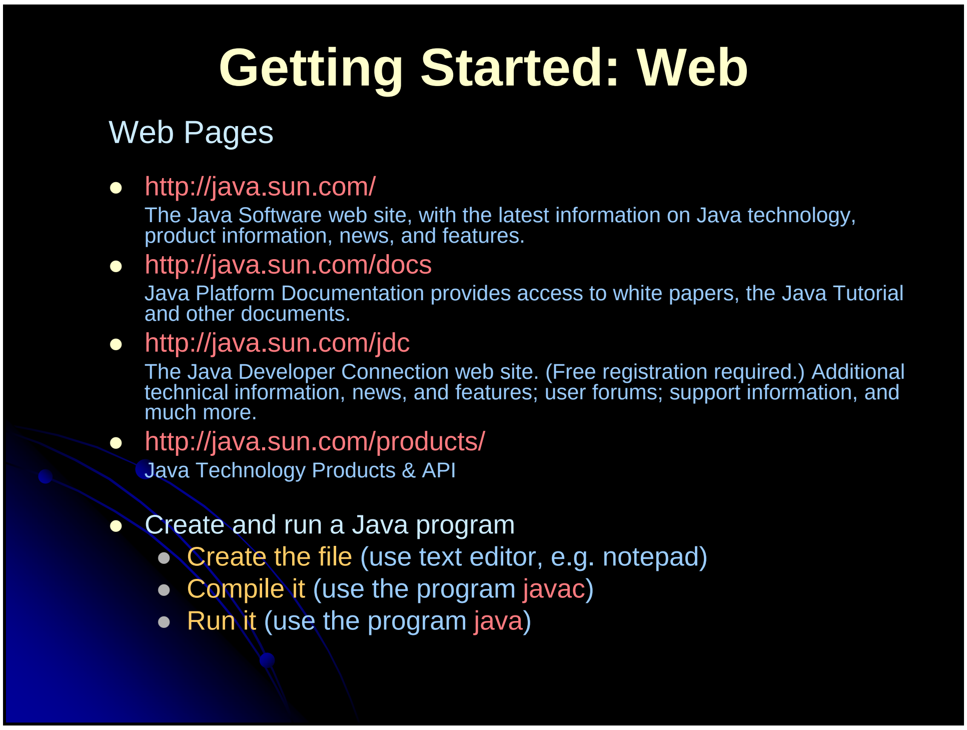
The Future of Java: Java is Driving the Software Industry



Which Java Version Should You Use?

- Applets
  - Use JDK 1.1 if you want to support the WWW at large.
  - Internet Explorer 4.0 and later and Netscape 4.06 and later support JDK 1.1. Netscape 6/7 support JDK 1.3/1.4.
  - Java Plug-In is required if you want to use Java 2 on a browser other than Netscape 6 or 7. Mozilla Firefox depends on the Java plug-in.
- Applications
  - For standard applications use JDK 1.4

  Download: j2sdk-1_4_2_11-windows-i586-p.exe from http://java.sun.com/
- Common Approach
  - Use JDK 1.4, but bookmark the JDK 1.1 API to check available methods when writing applets for Web at large.
    - *For class, use JDK 1.4 and Firefox or IE 6*



Getting Started: Web

Web Pages

- http://java.sun.com/

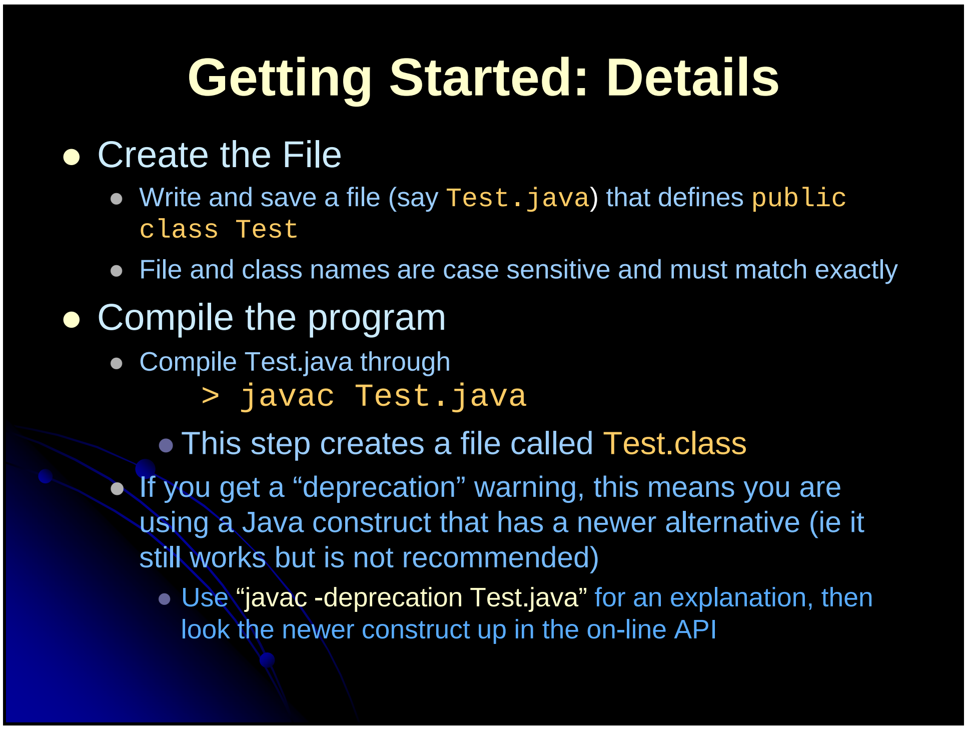
  The Java Software web site, with the latest information on Java technology, product information, news, and features.
- http://java.sun.com/docs

  Java Platform Documentation provides access to white papers, the Java Tutorial and other documents.
- http://java.sun.com/jdc

  The Java Developer Connection web site. (Free registration required.) Additional technical information, news, and features; user forums; support information, and much more.
- http://java.sun.com/products/

  Java Technology Products & API

- Create and run a Java program
  - Create the file (use text editor, e.g. notepad)
  - Compile it (use the program javac)
  - Run it (use the program java)

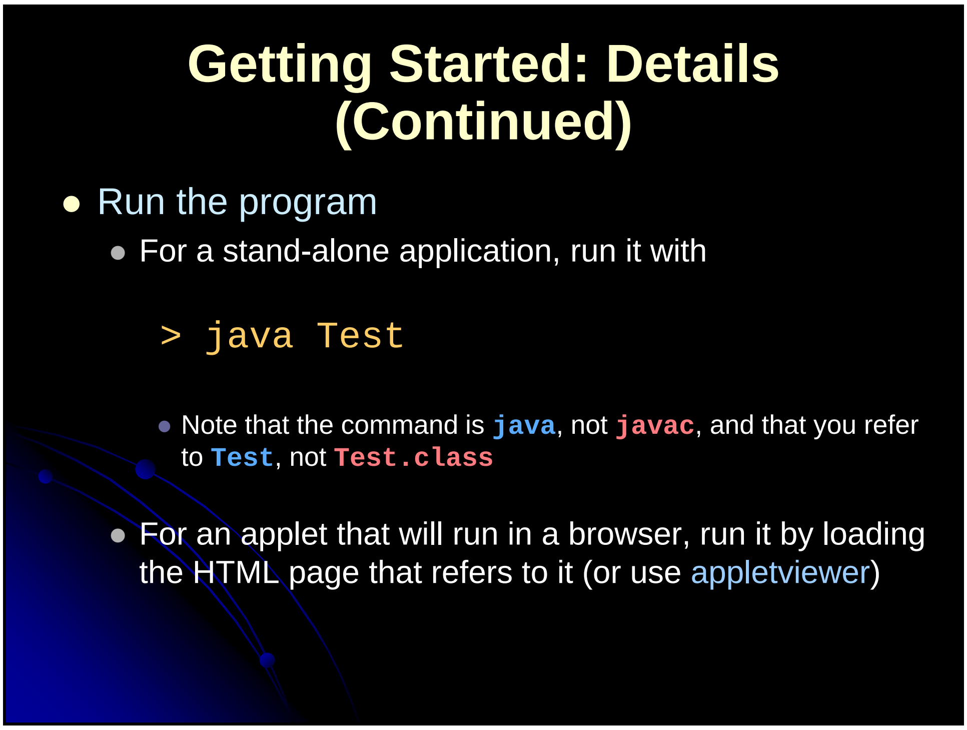


Getting Started: Details

- Create the File
  - Write and save a file (say `Test.java`) that defines `public class Test`
  - File and class names are case sensitive and must match exactly
- Compile the program
  - Compile Test.java through

```
> javac Test.java
```

    - This step creates a file called Test.class
  - If you get a “deprecation” warning, this means you are using a Java construct that has a newer alternative (ie it still works but is not recommended)
    - Use “javac -deprecation Test.java” for an explanation, then look the newer construct up in the on-line API

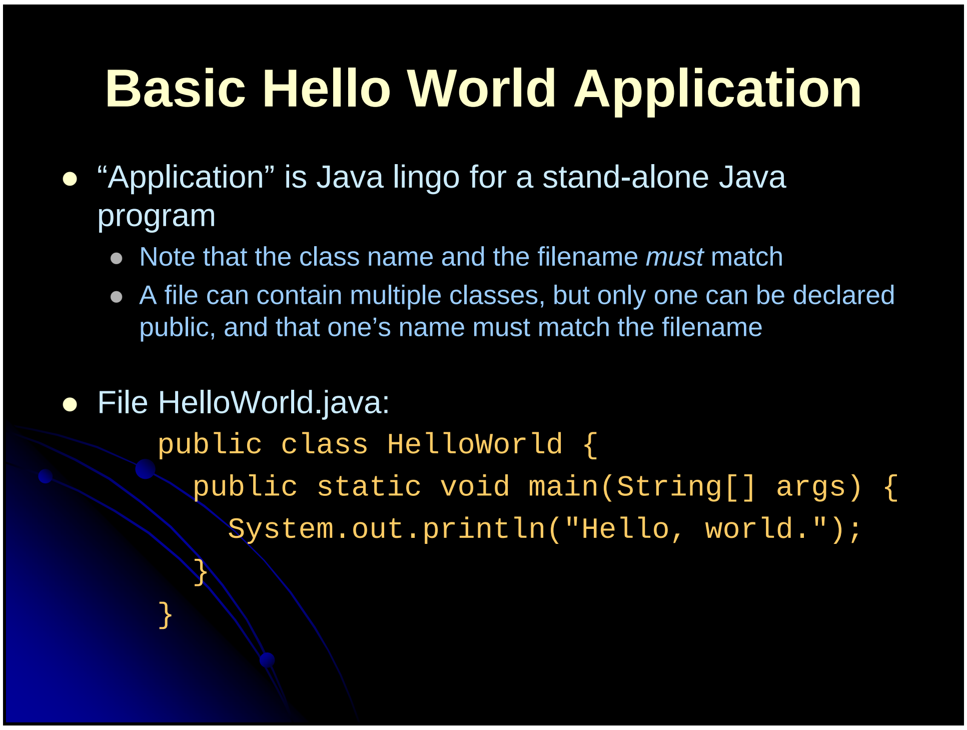


Getting Started: Details (Continued)

- Run the program
  - For a stand-alone application, run it with

```
> java Test
```

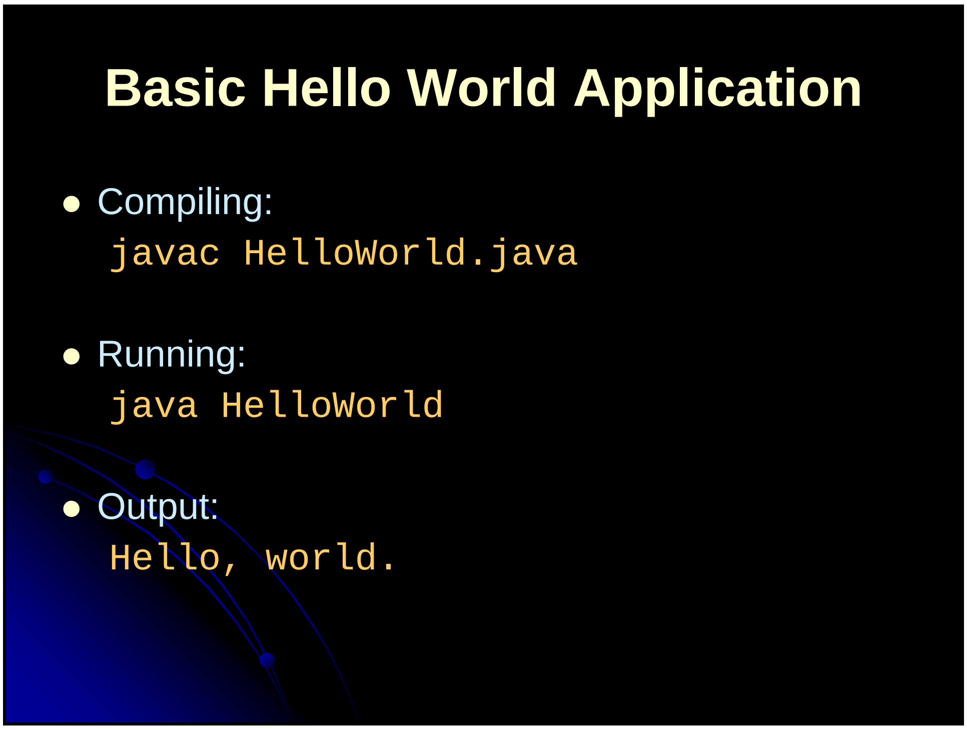
    - Note that the command is **`java`**, not **`javac`**, and that you refer to **`Test`**, not **`Test.class`**
  - For an applet that will run in a browser, run it by loading the HTML page that refers to it (or use appletviewer)



Basic Hello World Application

- “Application” is Java lingo for a stand-alone Java program
  - Note that the class name and the filename *must* match
  - A file can contain multiple classes, but only one can be declared public, and that one’s name must match the filename
- File HelloWorld.java:

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world.");
  }
}
```



Basic Hello World Application

- Compiling:

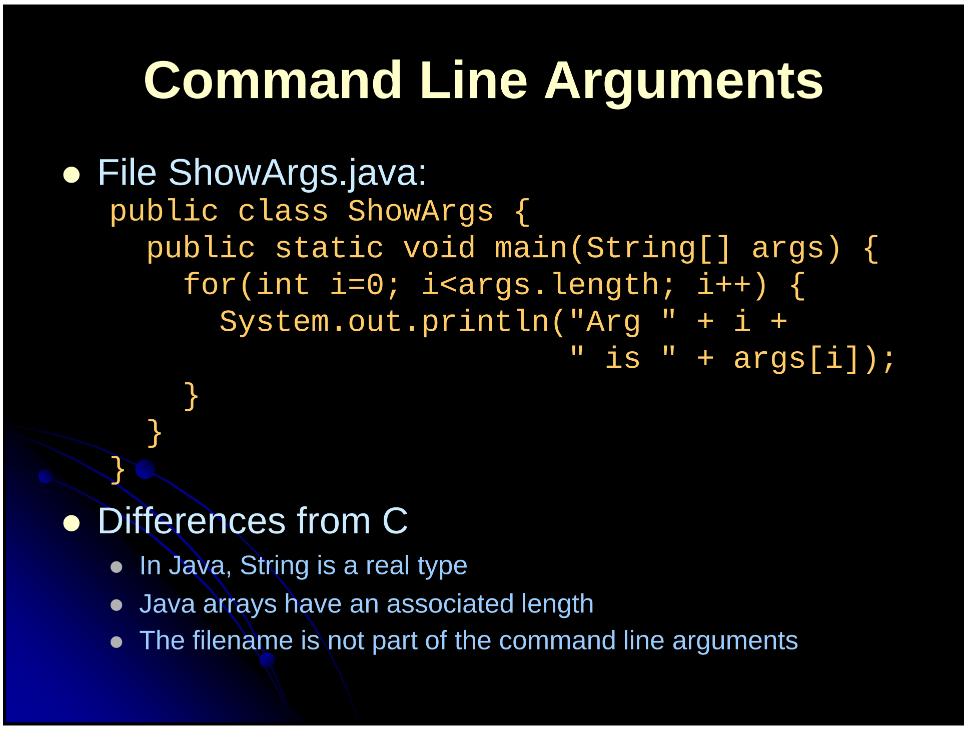
```
javac HelloWorld.java
```

- Running:

```
java HelloWorld
```

- Output:

```
Hello, world.
```

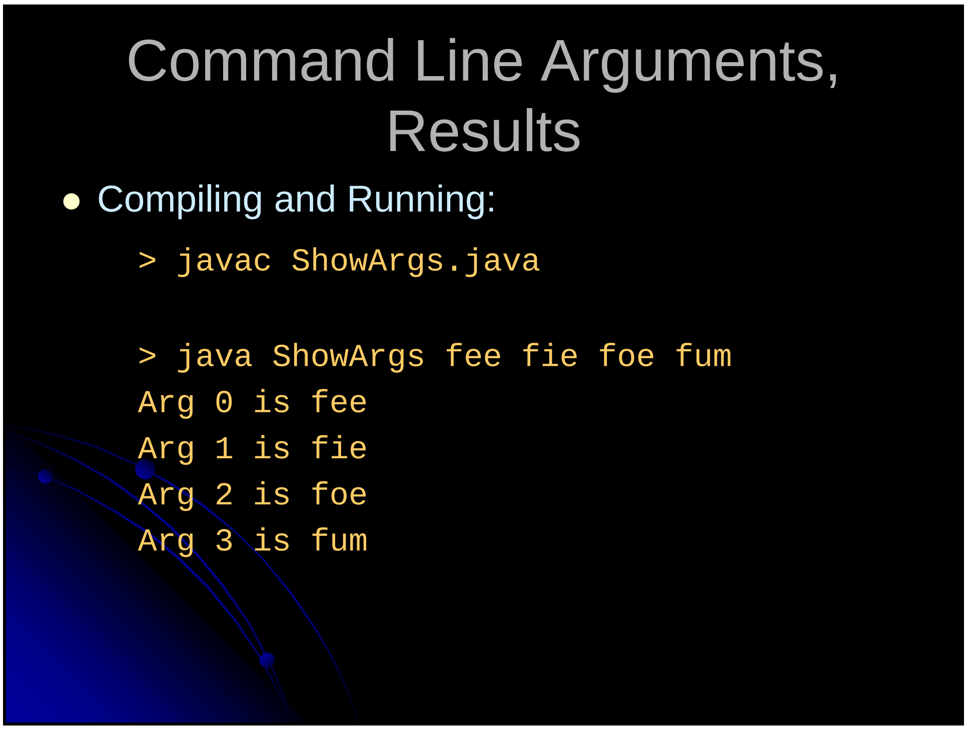


Command Line Arguments

- File ShowArgs.java:

```
public class ShowArgs {
  public static void main(String[] args) {
    for(int i=0; i<args.length; i++) {
      System.out.println("Arg " + i +
                         " is " + args[i]);
    }
  }
}
```

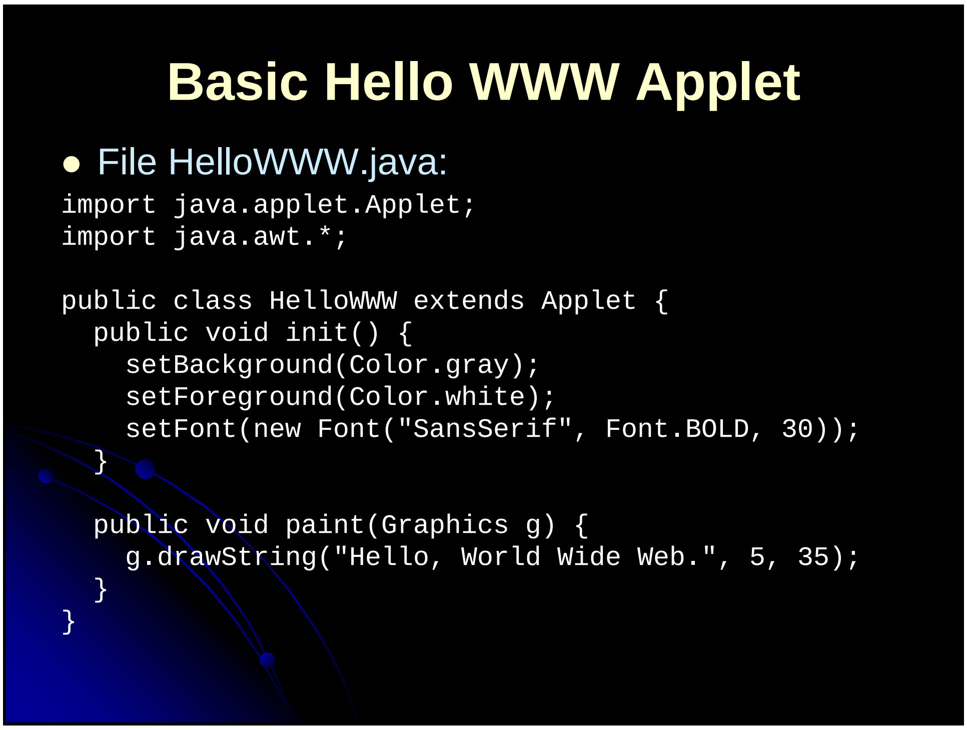
- Differences from C
  - In Java, String is a real type
  - Java arrays have an associated length
  - The filename is not part of the command line arguments



Command Line Arguments, Results

- Compiling and Running:

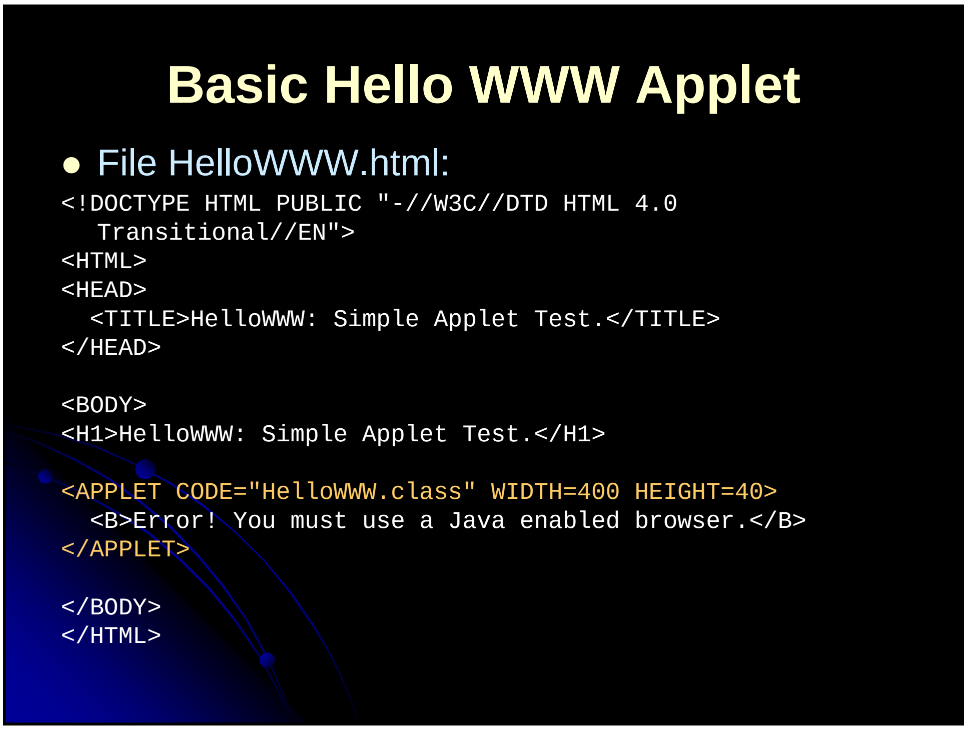
```
> javac ShowArgs.java

> java ShowArgs fee fie foe fum
Arg 0 is fee
Arg 1 is fie
Arg 2 is foe
Arg 3 is fum
```



Basic Hello WWW Applet

- File HelloWWW.java:

```
import java.applet.Applet;
import java.awt.*;

public class HelloWWW extends Applet {
  public void init() {
    setBackground(Color.gray);
    setForeground(Color.white);
    setFont(new Font("SansSerif", Font.BOLD, 30));
  }

  public void paint(Graphics g) {
    g.drawString("Hello, World Wide Web.", 5, 35);
  }
}
```



Basic Hello WWW Applet

- File HelloWWW.html:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD>
  <TITLE>HelloWWW: Simple Applet Test.</TITLE>
</HEAD>

<BODY>
<H1>HelloWWW: Simple Applet Test.</H1>

<APPLET CODE="HelloWWW.class" WIDTH=400 HEIGHT=40>
  <B>Error! You must use a Java enabled browser.</B>
</APPLET>

</BODY>
</HTML>
```

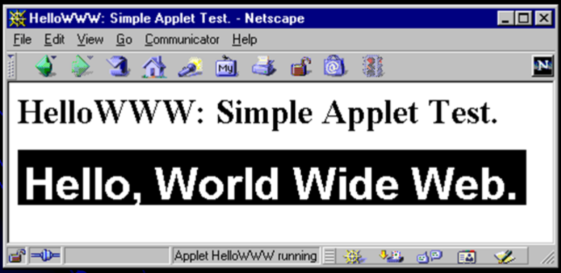
Basic Hello WWW Applet

- Compiling:

  `javac HelloWWW.java`

- Running:

  Load `HelloWWW.html` in a Java-enabled browser



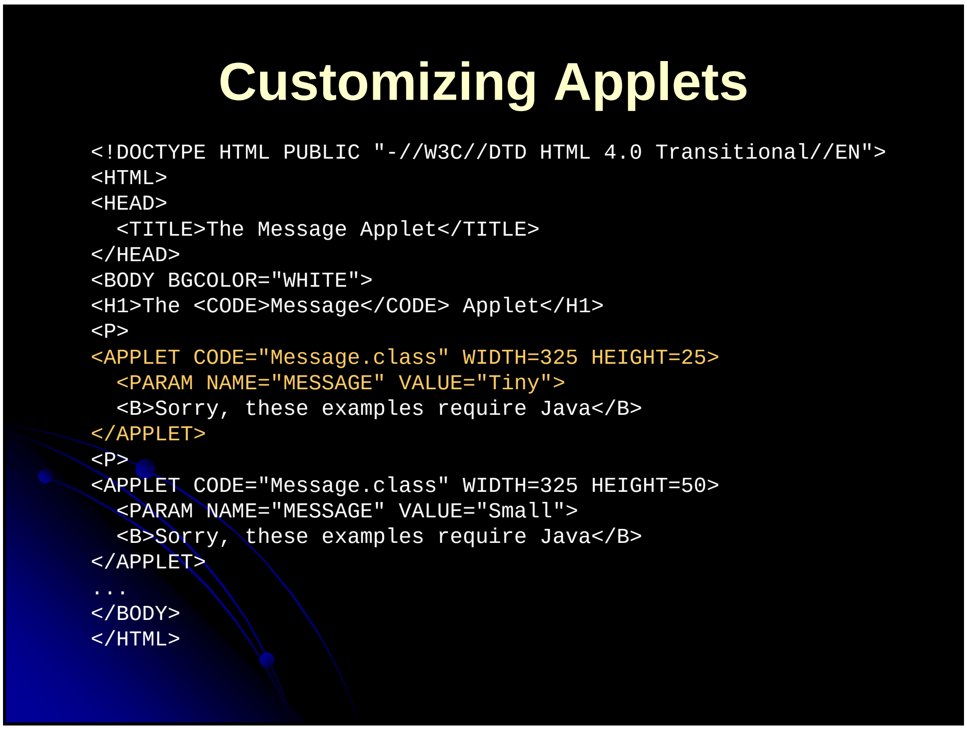
HelloWWW: Simple Applet Test.

Hello, World Wide Web.

Applet HelloWWW running

Customizing Applets

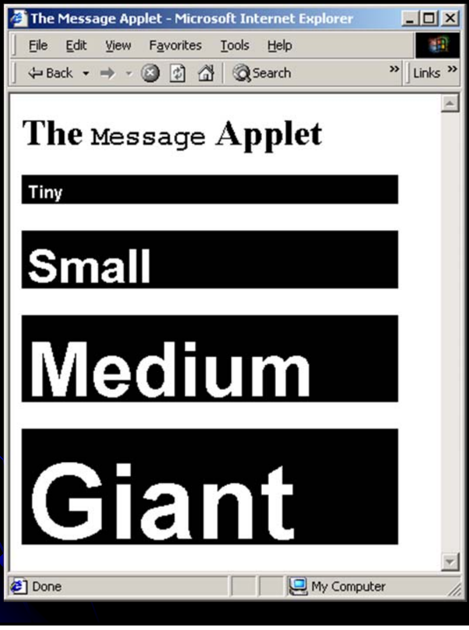
```
import java.applet.Applet;
import java.awt.*;

public class Message extends Applet {
  private int fontSize;
  private String message;

  public void init() {
    setBackground(Color.black);
    setForeground(Color.white);
    fontSize = getSize().height - 10;
    setFont(new Font("SansSerif", Font.BOLD, fontSize));
    // Read heading message from PARAM entry in HTML.
    message = getParameter("MESSAGE");
  }

  public void paint(Graphics g) {
    if (message != null)
      g.drawString(message, 5, fontSize+5);
  }
}
```



Customizing Applets

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>The Message Applet</TITLE>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>The <CODE>Message</CODE> Applet</H1>
<P>
<APPLET CODE="Message.class" WIDTH=325 HEIGHT=25>
  <PARAM NAME="MESSAGE" VALUE="Tiny">
  <B>Sorry, these examples require Java</B>
</APPLET>
<P>
<APPLET CODE="Message.class" WIDTH=325 HEIGHT=50>
  <PARAM NAME="MESSAGE" VALUE="Small">
  <B>Sorry, these examples require Java</B>
</APPLET>
...
</BODY>
</HTML>
```

Customizing Applets



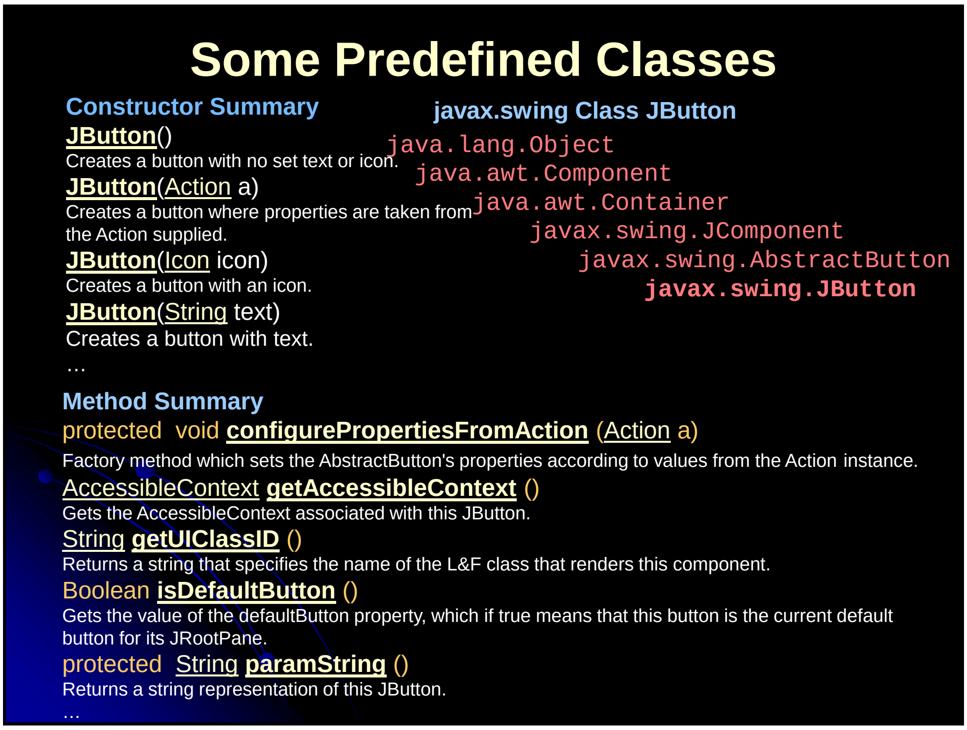
The Message Applet

Tiny

Small

Medium

Giant



Some Predefined Classes

javax.swing Class JButton

```
java.lang.Object
 java.awt.Component
   java.awt.Container
      javax.swing.JComponent
        javax.swing.AbstractButton
           javax.swing.JButton
```

Constructor Summary

JButton()
Creates a button with no set text or icon.

JButton(Action a)
Creates a button where properties are taken from the Action supplied.

JButton(Icon icon)
Creates a button with an icon.

JButton(String text)
Creates a button with text.

…

Method Summary

protected void **configurePropertiesFromAction** (Action a)
Factory method which sets the AbstractButton's properties according to values from the Action instance.

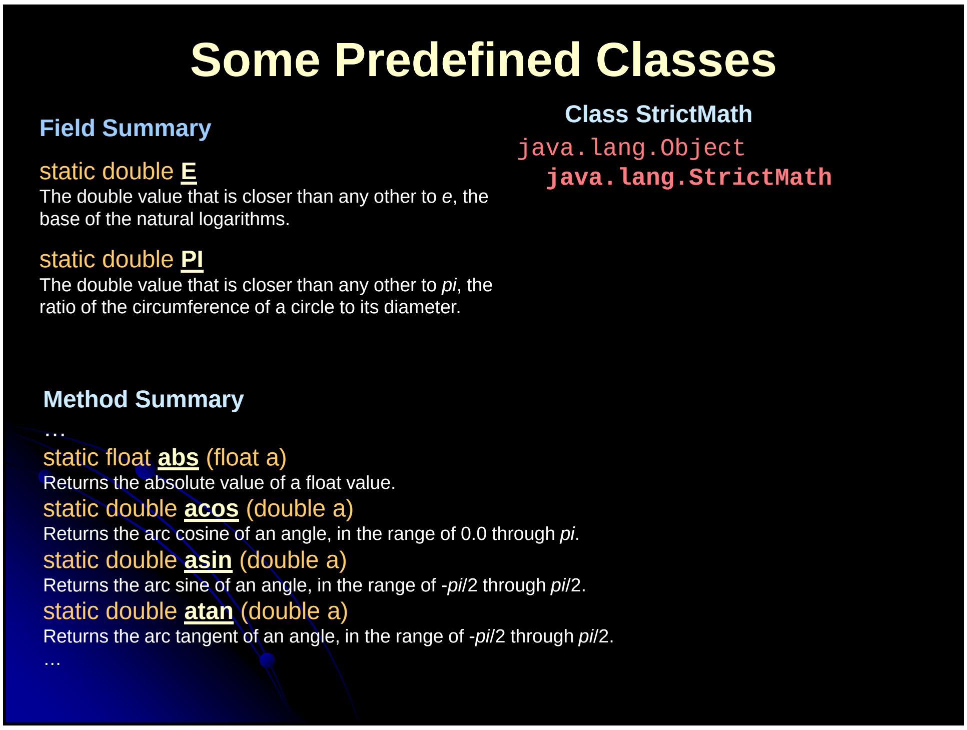
AccessibleContext **getAccessibleContext** ()
Gets the AccessibleContext associated with this JButton.

String **getUIClassID** ()
Returns a string that specifies the name of the L&F class that renders this component.

Boolean **isDefaultButton** ()
Gets the value of the defaultButton property, which if true means that this button is the current default button for its JRootPane.

protected String **paramString** ()
Returns a string representation of this JButton.

…



Some Predefined Classes

Class StrictMath

```
java.lang.Object
  java.lang.StrictMath
```

Field Summary

static double **E**
The double value that is closer than any other to *e*, the base of the natural logarithms.

static double **PI**
The double value that is closer than any other to *pi*, the ratio of the circumference of a circle to its diameter.

Method Summary

…

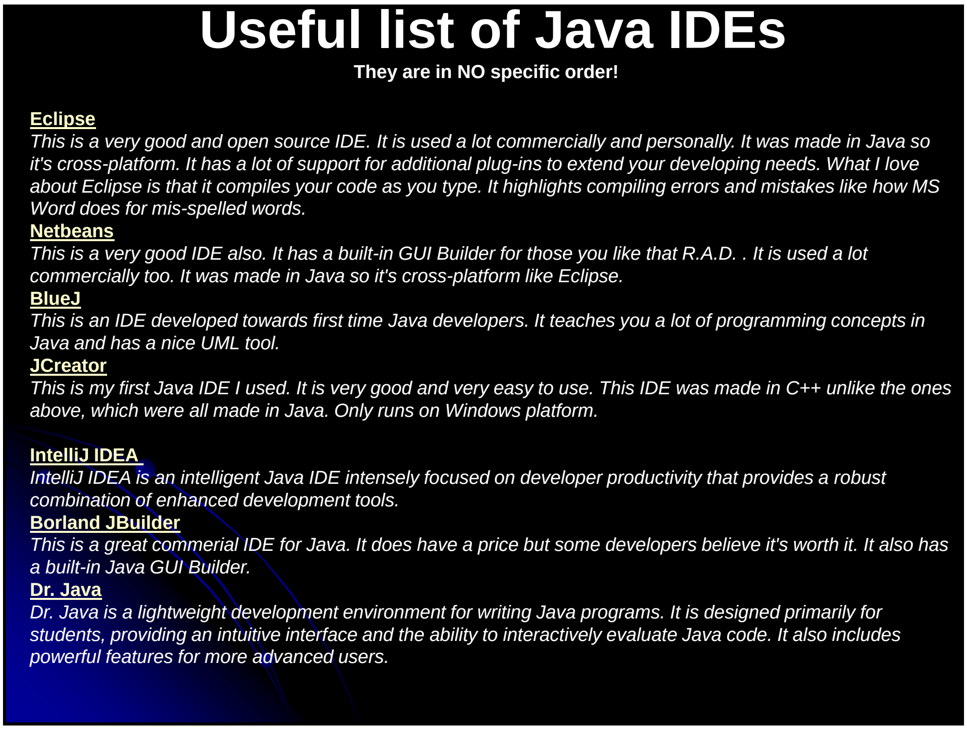
static float **abs** (float a)
Returns the absolute value of a float value.

static double **acos** (double a)
Returns the arc cosine of an angle, in the range of 0.0 through *pi*.

static double **asin** (double a)
Returns the arc sine of an angle, in the range of -*pi*/2 through *pi*/2.

static double **atan** (double a)
Returns the arc tangent of an angle, in the range of -*pi*/2 through *pi*/2.

…



Useful list of Java IDEs

They are in NO specific order!

Eclipse
This is a very good and open source IDE. It is used a lot commercially and personally. It was made in Java so it's cross-platform. It has a lot of support for additional plug-ins to extend your developing needs. What I love about Eclipse is that it compiles your code as you type. It highlights compiling errors and mistakes like how MS Word does for mis-spelled words.

Netbeans
This is a very good IDE also. It has a built-in GUI Builder for those you like that R.A.D. . It is used a lot commercially too. It was made in Java so it's cross-platform like Eclipse.

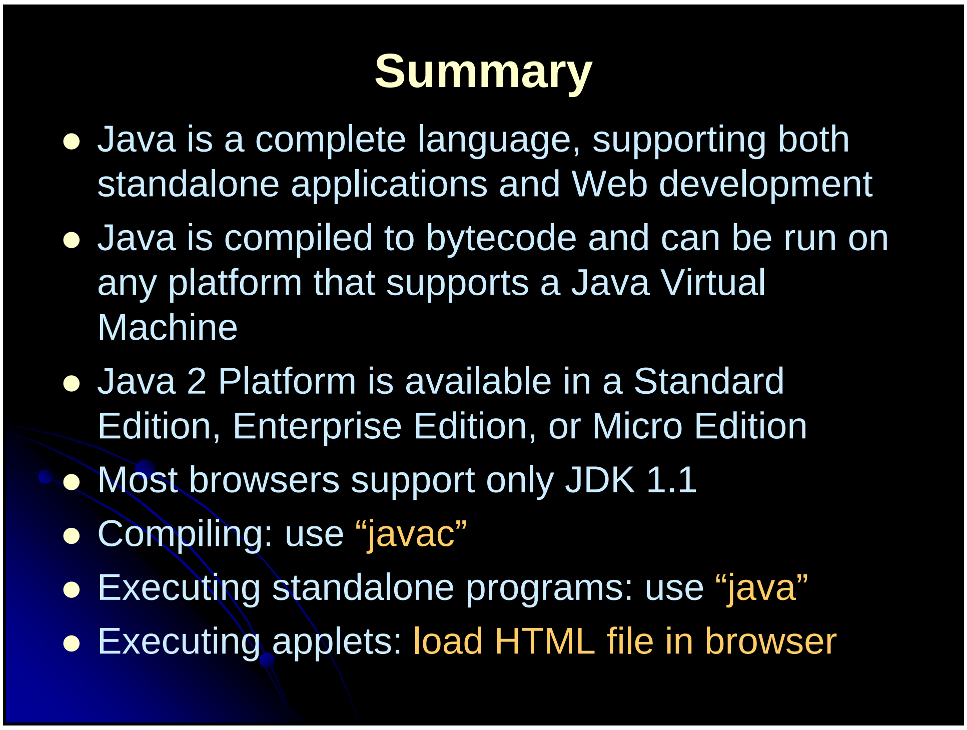
BlueJ
This is an IDE developed towards first time Java developers. It teaches you a lot of programming concepts in Java and has a nice UML tool.

JCreator
This is my first Java IDE I used. It is very good and very easy to use. This IDE was made in C++ unlike the ones above, which were all made in Java. Only runs on Windows platform.

IntelliJ IDEA
IntelliJ IDEA is an intelligent Java IDE intensely focused on developer productivity that provides a robust combination of enhanced development tools.

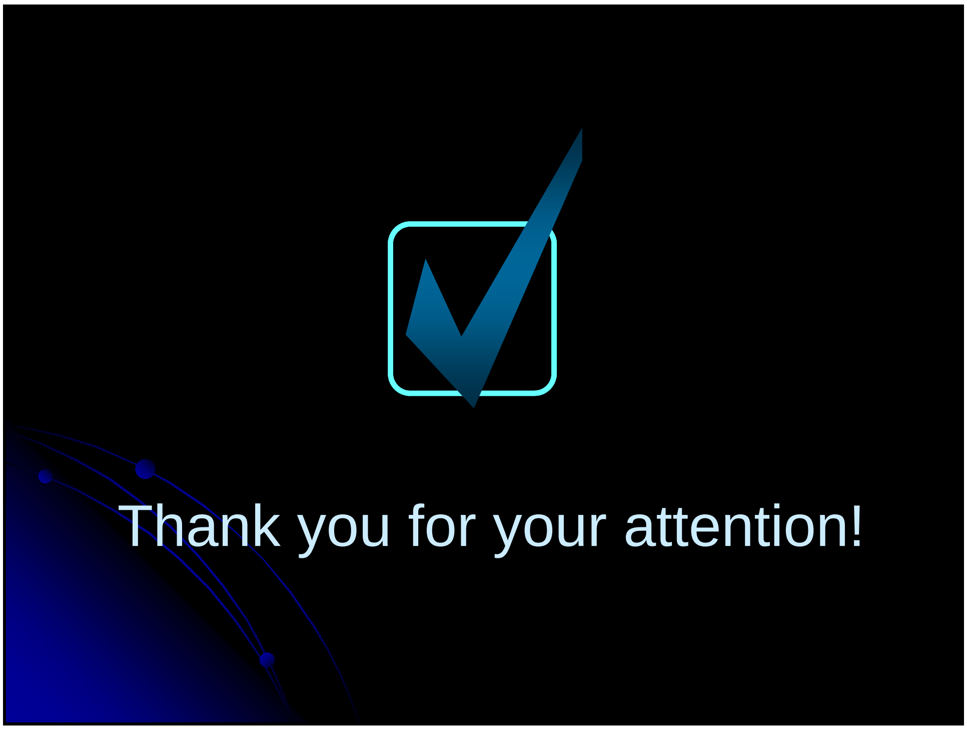
Borland JBuilder
This is a great commerial IDE for Java. It does have a price but some developers believe it's worth it. It also has a built-in Java GUI Builder.

Dr. Java
Dr. Java is a lightweight development environment for writing Java programs. It is designed primarily for students, providing an intuitive interface and the ability to interactively evaluate Java code. It also includes powerful features for more advanced users.

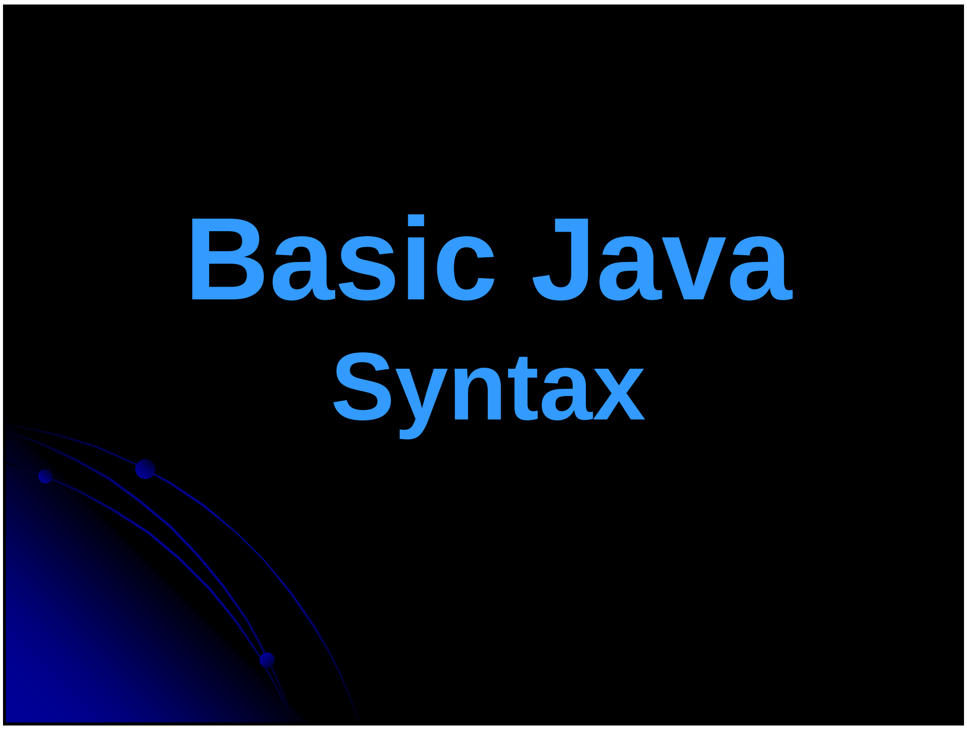


Summary

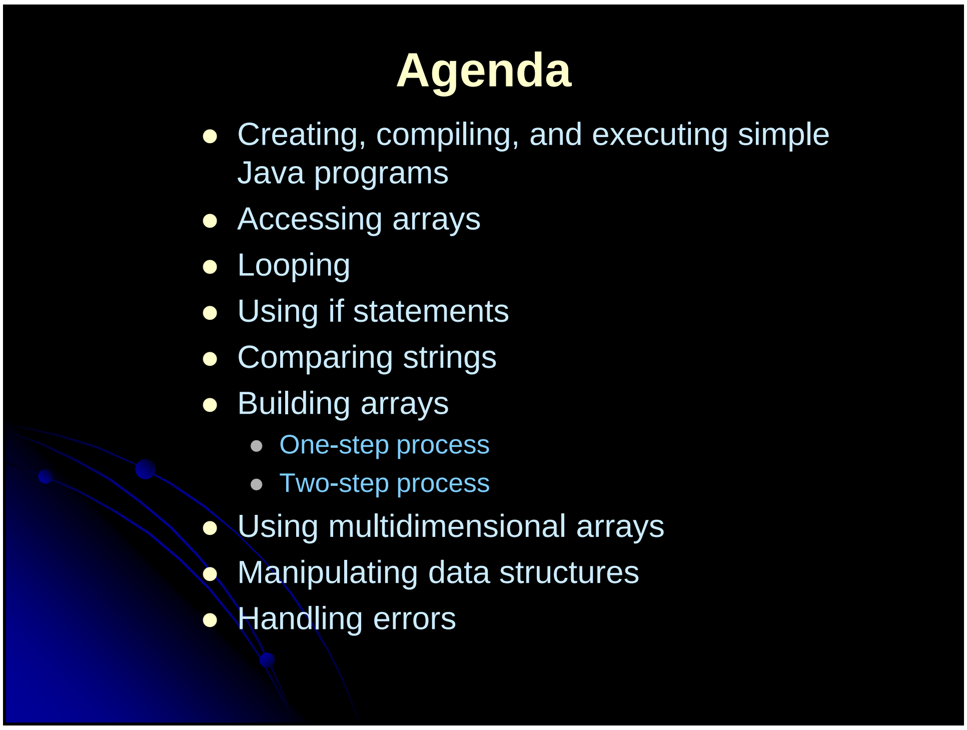
- Java is a complete language, supporting both standalone applications and Web development
- Java is compiled to bytecode and can be run on any platform that supports a Java Virtual Machine
- Java 2 Platform is available in a Standard Edition, Enterprise Edition, or Micro Edition
- Most browsers support only JDK 1.1
- Compiling: use “javac”
- Executing standalone programs: use “java”
- Executing applets: load HTML file in browser



Thank you for your attention!



Basic Java Syntax



Agenda

- Creating, compiling, and executing simple Java programs
- Accessing arrays
- Looping
- Using if statements
- Comparing strings
- Building arrays
  - One-step process
  - Two-step process
- Using multidimensional arrays
- Manipulating data structures
- Handling errors

Getting Started

- Name of file must match name of class
  - It is case sensitive, even on Windows
- Processing starts in main
  - `public static void main(String[] args)`
  - Routines usually called “methods,” not “functions.”
- Printing is done with System.out
  - System.out.println, System.out.print
- Compile with “javac”
  - Open DOS window; work from there
  - Supply full case-sensitive file name (with file extension)
- Execute with “java”
  - Supply base class name (no file extension)

Example

- File: HelloWorld.java

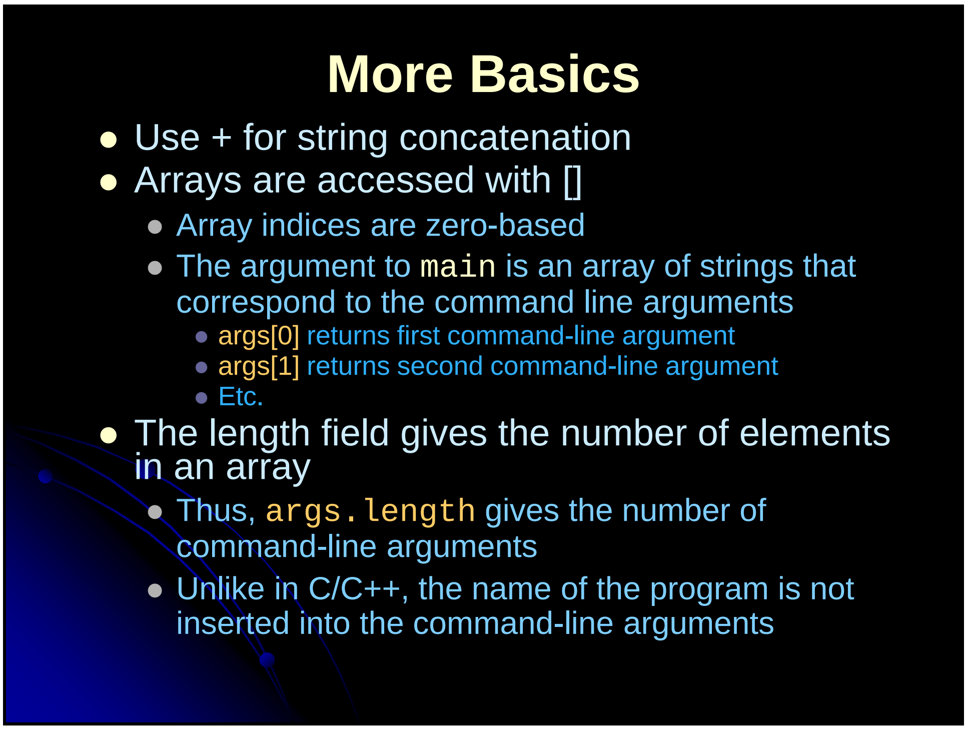
```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, world.");
  }
}
```

- Compiling

```
DOS> javac HelloWorld.java
```

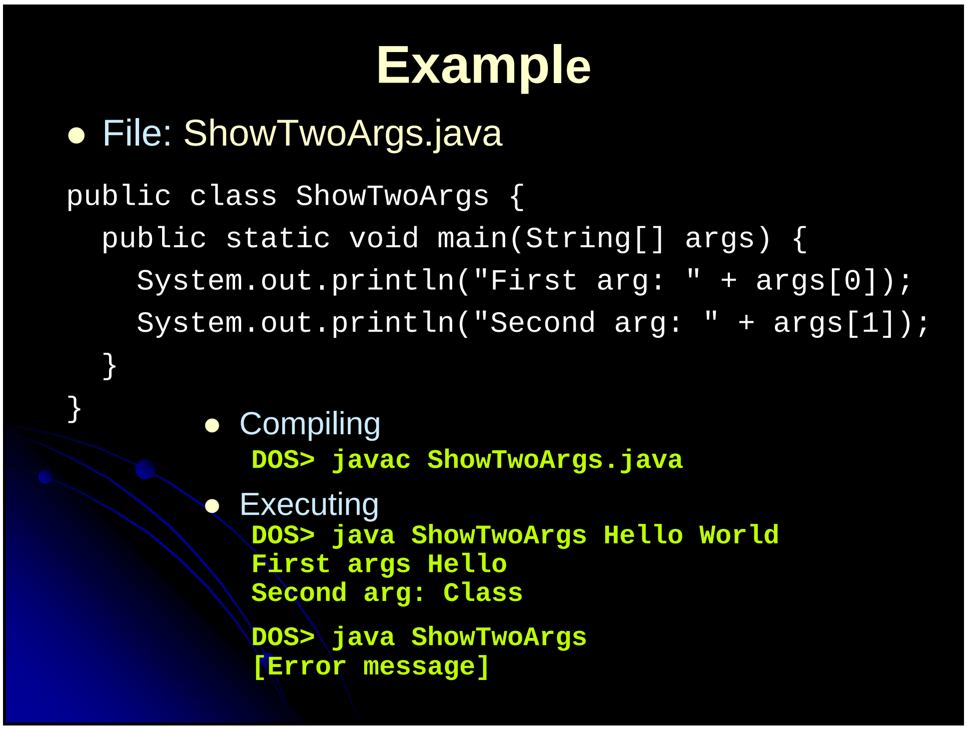
- Executing

```
DOS> java HelloWorld
Hello, world.
```



More Basics

- Use + for string concatenation
- Arrays are accessed with []
  - Array indices are zero-based
  - The argument to `main` is an array of strings that correspond to the command line arguments
    - args[0] returns first command-line argument
    - args[1] returns second command-line argument
    - Etc.
- The length field gives the number of elements in an array
  - Thus, `args.length` gives the number of command-line arguments
  - Unlike in C/C++, the name of the program is not inserted into the command-line arguments



Example

- File: ShowTwoArgs.java

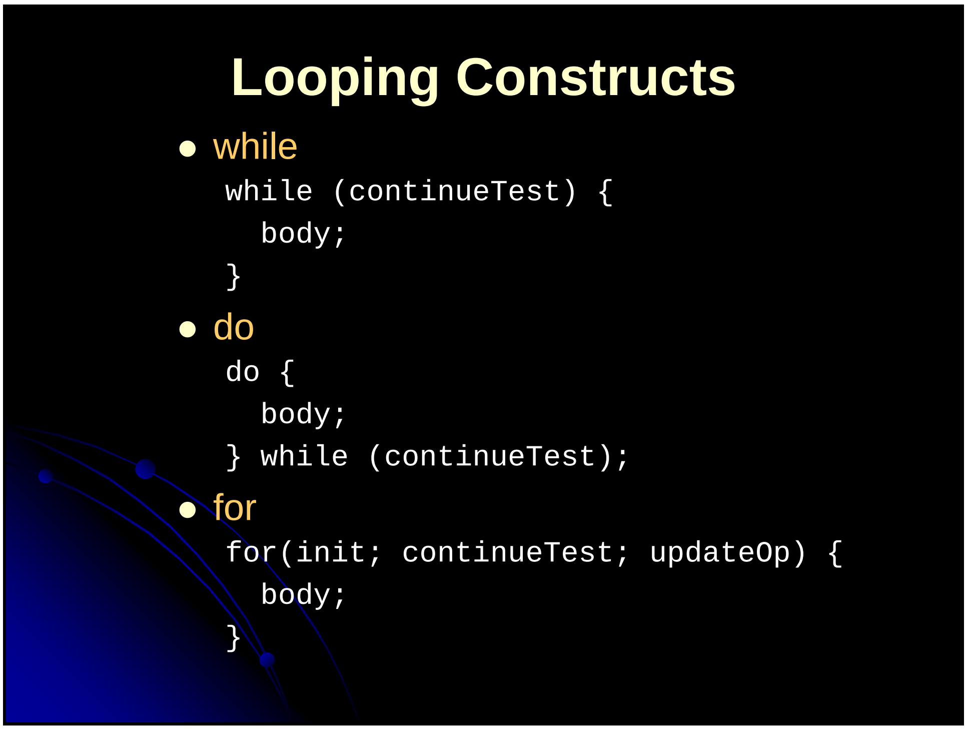
```
public class ShowTwoArgs {
  public static void main(String[] args) {
    System.out.println("First arg: " + args[0]);
    System.out.println("Second arg: " + args[1]);
  }
}
```

- Compiling

```
DOS> javac ShowTwoArgs.java
```

- Executing

```
DOS> java ShowTwoArgs Hello World
First args Hello
Second arg: Class

DOS> java ShowTwoArgs
[Error message]
```



Looping Constructs

- while

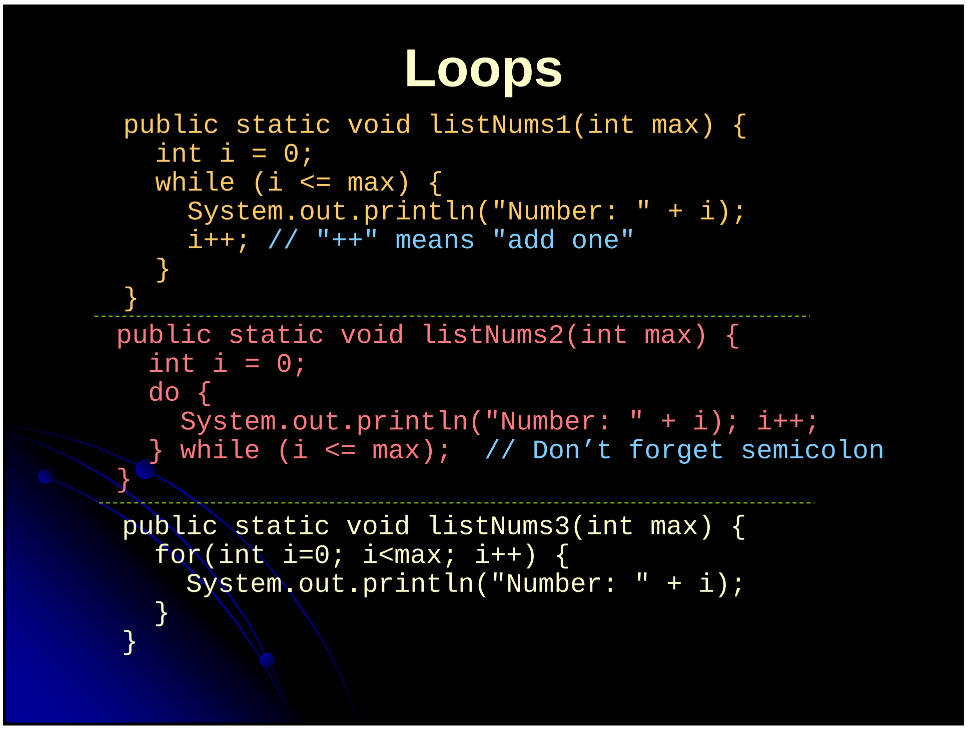
```
while (continueTest) {
  body;
}
```

- do

```
do {
  body;
} while (continueTest);
```

- for

```
for(init; continueTest; updateOp) {
  body;
}
```

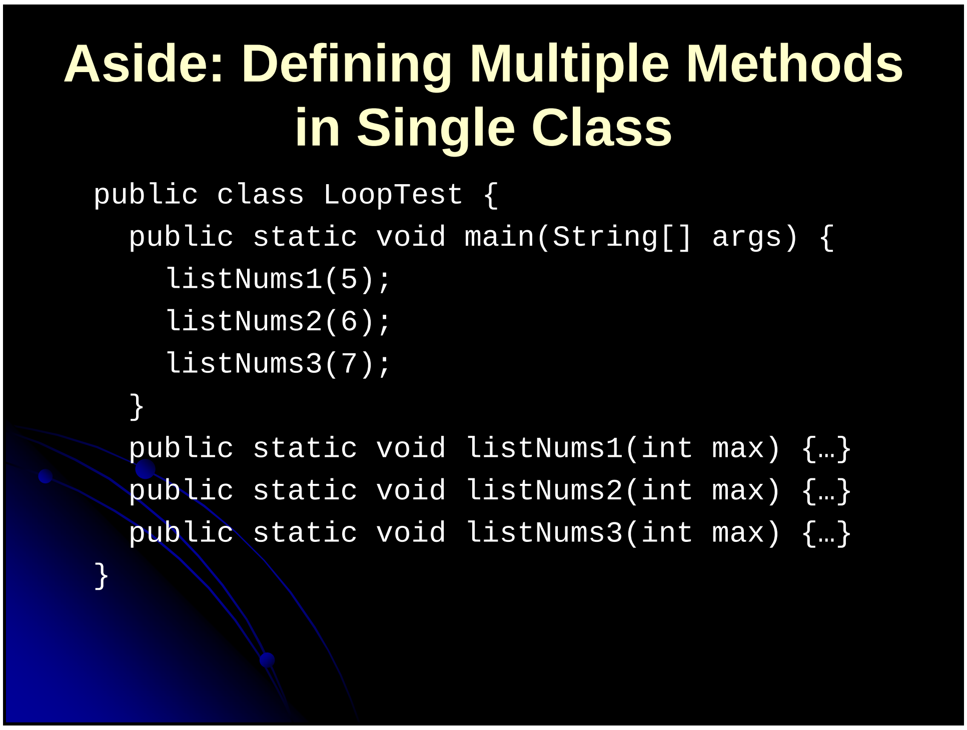


Loops

```
public static void listNums1(int max) {
  int i = 0;
  while (i <= max) {
    System.out.println("Number: " + i);
    i++; // "++" means "add one"
  }
}
```

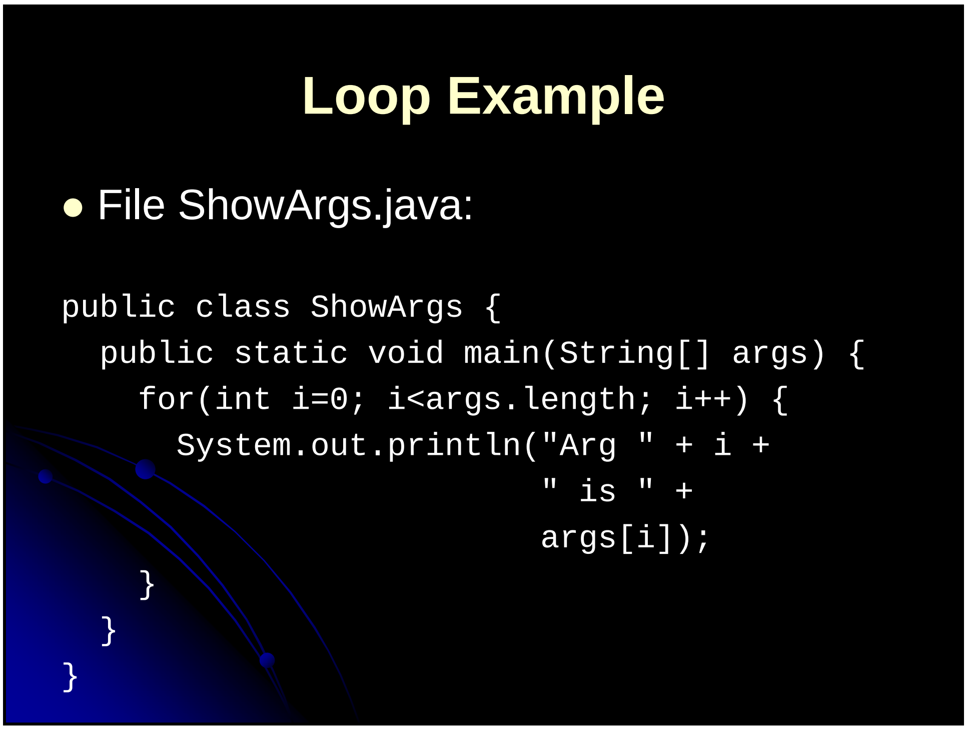
```
public static void listNums2(int max) {
  int i = 0;
  do {
    System.out.println("Number: " + i); i++;
  } while (i <= max);  // Don’t forget semicolon
}
```

```
public static void listNums3(int max) {
  for(int i=0; i<max; i++) {
    System.out.println("Number: " + i);
  }
}
```



Aside: Defining Multiple Methods in Single Class

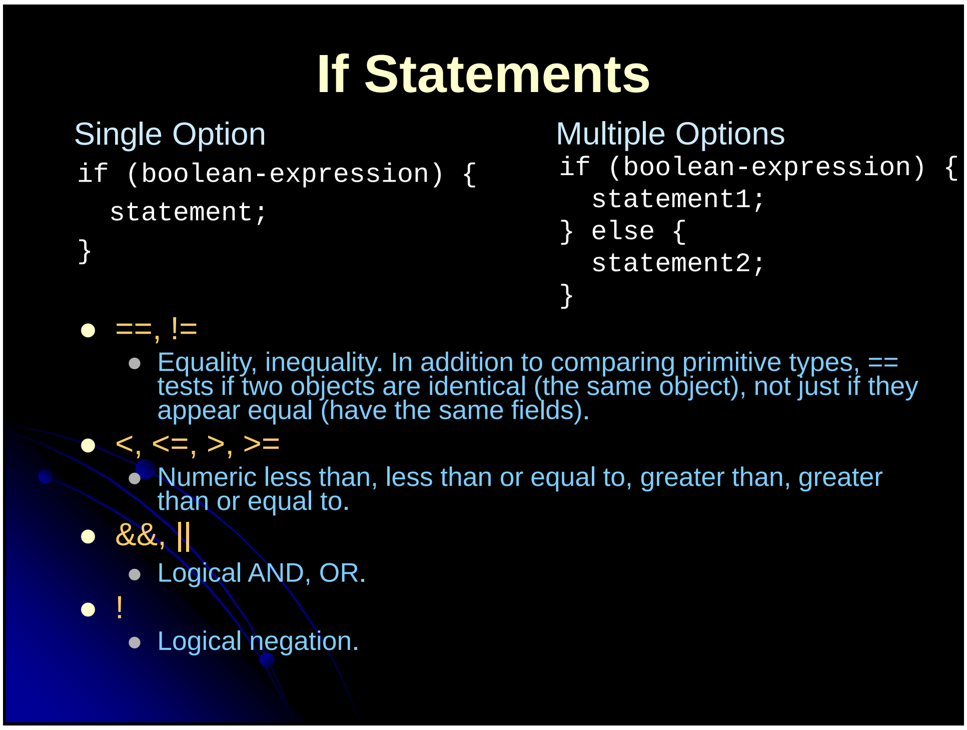
```
public class LoopTest {
  public static void main(String[] args) {
    listNums1(5);
    listNums2(6);
    listNums3(7);
  }
  public static void listNums1(int max) {…}
  public static void listNums2(int max) {…}
  public static void listNums3(int max) {…}
}
```



Loop Example

- File ShowArgs.java:

```
public class ShowArgs {
  public static void main(String[] args) {
    for(int i=0; i<args.length; i++) {
      System.out.println("Arg " + i +
                         " is " +
                         args[i]);
    }
  }
}
```



If Statements

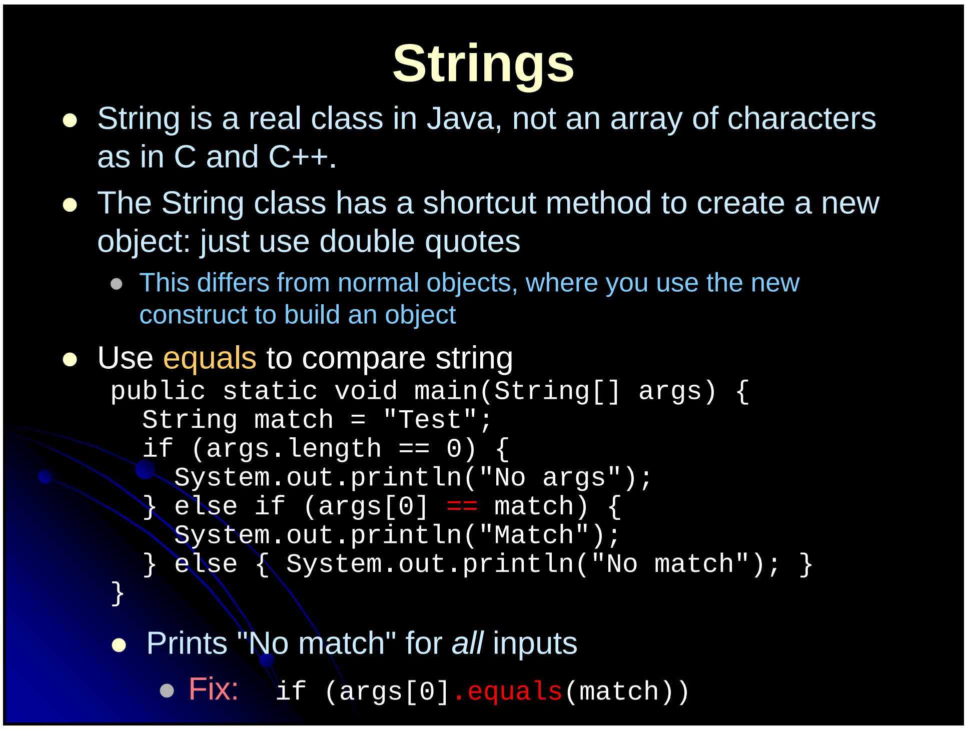
Single Option

```
if (boolean-expression) {
  statement;
}
```

Multiple Options

```
if (boolean-expression) {
  statement1;
} else {
  statement2;
}
```

- ==, !=
  - Equality, inequality. In addition to comparing primitive types, == tests if two objects are identical (the same object), not just if they appear equal (have the same fields).
- <, <=, >, >=
  - Numeric less than, less than or equal to, greater than, greater than or equal to.
- &&, ||
  - Logical AND, OR.
- !
  - Logical negation.

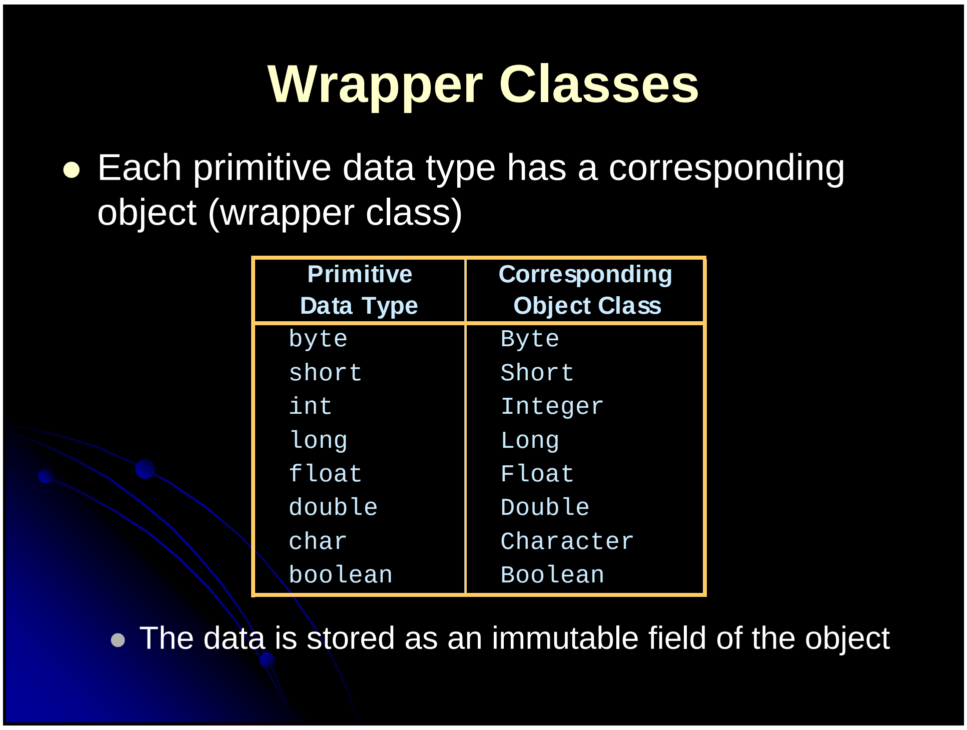


Strings

- String is a real class in Java, not an array of characters as in C and C++.
- The String class has a shortcut method to create a new object: just use double quotes
  - This differs from normal objects, where you use the new construct to build an object
- Use equals to compare string

```
public static void main(String[] args) {
  String match = "Test";
  if (args.length == 0) {
    System.out.println("No args");
  } else if (args[0] == match) {
    System.out.println("Match");
  } else { System.out.println("No match"); }
}
```

- Prints "No match" for *all* inputs
  - Fix: `if (args[0].equals(match))`

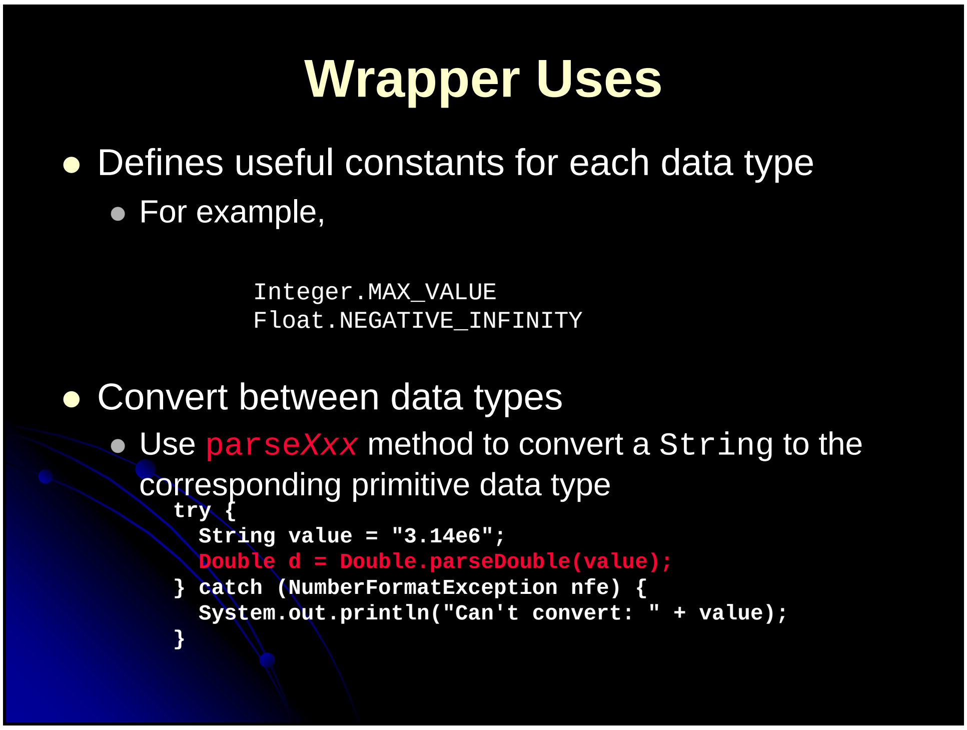


Wrapper Classes

- Each primitive data type has a corresponding object (wrapper class)

Primitive Data Type	Corresponding Object Class
`byte`	`Byte`
`short`	`Short`
`int`	`Integer`
`long`	`Long`
`float`	`Float`
`double`	`Double`
`char`	`Character`
`boolean`	`Boolean`

  - The data is stored as an immutable field of the object



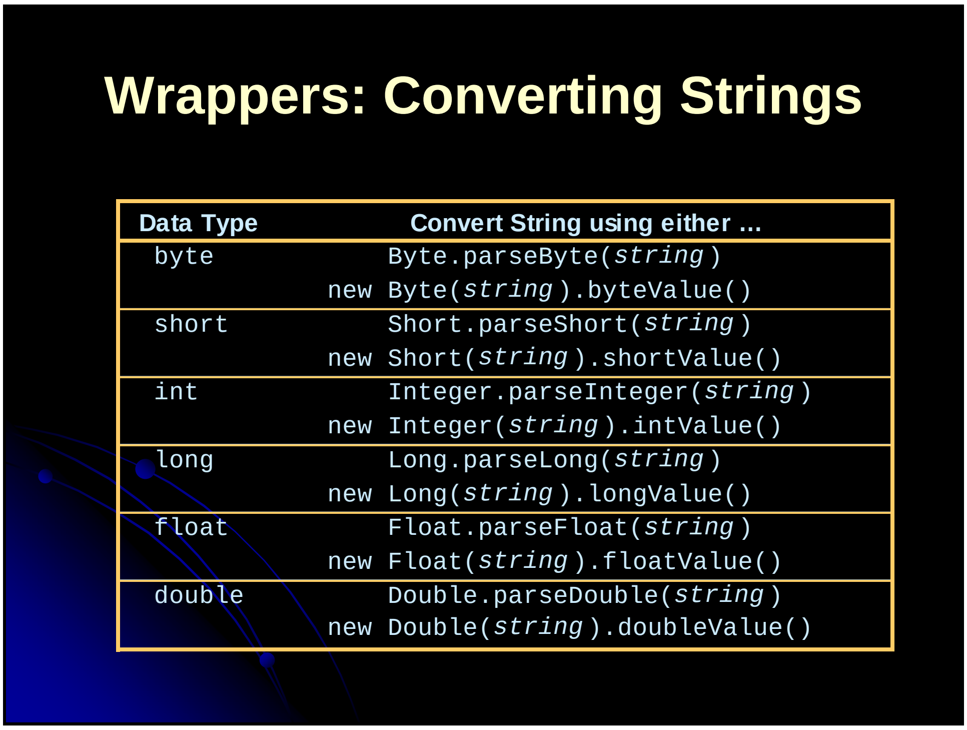
Wrapper Uses

- Defines useful constants for each data type
  - For example,

```
Integer.MAX_VALUE
Float.NEGATIVE_INFINITY
```

- Convert between data types
  - Use `parse`*`Xxx`* method to convert a `String` to the corresponding primitive data type

```
try {
  String value = "3.14e6";
  Double d = Double.parseDouble(value);
} catch (NumberFormatException nfe) {
  System.out.println("Can't convert: " + value);
}
```



Wrappers: Converting Strings

Data Type	Convert String using either …

`byte`	`Byte.parseByte(`*`string`*`)`
	`new Byte(`*`string`*`).byteValue()`

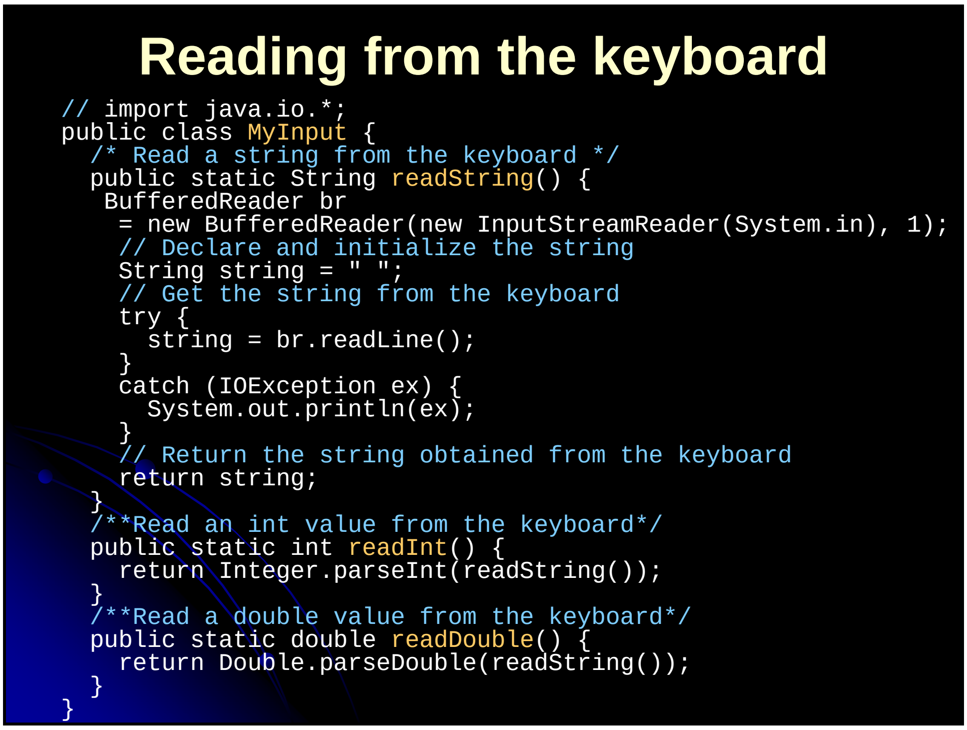
`short`	`Short.parseShort(`*`string`*`)`
	`new Short(`*`string`*`).shortValue()`

`int`	`Integer.parseInteger(`*`string`*`)`
	`new Integer(`*`string`*`).intValue()`

`long`	`Long.parseLong(`*`string`*`)`
	`new Long(`*`string`*`).longValue()`

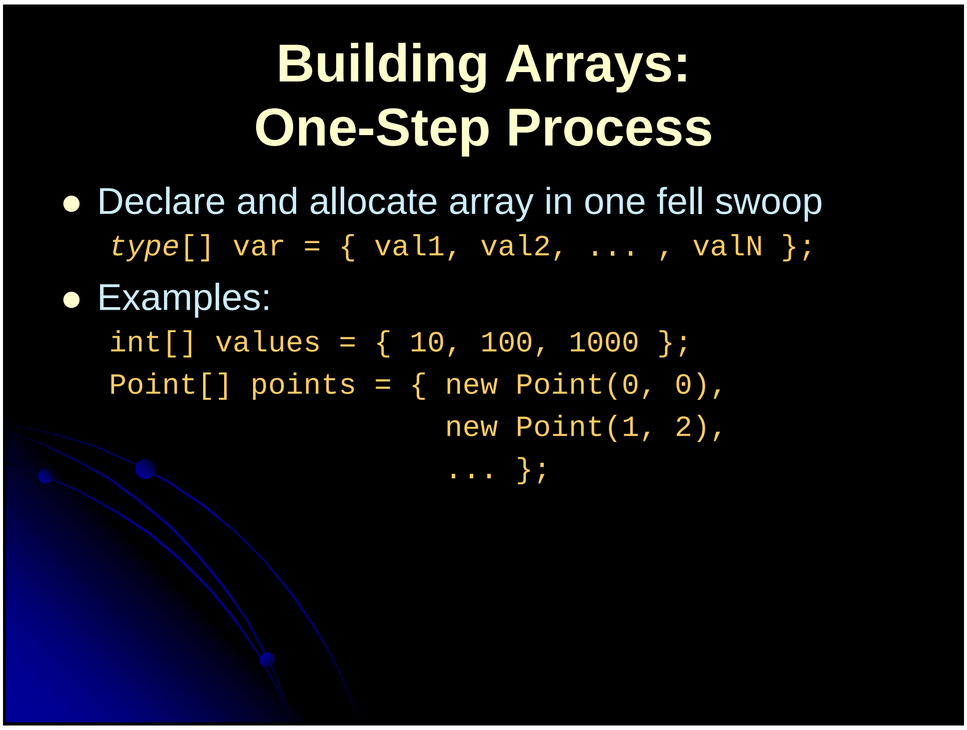
`float`	`Float.parseFloat(`*`string`*`)`
	`new Float(`*`string`*`).floatValue()`

`double`	`Double.parseDouble(`*`string`*`)`
	`new Double(`*`string`*`).doubleValue()`



Reading from the keyboard

```
// import java.io.*;
public class MyInput {
  /* Read a string from the keyboard */
  public static String readString() {
   BufferedReader br
    = new BufferedReader(new InputStreamReader(System.in), 1);
    // Declare and initialize the string
    String string = " ";
    // Get the string from the keyboard
    try {
      string = br.readLine();
    }
    catch (IOException ex) {
      System.out.println(ex);
    }
    // Return the string obtained from the keyboard
    return string;
  }
  /**Read an int value from the keyboard*/
  public static int readInt() {
    return Integer.parseInt(readString());
  }
  /**Read a double value from the keyboard*/
  public static double readDouble() {
    return Double.parseDouble(readString());
  }
}
```



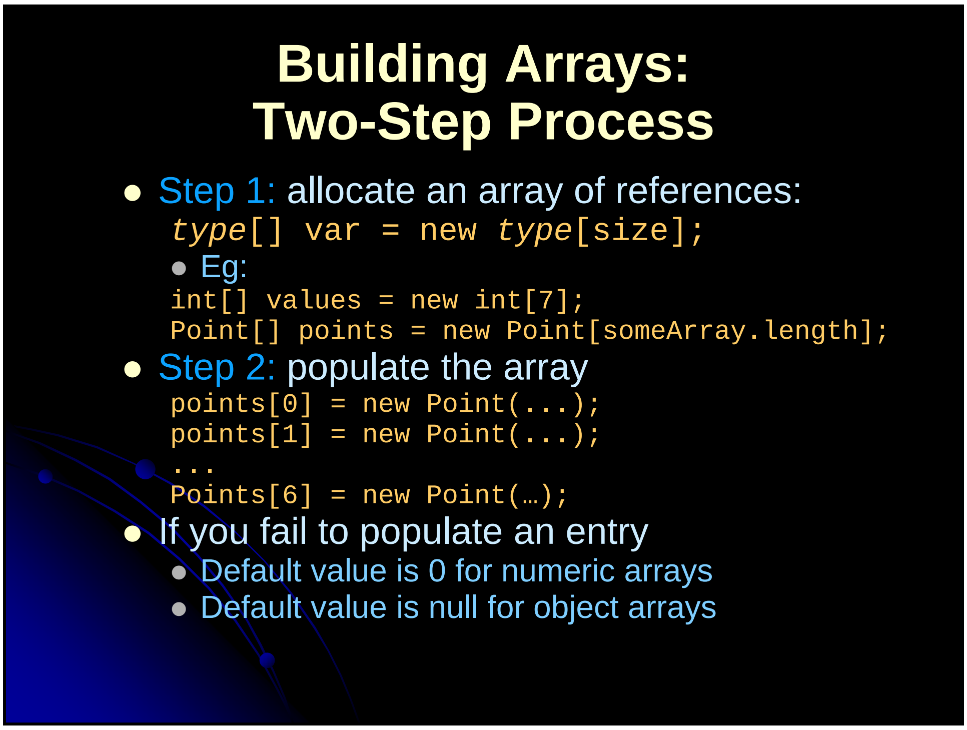
Building Arrays: One-Step Process

- Declare and allocate array in one fell swoop

```
type[] var = { val1, val2, ... , valN };
```

- Examples:

```
int[] values = { 10, 100, 1000 };
Point[] points = { new Point(0, 0),
                   new Point(1, 2),
                   ... };
```



Building Arrays: Two-Step Process

- Step 1: allocate an array of references:

```
type[] var = new type[size];
```

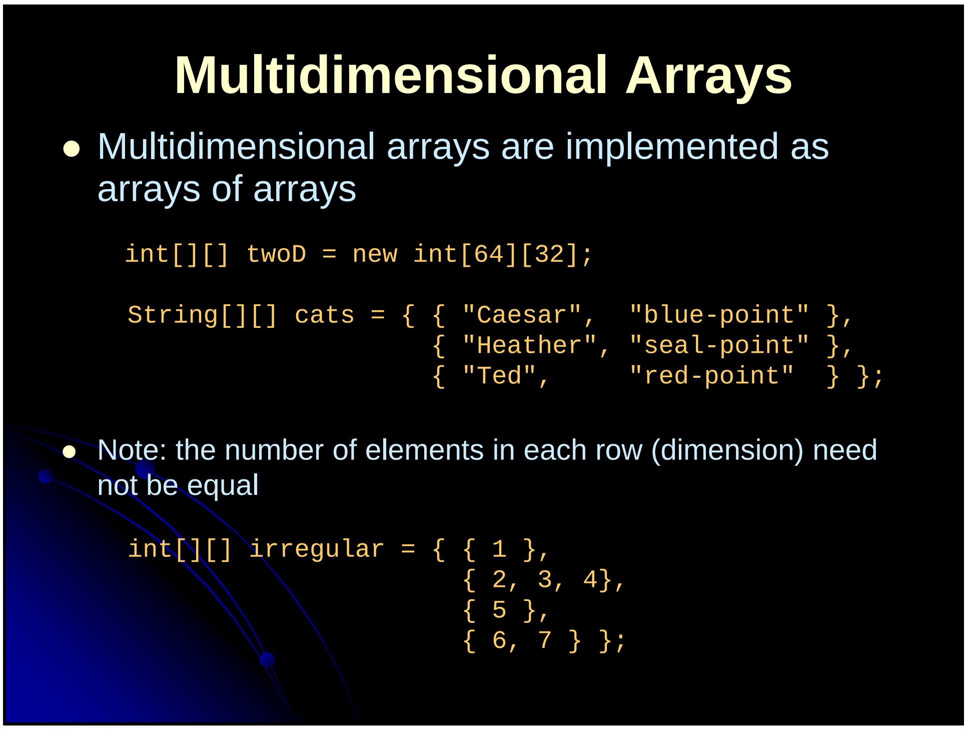
  - Eg:

```
int[] values = new int[7];
Point[] points = new Point[someArray.length];
```

- Step 2: populate the array

```
points[0] = new Point(...);
points[1] = new Point(...);
...
Points[6] = new Point(…);
```

- If you fail to populate an entry
  - Default value is 0 for numeric arrays
  - Default value is null for object arrays



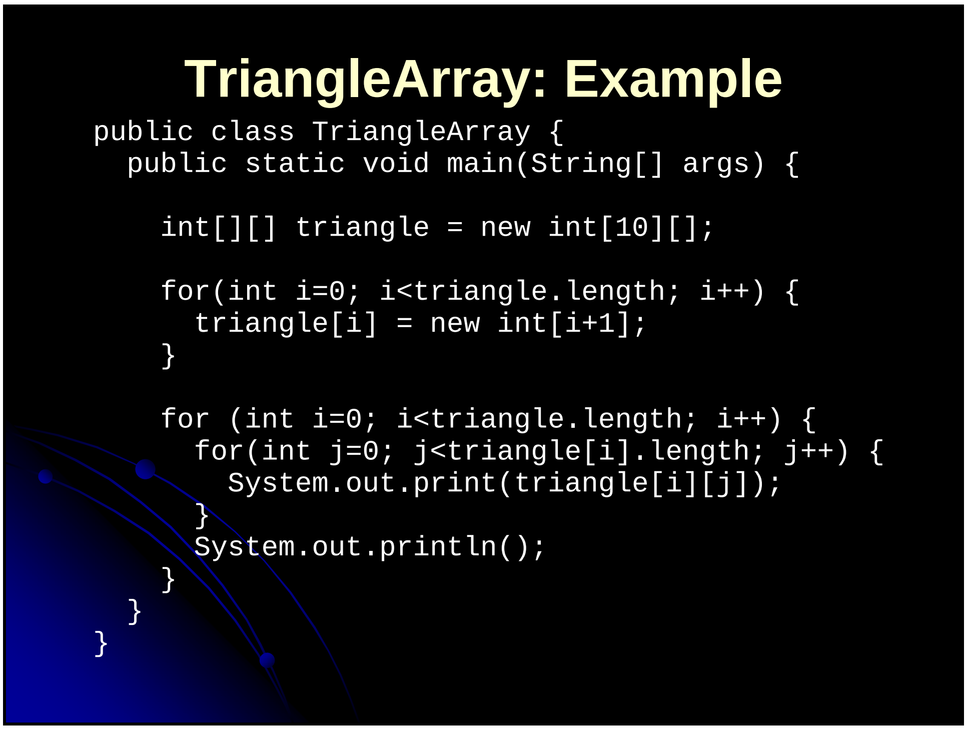
Multidimensional Arrays

- Multidimensional arrays are implemented as arrays of arrays

```
int[][] twoD = new int[64][32];

String[][] cats = { { "Caesar",  "blue-point" },
                    { "Heather", "seal-point" },
                    { "Ted",     "red-point"  } };
```

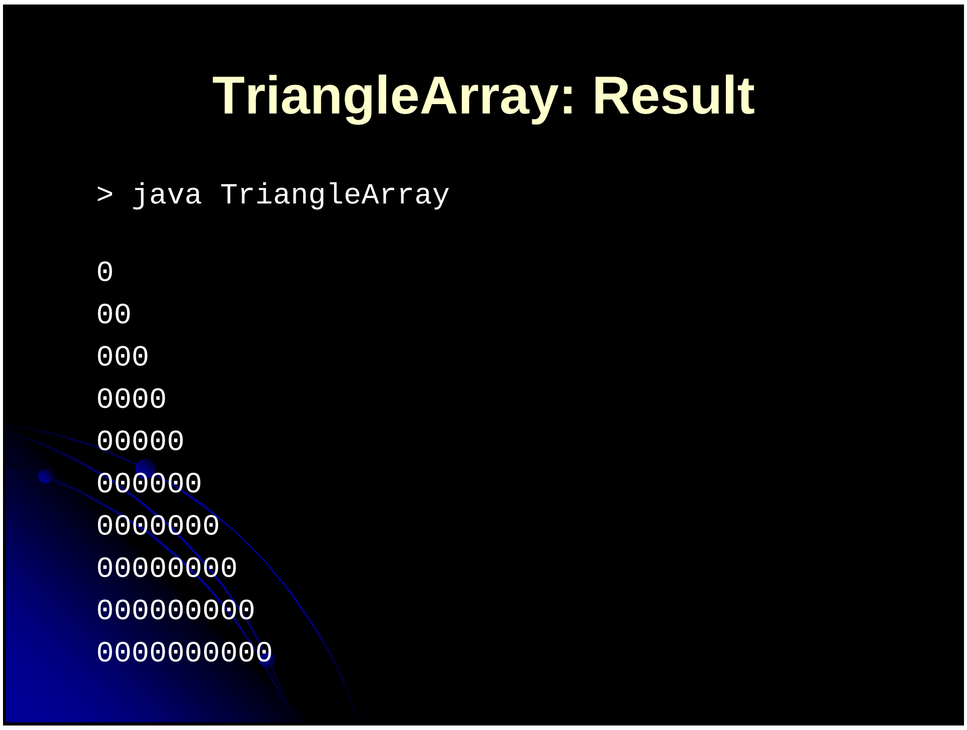
- Note: the number of elements in each row (dimension) need not be equal

```
int[][] irregular = { { 1 },
                      { 2, 3, 4},
                      { 5 },
                      { 6, 7 } };
```



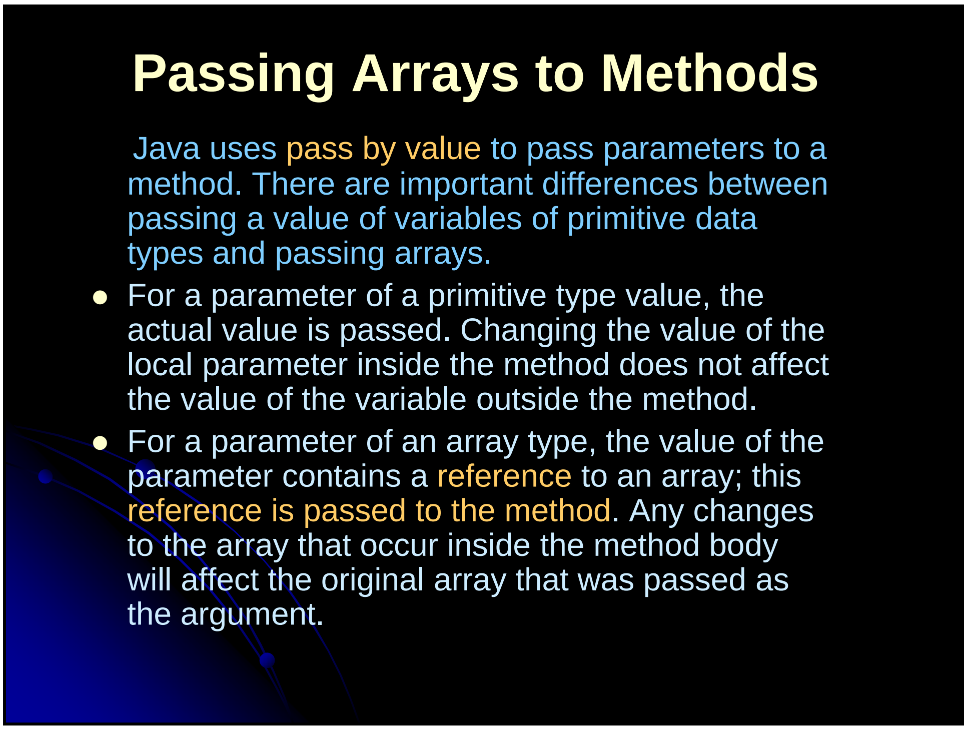
TriangleArray: Example

```
public class TriangleArray {
  public static void main(String[] args) {

    int[][] triangle = new int[10][];

    for(int i=0; i<triangle.length; i++) {
      triangle[i] = new int[i+1];
    }

    for (int i=0; i<triangle.length; i++) {
      for(int j=0; j<triangle[i].length; j++) {
        System.out.print(triangle[i][j]);
      }
      System.out.println();
    }
  }
}
```



TriangleArray: Result

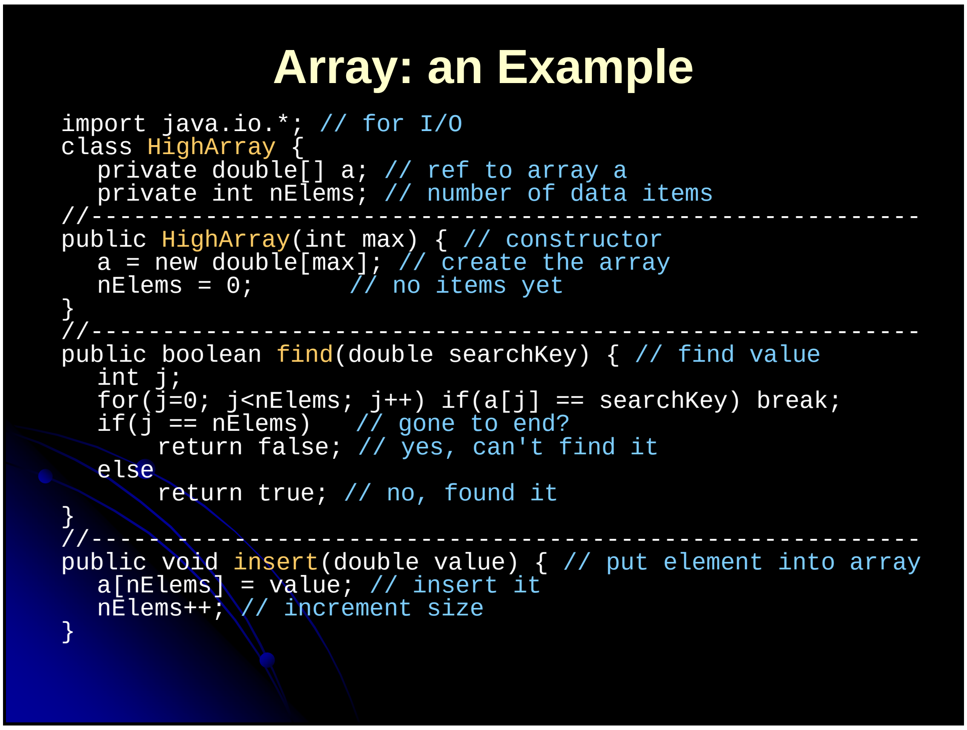
```
> java TriangleArray

0
00
000
0000
00000
000000
0000000
00000000
000000000
0000000000
```



Passing Arrays to Methods

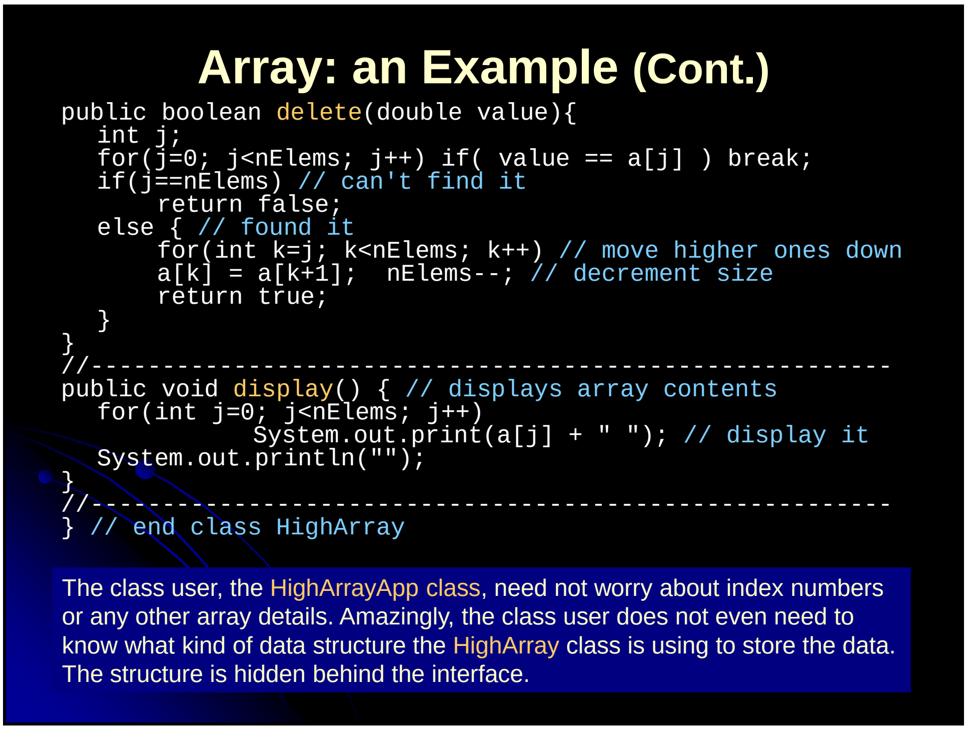
Java uses pass by value to pass parameters to a method. There are important differences between passing a value of variables of primitive data types and passing arrays.

- For a parameter of a primitive type value, the actual value is passed. Changing the value of the local parameter inside the method does not affect the value of the variable outside the method.
- For a parameter of an array type, the value of the parameter contains a reference to an array; this reference is passed to the method. Any changes to the array that occur inside the method body will affect the original array that was passed as the argument.



Array: an Example

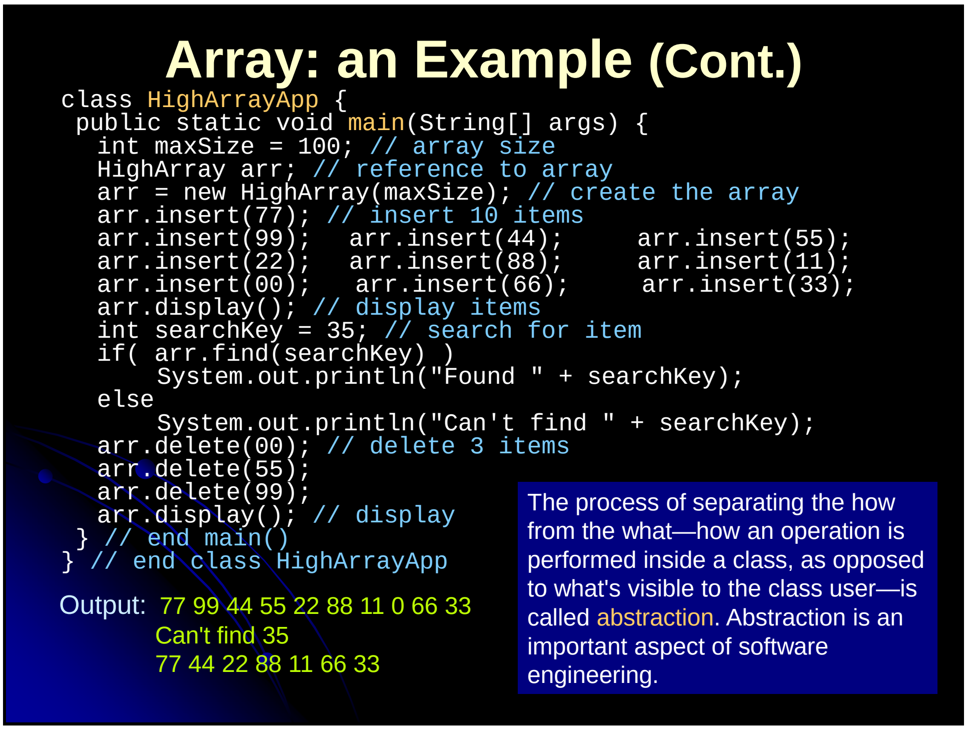
```
import java.io.*; // for I/O
class HighArray {
   private double[] a; // ref to array a
   private int nElems; // number of data items
//------------------------------------------------------------
public HighArray(int max) { // constructor
   a = new double[max]; // create the array
   nElems = 0;       // no items yet
}
//------------------------------------------------------------
public boolean find(double searchKey) { // find value
   int j;
   for(j=0; j<nElems; j++) if(a[j] == searchKey) break;
   if(j == nElems)   // gone to end?
        return false; // yes, can't find it
   else
        return true; // no, found it
}
//------------------------------------------------------------
public void insert(double value) { // put element into array
   a[nElems] = value; // insert it
   nElems++; // increment size
}
```



Array: an Example (Cont.)

```
public boolean delete(double value){
   int j;
   for(j=0; j<nElems; j++) if( value == a[j] ) break;
   if(j==nElems) // can't find it
        return false;
   else { // found it
        for(int k=j; k<nElems; k++) // move higher ones down
        a[k] = a[k+1];  nElems--; // decrement size
        return true;
   }
}
//----------------------------------------------------------
public void display() { // displays array contents
   for(int j=0; j<nElems; j++)
             System.out.print(a[j] + " "); // display it
   System.out.println("");
}
//----------------------------------------------------------
} // end class HighArray
```

The class user, the HighArrayApp class, need not worry about index numbers or any other array details. Amazingly, the class user does not even need to know what kind of data structure the HighArray class is using to store the data. The structure is hidden behind the interface.

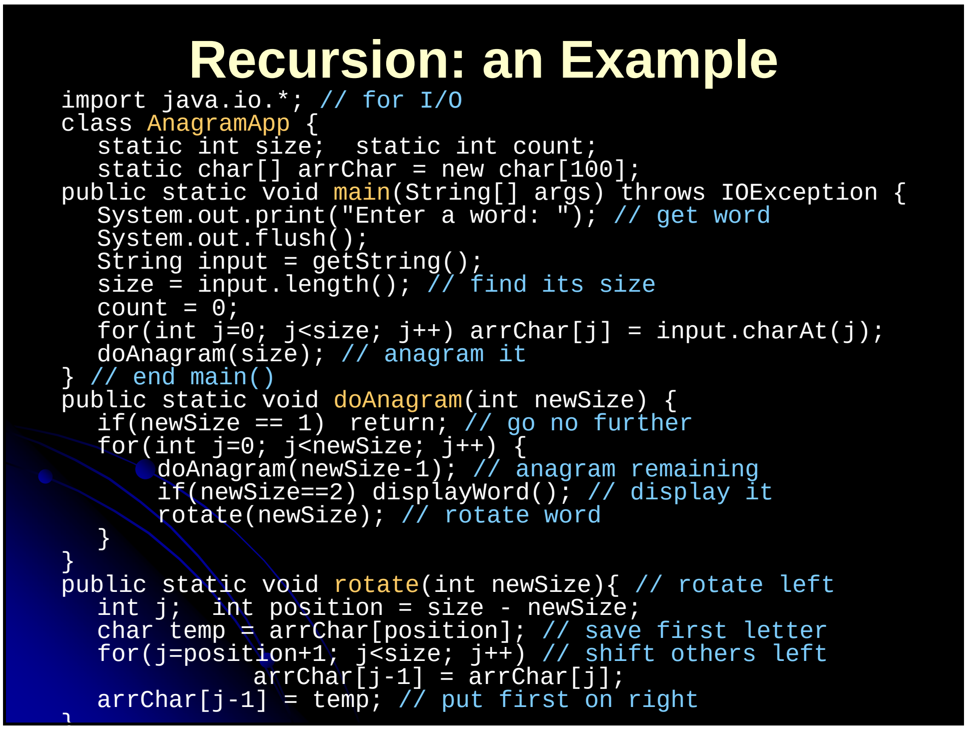


Array: an Example (Cont.)

```
class HighArrayApp {
 public static void main(String[] args) {
   int maxSize = 100; // array size
   HighArray arr; // reference to array
   arr = new HighArray(maxSize); // create the array
   arr.insert(77); // insert 10 items
   arr.insert(99);   arr.insert(44);     arr.insert(55);
   arr.insert(22);   arr.insert(88);     arr.insert(11);
   arr.insert(00);    arr.insert(66);     arr.insert(33);
   arr.display(); // display items
   int searchKey = 35; // search for item
   if( arr.find(searchKey) )
       System.out.println("Found " + searchKey);
   else
       System.out.println("Can't find " + searchKey);
   arr.delete(00); // delete 3 items
   arr.delete(55);
   arr.delete(99);
   arr.display(); // display
 } // end main()
} // end class HighArrayApp
```

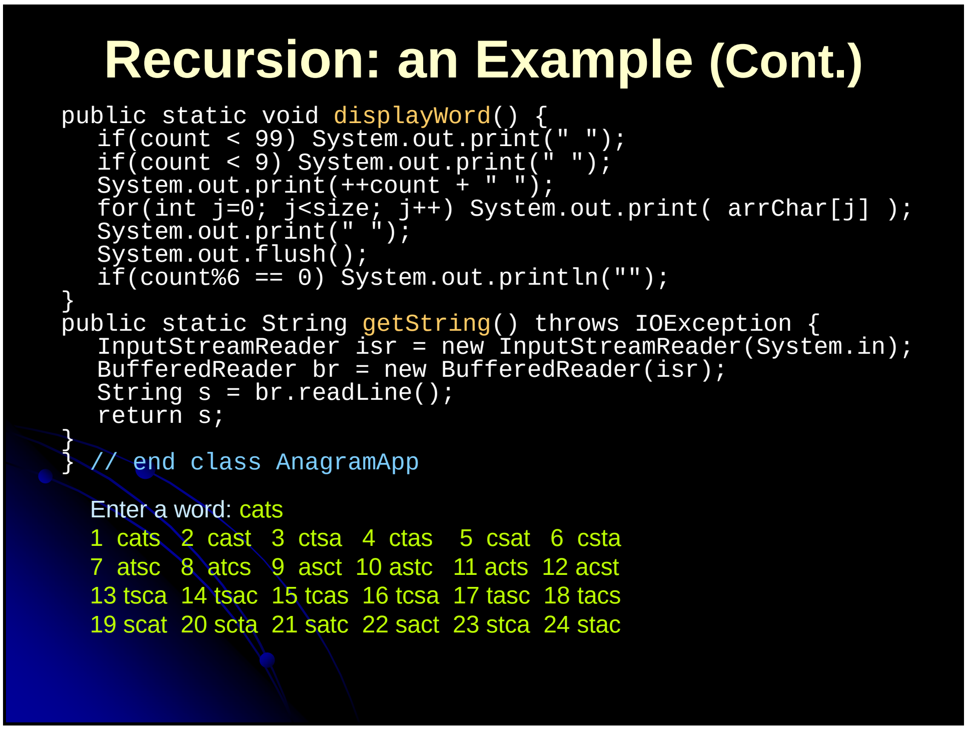
Output:
```
77 99 44 55 22 88 11 0 66 33
Can't find 35
77 44 22 88 11 66 33
```

The process of separating the how from the what—how an operation is performed inside a class, as opposed to what's visible to the class user—is called abstraction. Abstraction is an important aspect of software engineering.



Recursion: an Example

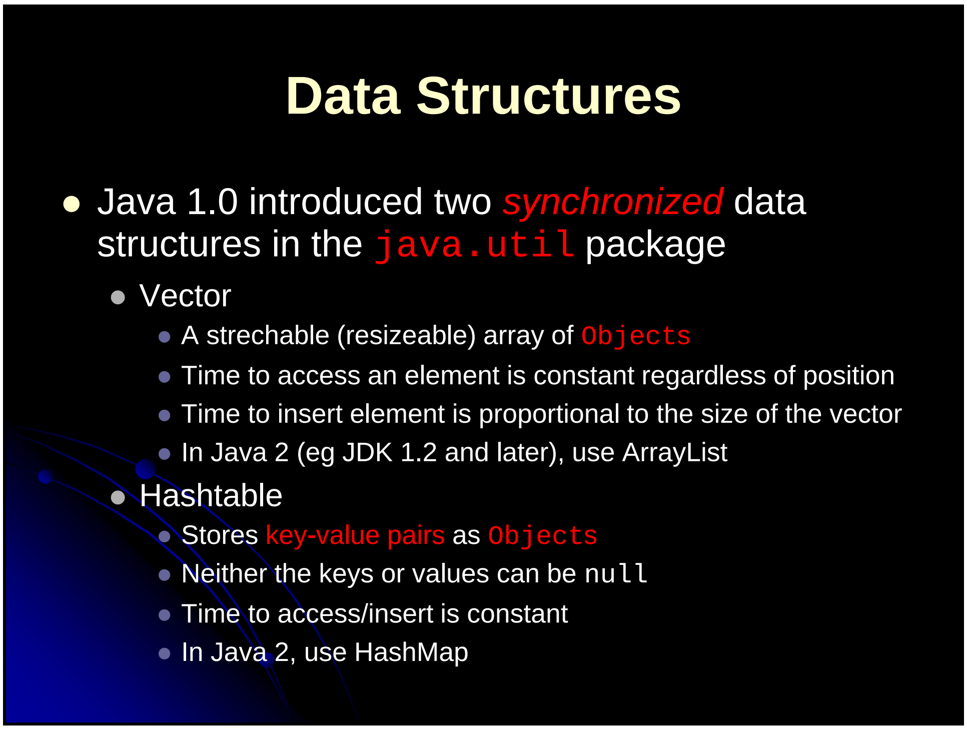
```
import java.io.*; // for I/O
class AnagramApp {
   static int size;  static int count;
   static char[] arrChar = new char[100];
public static void main(String[] args) throws IOException {
   System.out.print("Enter a word: "); // get word
   System.out.flush();
   String input = getString();
   size = input.length(); // find its size
   count = 0;
   for(int j=0; j<size; j++) arrChar[j] = input.charAt(j);
   doAnagram(size); // anagram it
} // end main()
public static void doAnagram(int newSize) {
   if(newSize == 1)  return; // go no further
   for(int j=0; j<newSize; j++) {
       doAnagram(newSize-1); // anagram remaining
       if(newSize==2) displayWord(); // display it
       rotate(newSize); // rotate word
   }
}
public static void rotate(int newSize){ // rotate left
   int j;  int position = size - newSize;
   char temp = arrChar[position]; // save first letter
   for(j=position+1; j<size; j++) // shift others left
             arrChar[j-1] = arrChar[j];
   arrChar[j-1] = temp; // put first on right
}
```



Recursion: an Example (Cont.)

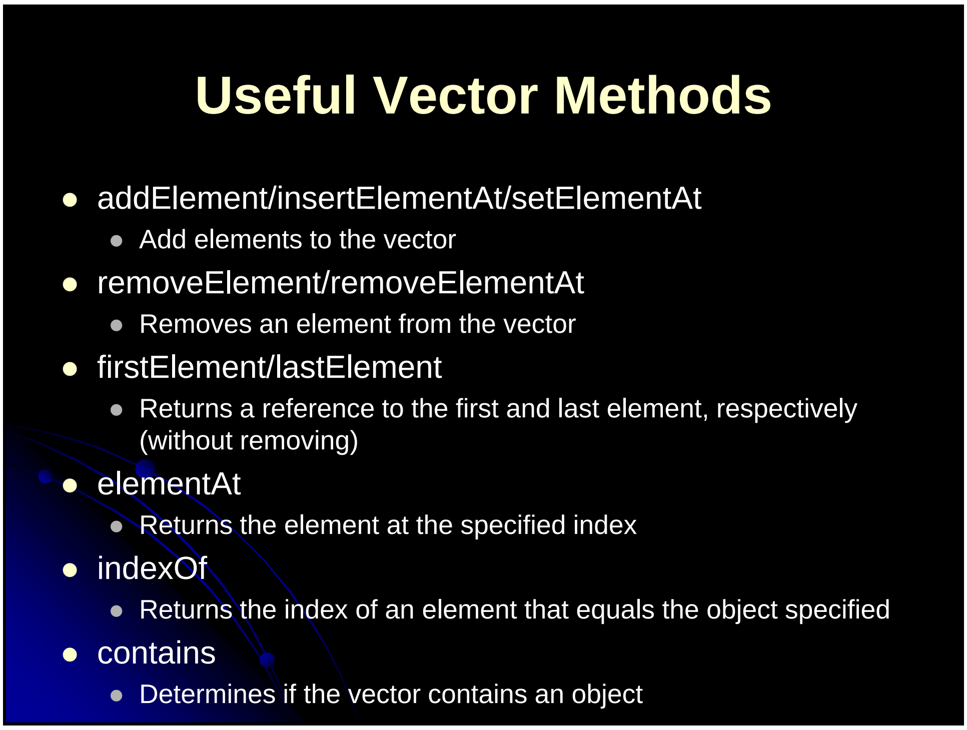
```
public static void displayWord() {
   if(count < 99) System.out.print(" ");
   if(count < 9) System.out.print(" ");
   System.out.print(++count + " ");
   for(int j=0; j<size; j++) System.out.print( arrChar[j] );
   System.out.print(" ");
   System.out.flush();
   if(count%6 == 0) System.out.println("");
}
public static String getString() throws IOException {
   InputStreamReader isr = new InputStreamReader(System.in);
   BufferedReader br = new BufferedReader(isr);
   String s = br.readLine();
   return s;
}
} // end class AnagramApp
```

```
Enter a word: cats
1  cats  2  cast  3  ctsa  4  ctas   5  csat  6  csta
7  atsc  8  atcs  9  asct  10 astc  11 acts  12 acst
13 tsca  14 tsac  15 tcas  16 tcsa  17 tasc  18 tacs
19 scat  20 scta  21 satc  22 sact  23 stca  24 stac
```



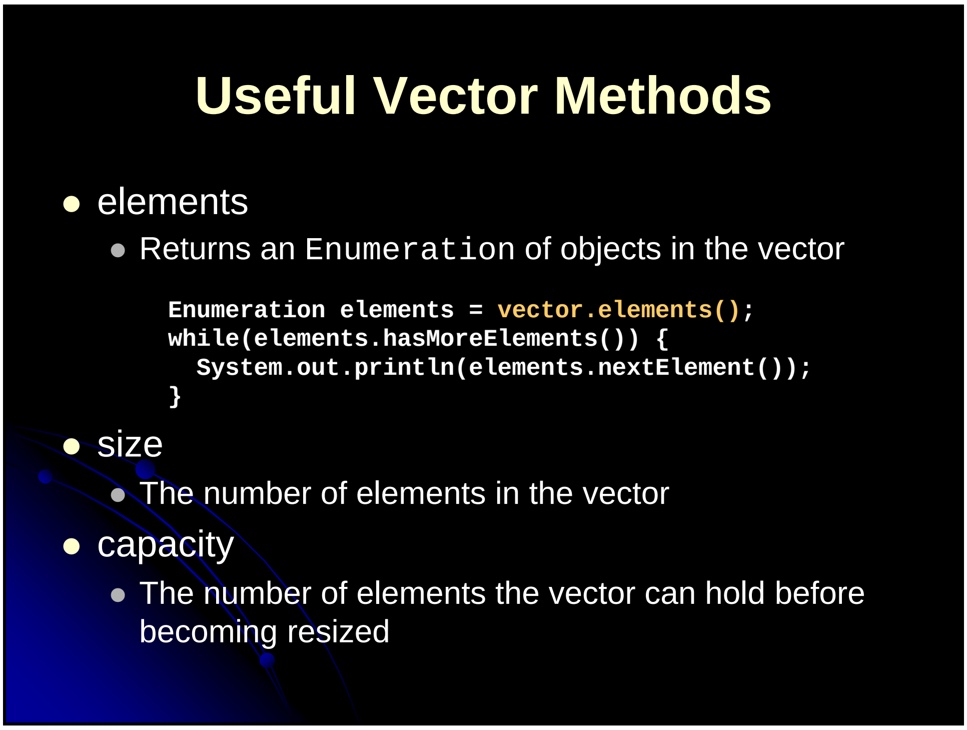
Data Structures

- Java 1.0 introduced two *synchronized* data structures in the `java.util` package
  - Vector
    - A strechable (resizeable) array of `Objects`
    - Time to access an element is constant regardless of position
    - Time to insert element is proportional to the size of the vector
    - In Java 2 (eg JDK 1.2 and later), use ArrayList
  - Hashtable
    - Stores key-value pairs as `Objects`
    - Neither the keys or values can be `null`
    - Time to access/insert is constant
    - In Java 2, use HashMap



Useful Vector Methods

- addElement/insertElementAt/setElementAt
  - Add elements to the vector
- removeElement/removeElementAt
  - Removes an element from the vector
- firstElement/lastElement
  - Returns a reference to the first and last element, respectively (without removing)
- elementAt
  - Returns the element at the specified index
- indexOf
  - Returns the index of an element that equals the object specified
- contains
  - Determines if the vector contains an object

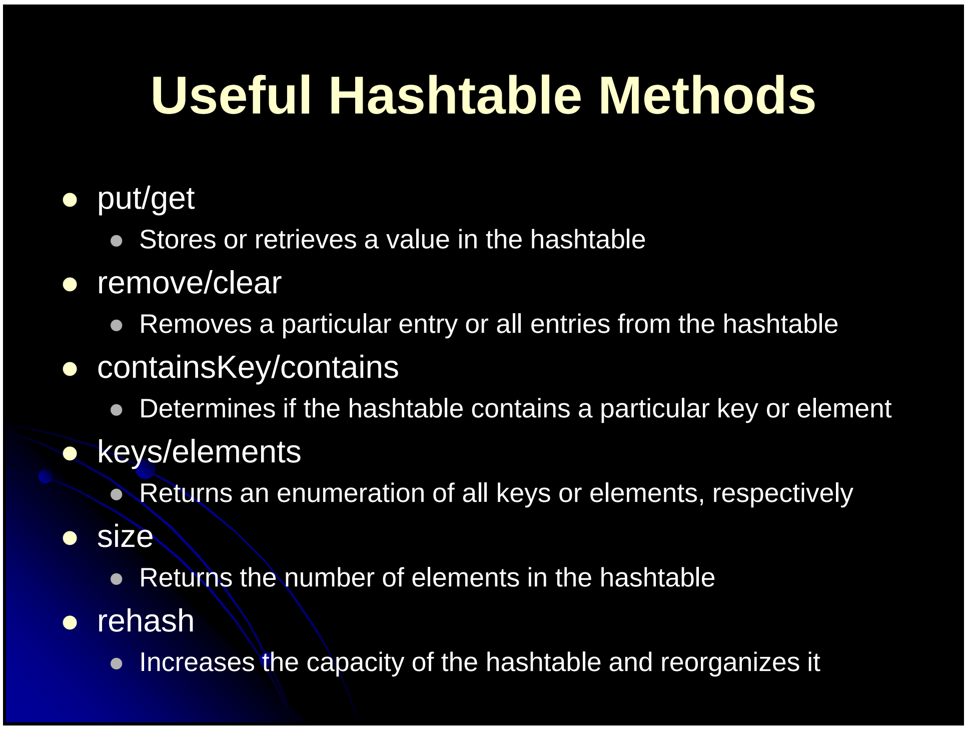


Useful Vector Methods

- elements
  - Returns an `Enumeration` of objects in the vector

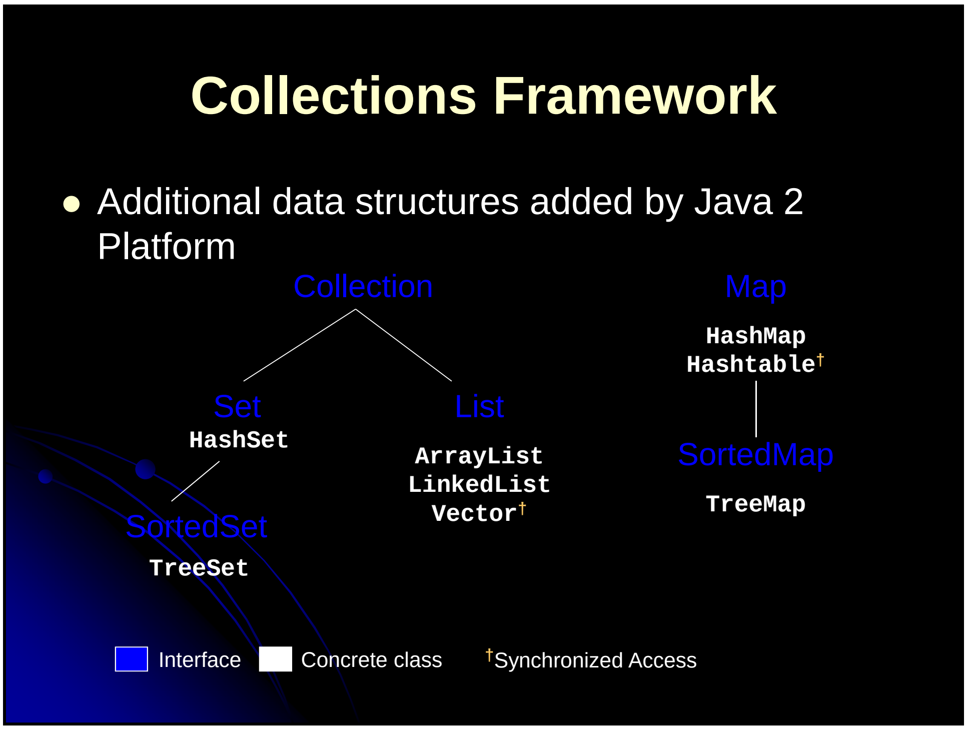
```
Enumeration elements = vector.elements();
while(elements.hasMoreElements()) {
  System.out.println(elements.nextElement());
}
```

- size
  - The number of elements in the vector
- capacity
  - The number of elements the vector can hold before becoming resized



Useful Hashtable Methods

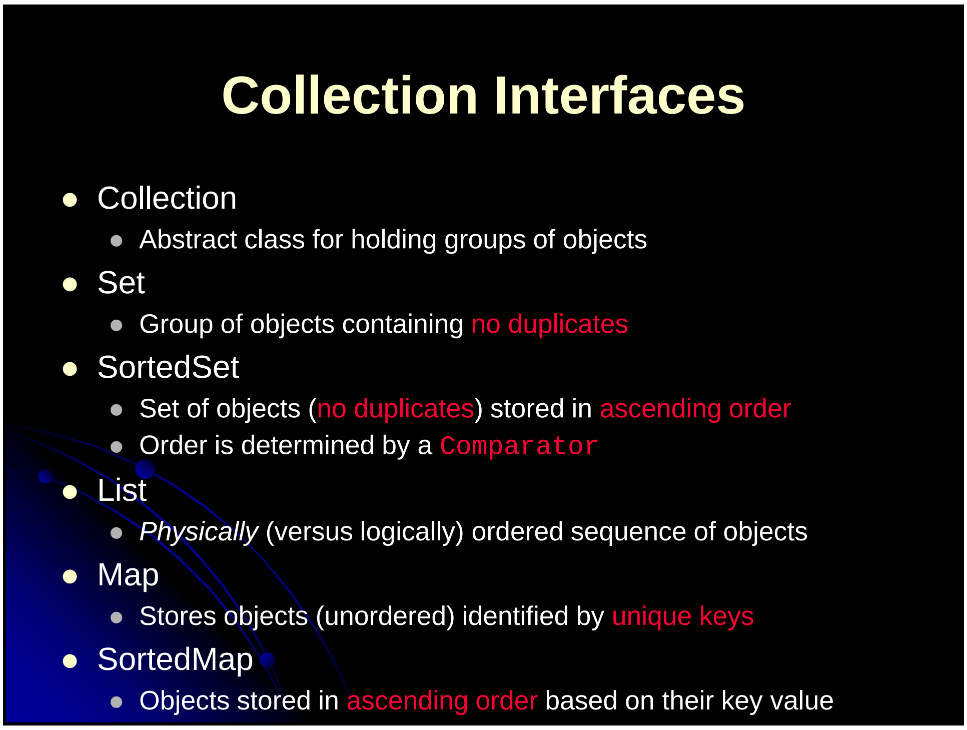
- put/get
  - Stores or retrieves a value in the hashtable
- remove/clear
  - Removes a particular entry or all entries from the hashtable
- containsKey/contains
  - Determines if the hashtable contains a particular key or element
- keys/elements
  - Returns an enumeration of all keys or elements, respectively
- size
  - Returns the number of elements in the hashtable
- rehash
  - Increases the capacity of the hashtable and reorganizes it



Collections Framework

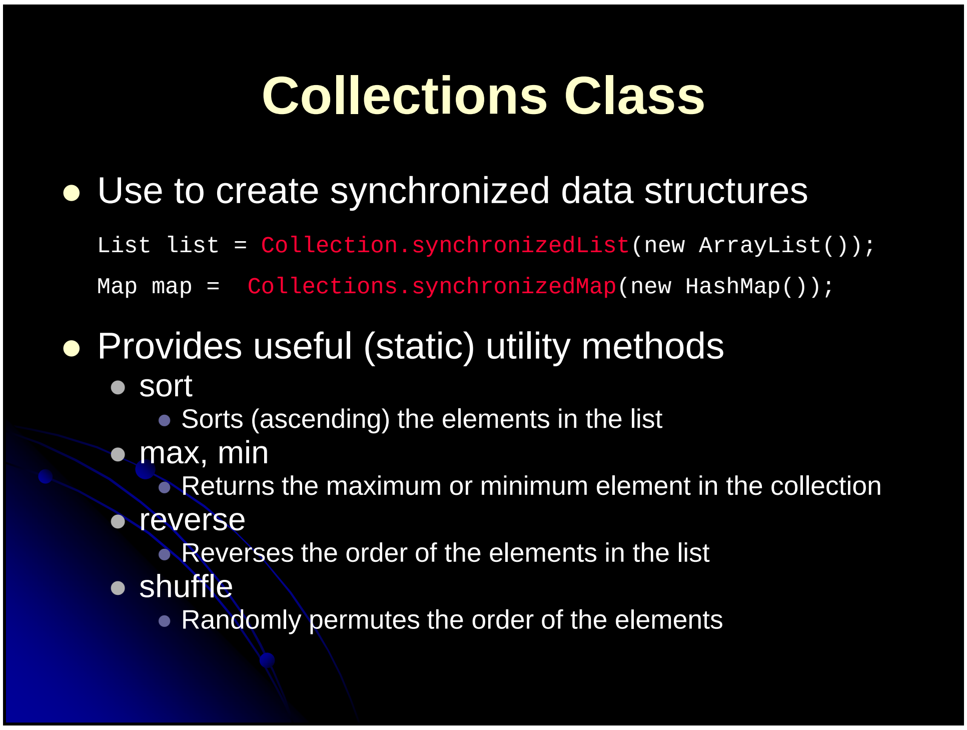
- Additional data structures added by Java 2 Platform

🟦 Interface □ Concrete class †Synchronized Access



Collection Interfaces

- Collection
  - Abstract class for holding groups of objects
- Set
  - Group of objects containing no duplicates
- SortedSet
  - Set of objects (no duplicates) stored in ascending order
  - Order is determined by a `Comparator`
- List
  - *Physically* (versus logically) ordered sequence of objects
- Map
  - Stores objects (unordered) identified by unique keys
- SortedMap
  - Objects stored in ascending order based on their key value

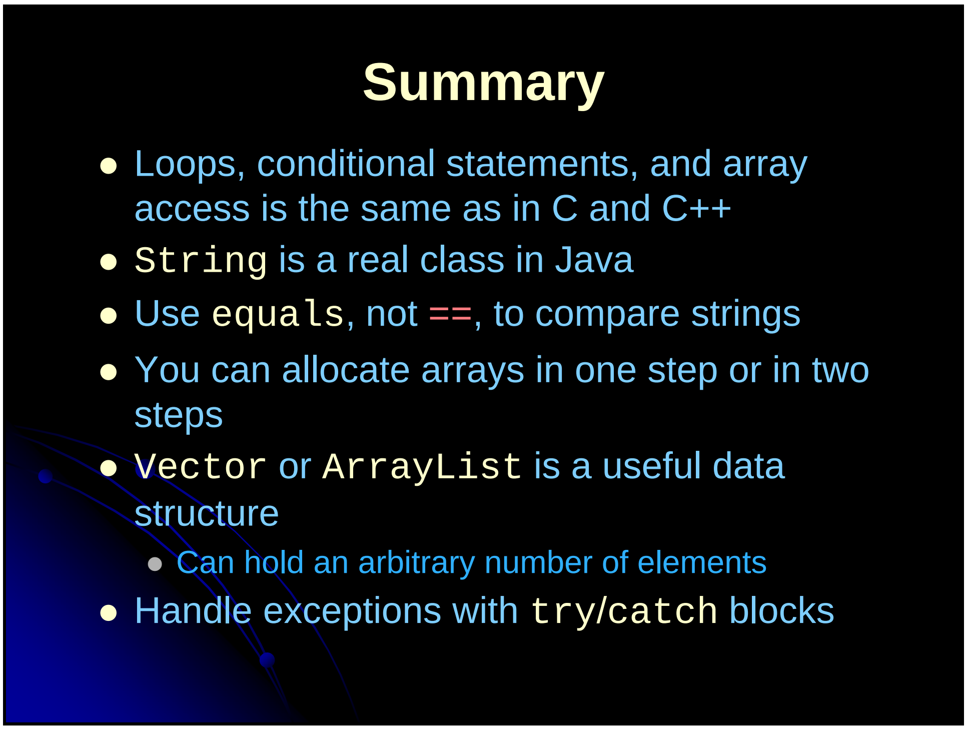


Collections Class

- Use to create synchronized data structures

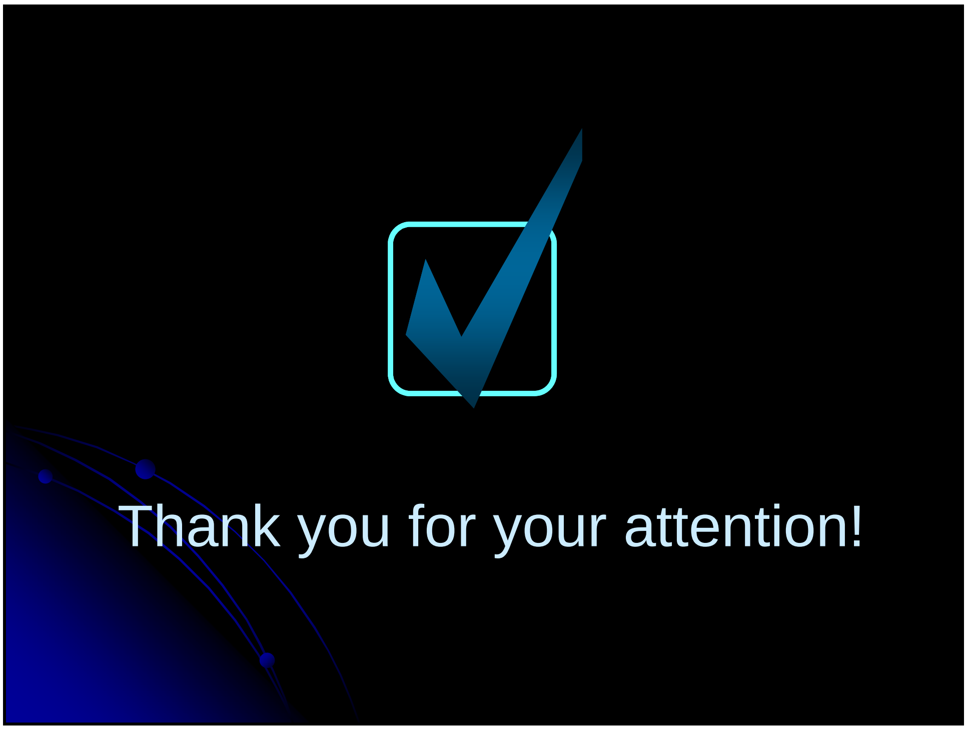
```
List list = Collection.synchronizedList(new ArrayList());
Map map =  Collections.synchronizedMap(new HashMap());
```

- Provides useful (static) utility methods
  - sort
    - Sorts (ascending) the elements in the list
  - max, min
    - Returns the maximum or minimum element in the collection
  - reverse
    - Reverses the order of the elements in the list
  - shuffle
    - Randomly permutes the order of the elements

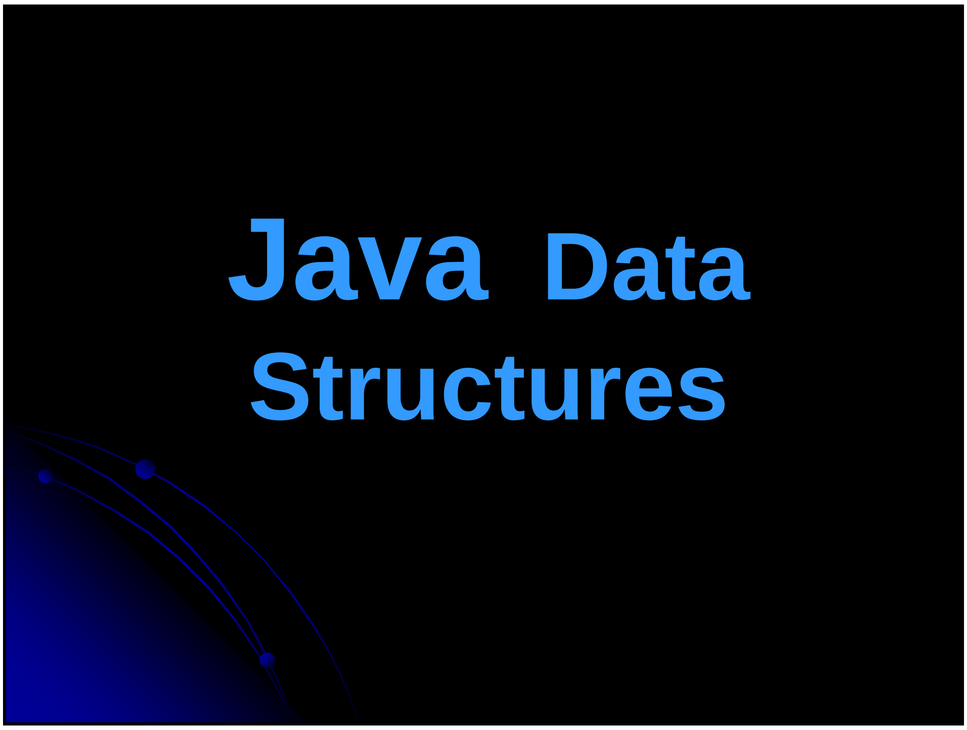


Summary

- Loops, conditional statements, and array access is the same as in C and C++
- `String` is a real class in Java
- Use `equals`, not `==`, to compare strings
- You can allocate arrays in one step or in two steps
- `Vector` or `ArrayList` is a useful data structure
  - Can hold an arbitrary number of elements
- Handle exceptions with `try/catch` blocks



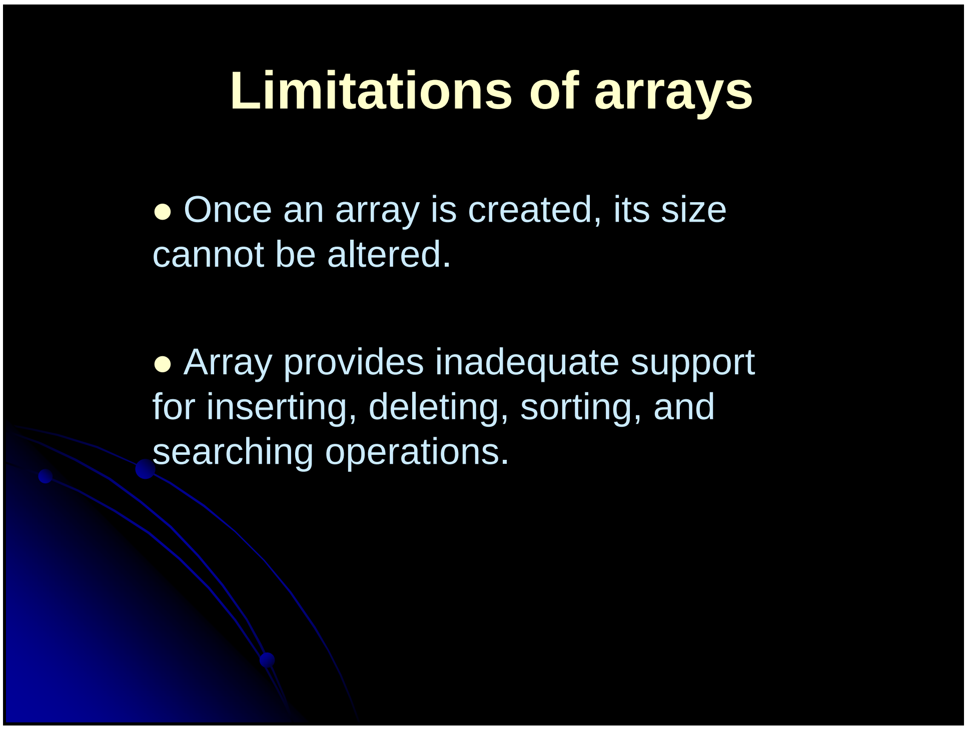
Thank you for your attention!



Java Data Structures

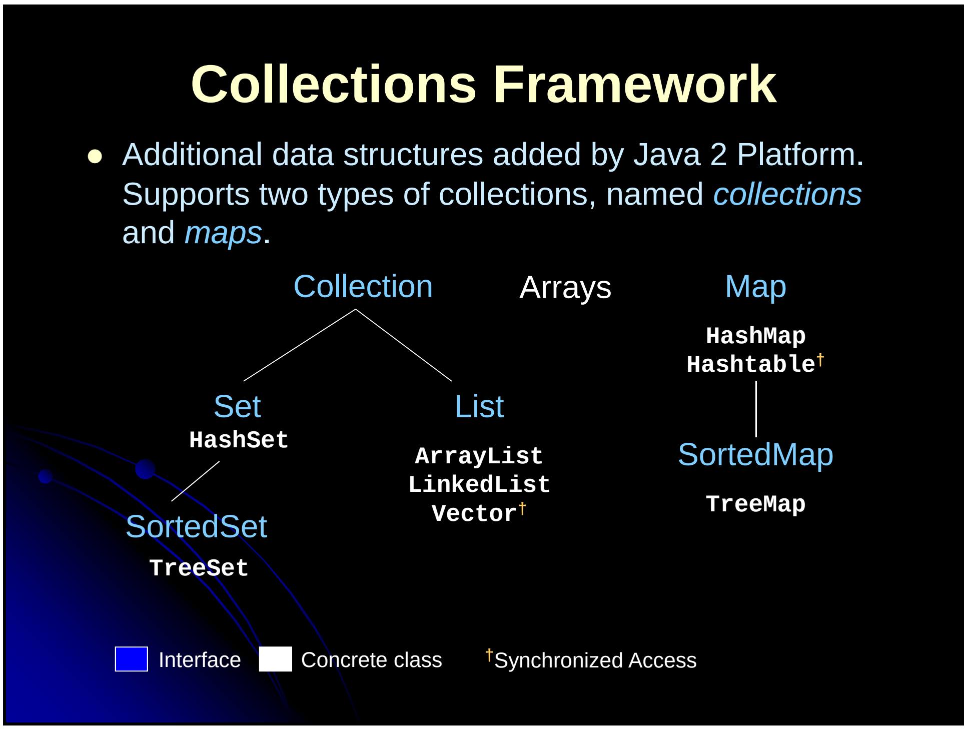
Agenda

- The limitations of arrays
- Java Collection Framework hierarchy
- Use the Iterator interface to traverse a collection
- Set interface, HashSet, and TreeSet
- List interface, ArrayList, and LinkedList
- Vector and Stack
- Map, HashMap, and TreeMap
- Collections and Arrays classes



Limitations of arrays

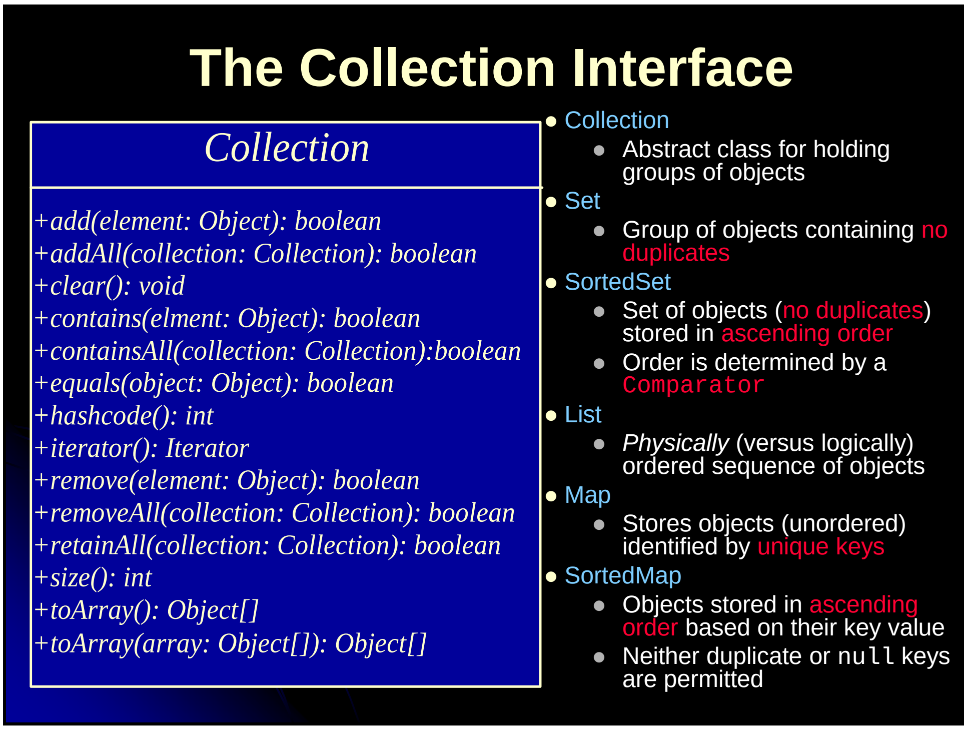
- Once an array is created, its size cannot be altered.
- Array provides inadequate support for inserting, deleting, sorting, and searching operations.



Collections Framework

- Additional data structures added by Java 2 Platform. Supports two types of collections, named *collections* and *maps*.

🟦 Interface ⬜ Concrete class †Synchronized Access

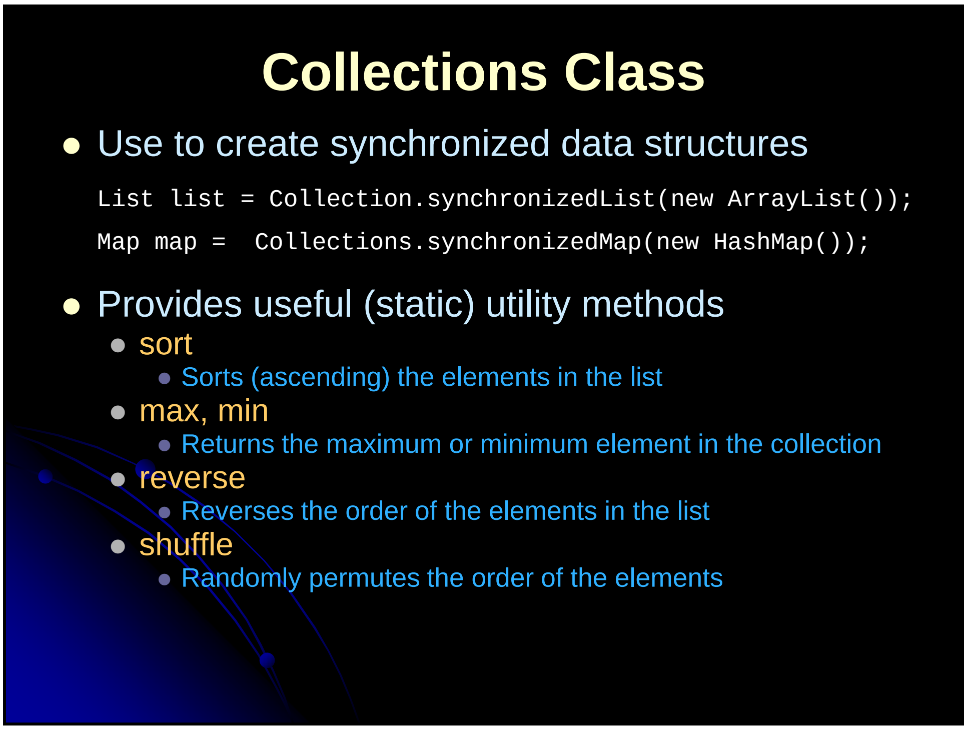


The Collection Interface

Collection

+add(element: Object): boolean
+addAll(collection: Collection): boolean
+clear(): void
+contains(elment: Object): boolean
+containsAll(collection: Collection):boolean
+equals(object: Object): boolean
+hashcode(): int
+iterator(): Iterator
+remove(element: Object): boolean
+removeAll(collection: Collection): boolean
+retainAll(collection: Collection): boolean
+size(): int
+toArray(): Object[]
+toArray(array: Object[]): Object[]

- Collection
  - Abstract class for holding groups of objects
- Set
  - Group of objects containing no duplicates
- SortedSet
  - Set of objects (no duplicates) stored in ascending order
  - Order is determined by a `Comparator`
- List
  - *Physically* (versus logically) ordered sequence of objects
- Map
  - Stores objects (unordered) identified by unique keys
- SortedMap
  - Objects stored in ascending order based on their key value
  - Neither duplicate or `null` keys are permitted

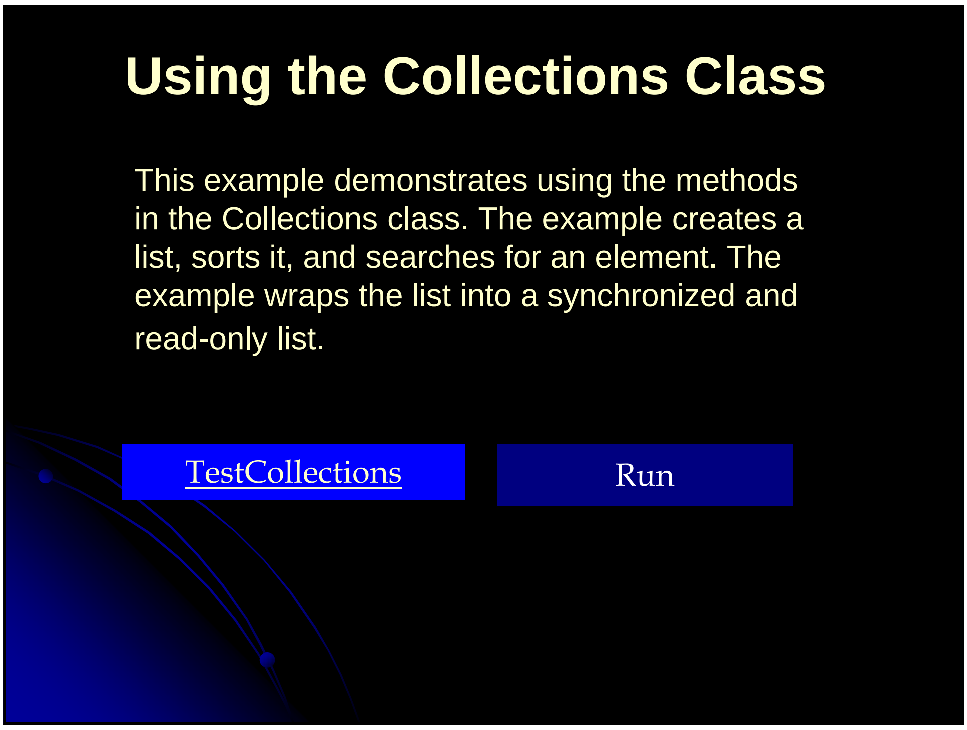


Collections Class

- Use to create synchronized data structures

```
List list = Collection.synchronizedList(new ArrayList());
Map map =  Collections.synchronizedMap(new HashMap());
```

- Provides useful (static) utility methods
  - sort
    - Sorts (ascending) the elements in the list
  - max, min
    - Returns the maximum or minimum element in the collection
  - reverse
    - Reverses the order of the elements in the list
  - shuffle
    - Randomly permutes the order of the elements

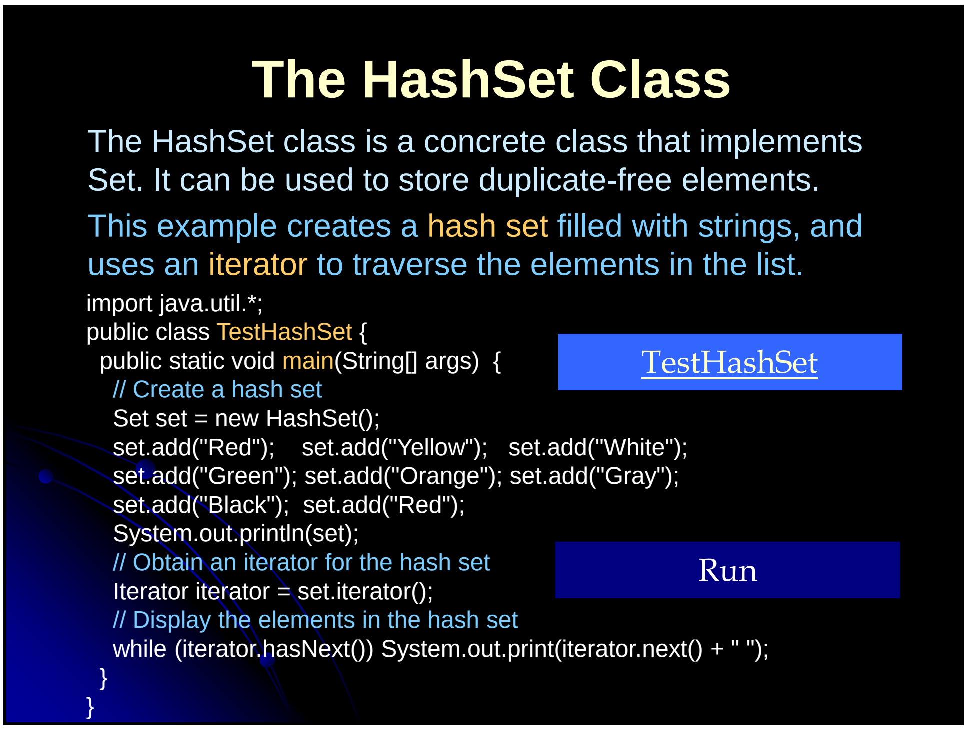


Using the Collections Class

This example demonstrates using the methods in the Collections class. The example creates a list, sorts it, and searches for an element. The example wraps the list into a synchronized and read-only list.

TestCollections

Run



The HashSet Class

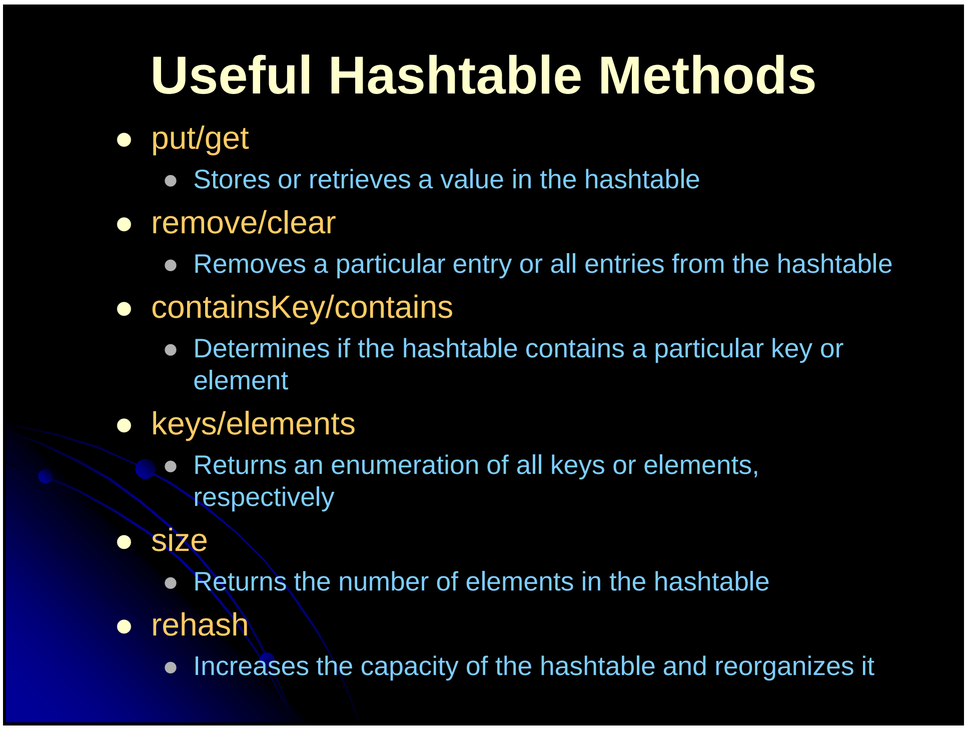
The HashSet class is a concrete class that implements Set. It can be used to store duplicate-free elements.

This example creates a hash set filled with strings, and uses an iterator to traverse the elements in the list.

```
import java.util.*;
public class TestHashSet {
  public static void main(String[] args)  {
    // Create a hash set
    Set set = new HashSet();
    set.add("Red");    set.add("Yellow");   set.add("White");
    set.add("Green"); set.add("Orange"); set.add("Gray");
    set.add("Black");  set.add("Red");
    System.out.println(set);
    // Obtain an iterator for the hash set
    Iterator iterator = set.iterator();
    // Display the elements in the hash set
    while (iterator.hasNext()) System.out.print(iterator.next() + " ");
  }
}
```

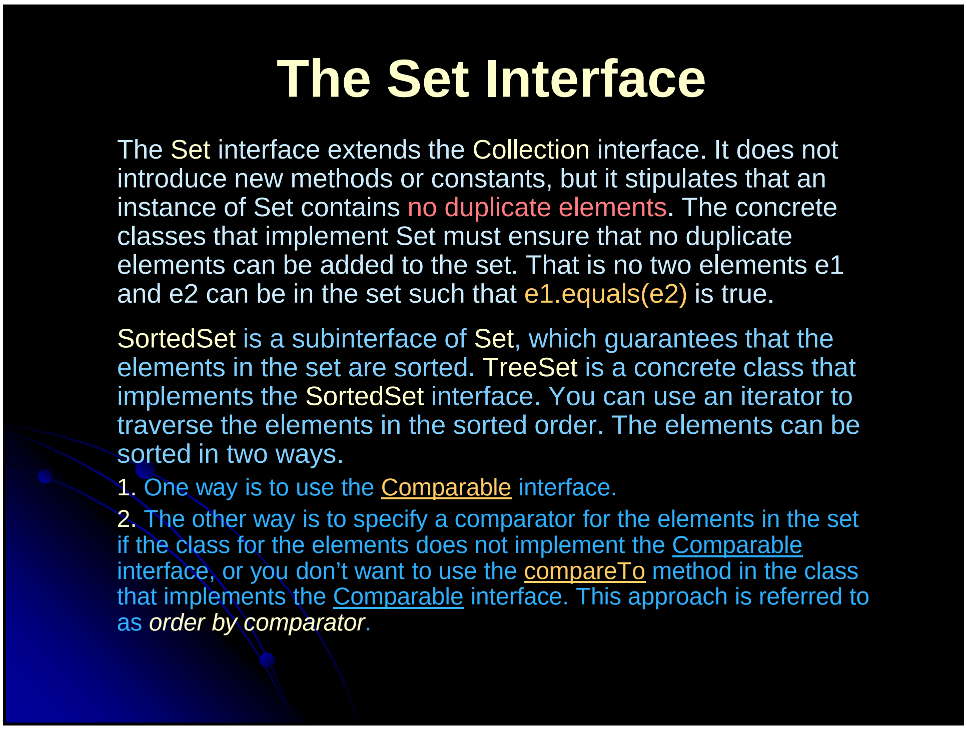
TestHashSet

Run



Useful Hashtable Methods

- put/get
  - Stores or retrieves a value in the hashtable
- remove/clear
  - Removes a particular entry or all entries from the hashtable
- containsKey/contains
  - Determines if the hashtable contains a particular key or element
- keys/elements
  - Returns an enumeration of all keys or elements, respectively
- size
  - Returns the number of elements in the hashtable
- rehash
  - Increases the capacity of the hashtable and reorganizes it



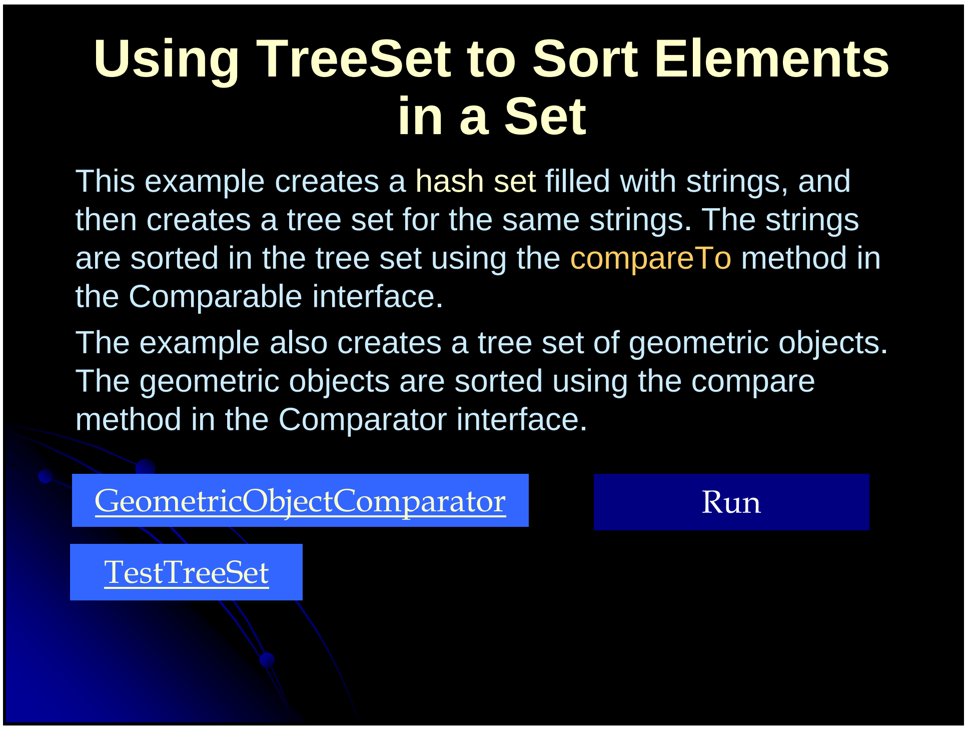
The Set Interface

The Set interface extends the Collection interface. It does not introduce new methods or constants, but it stipulates that an instance of Set contains no duplicate elements. The concrete classes that implement Set must ensure that no duplicate elements can be added to the set. That is no two elements e1 and e2 can be in the set such that e1.equals(e2) is true.

SortedSet is a subinterface of Set, which guarantees that the elements in the set are sorted. TreeSet is a concrete class that implements the SortedSet interface. You can use an iterator to traverse the elements in the sorted order. The elements can be sorted in two ways.

1. One way is to use the Comparable interface.

2. The other way is to specify a comparator for the elements in the set if the class for the elements does not implement the Comparable interface, or you don't want to use the compareTo method in the class that implements the Comparable interface. This approach is referred to as *order by comparator*.



Using TreeSet to Sort Elements in a Set

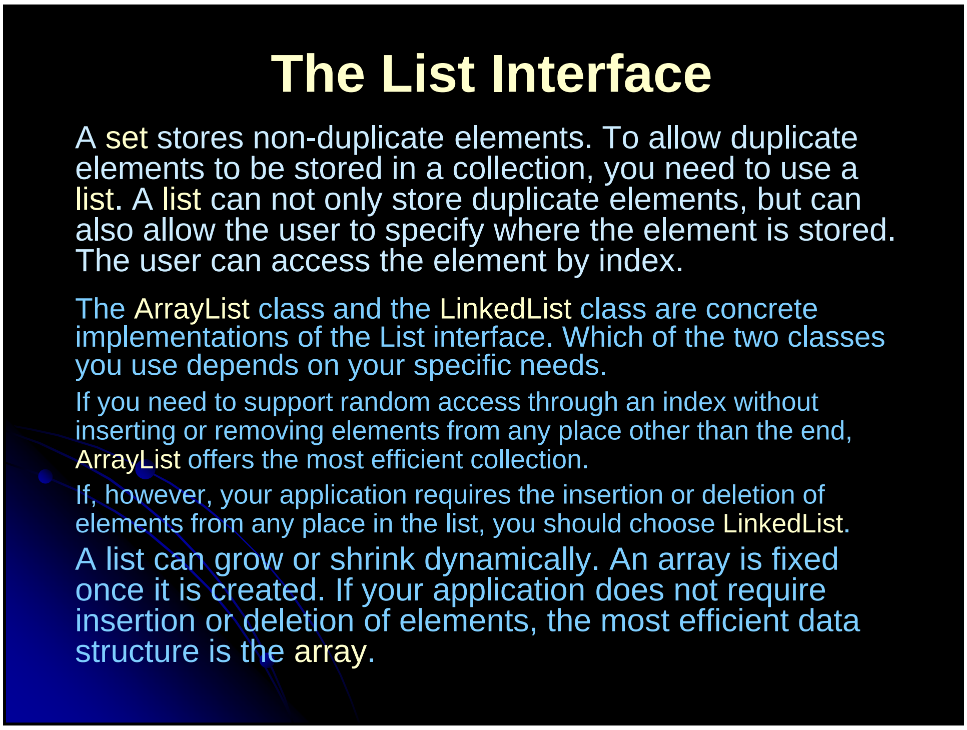
This example creates a hash set filled with strings, and then creates a tree set for the same strings. The strings are sorted in the tree set using the compareTo method in the Comparable interface.

The example also creates a tree set of geometric objects. The geometric objects are sorted using the compare method in the Comparator interface.

GeometricObjectComparator

Run

TestTreeSet



The List Interface

A set stores non-duplicate elements. To allow duplicate elements to be stored in a collection, you need to use a list. A list can not only store duplicate elements, but can also allow the user to specify where the element is stored. The user can access the element by index.

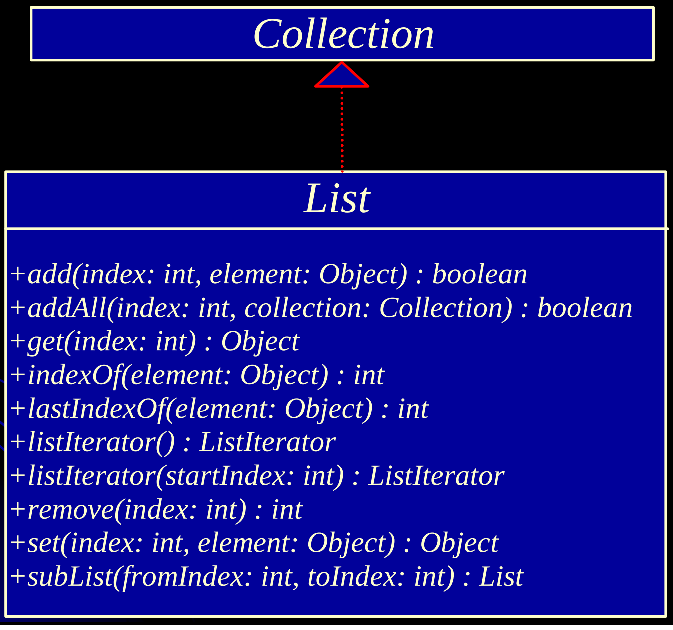
The ArrayList class and the LinkedList class are concrete implementations of the List interface. Which of the two classes you use depends on your specific needs.

If you need to support random access through an index without inserting or removing elements from any place other than the end, ArrayList offers the most efficient collection.

If, however, your application requires the insertion or deletion of elements from any place in the list, you should choose LinkedList.

A list can grow or shrink dynamically. An array is fixed once it is created. If your application does not require insertion or deletion of elements, the most efficient data structure is the array.

The List Interface

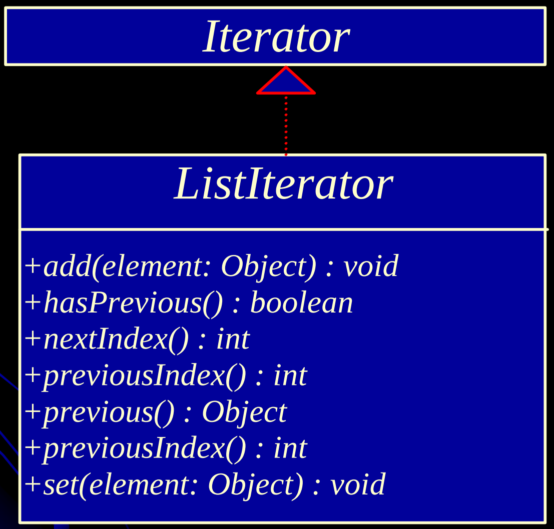


Collection

List

+add(index: int, element: Object) : boolean
+addAll(index: int, collection: Collection) : boolean
+get(index: int) : Object
+indexOf(element: Object) : int
+lastIndexOf(element: Object) : int
+listIterator() : ListIterator
+listIterator(startIndex: int) : ListIterator
+remove(index: int) : int
+set(index: int, element: Object) : Object
+subList(fromIndex: int, toIndex: int) : List

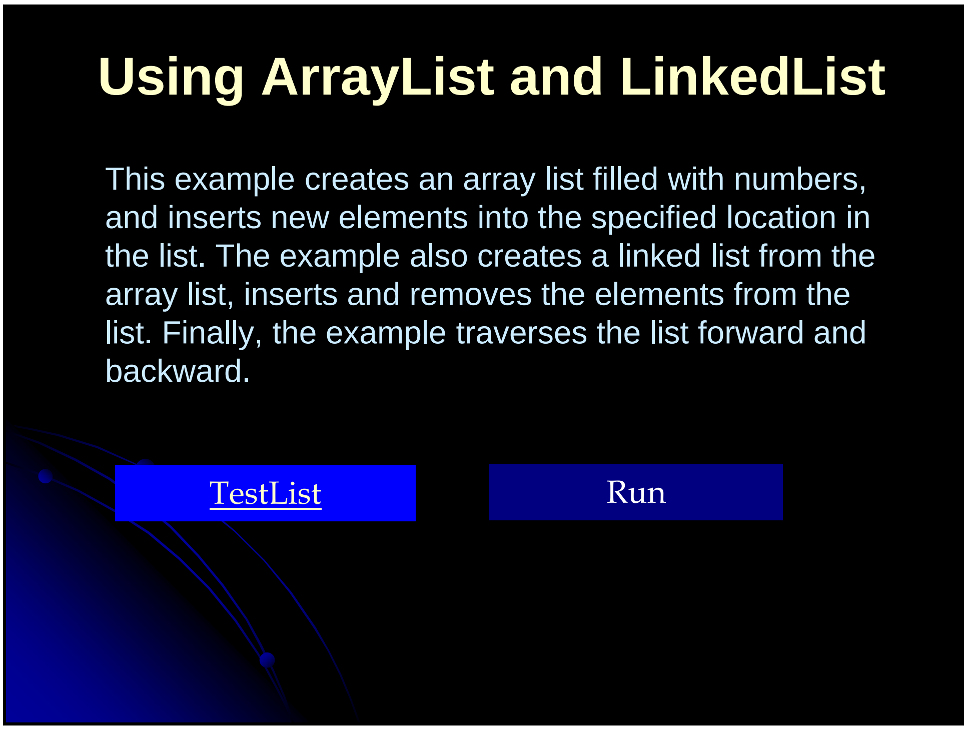
The List Iterator



Iterator

ListIterator

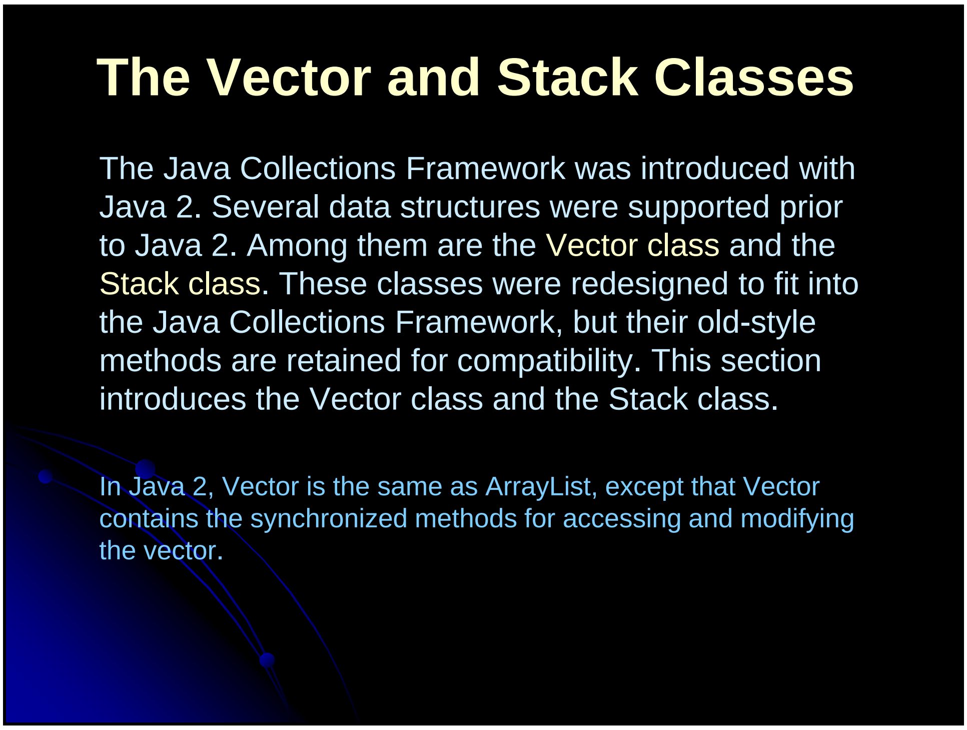
+add(element: Object) : void
+hasPrevious() : boolean
+nextIndex() : int
+previousIndex() : int
+previous() : Object
+previousIndex() : int
+set(element: Object) : void



Using ArrayList and LinkedList

This example creates an array list filled with numbers, and inserts new elements into the specified location in the list. The example also creates a linked list from the array list, inserts and removes the elements from the list. Finally, the example traverses the list forward and backward.

TestList
Run

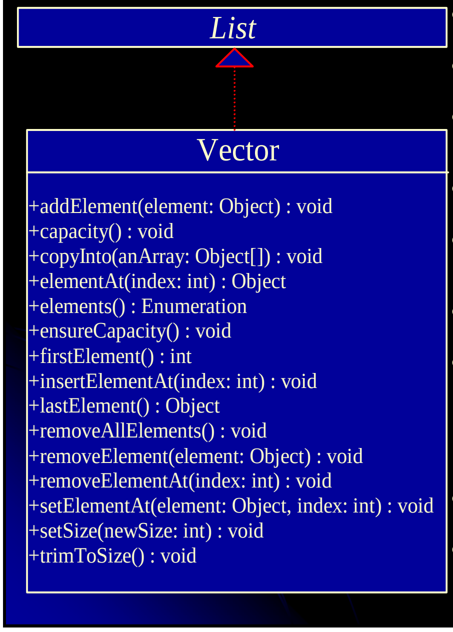


The Vector and Stack Classes

The Java Collections Framework was introduced with Java 2. Several data structures were supported prior to Java 2. Among them are the Vector class and the Stack class. These classes were redesigned to fit into the Java Collections Framework, but their old-style methods are retained for compatibility. This section introduces the Vector class and the Stack class.

In Java 2, Vector is the same as ArrayList, except that Vector contains the synchronized methods for accessing and modifying the vector.

The Vector Class



List

Vector

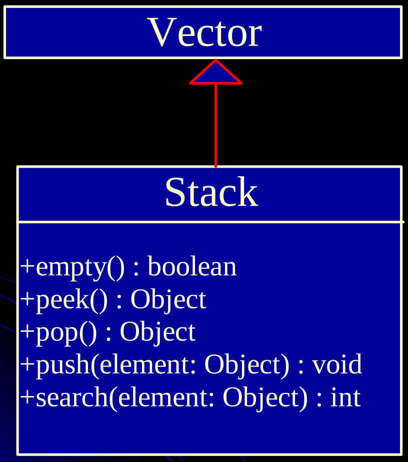
+addElement(element: Object) : void
+capacity() : void
+copyInto(anArray: Object[]) : void
+elementAt(index: int) : Object
+elements() : Enumeration
+ensureCapacity() : void
+firstElement() : int
+insertElementAt(index: int) : void
+lastElement() : Object
+removeAllElements() : void
+removeElement(element: Object) : void
+removeElementAt(index: int) : void
+setElementAt(element: Object, index: int) : void
+setSize(newSize: int) : void
+trimToSize() : void

- addElement/insertElementAt/setElementAt
  Add elements to the vector
- removeElement/removeElementAt
  Removes an element from the vector
- firstElement/lastElement
  Returns a reference to the first and last element, respectively (without removing)
- elementAt
  Returns the element at the specified index
- indexOf
  Returns the index of an element that equals the object specified
- contains
  Determines if the vector contains an object
- elements
  Returns an `Enumeration` of objects in the vector

```
Enumeration elements = vector.elements();
while(elements.hasMoreElements()) {
System.out.println(elements.nextElement());
}
```

- size
  The number of elements in the vector
- capacity
  The number of elements the vector can hold before becoming resized

The Stack Class



Vector

Stack

+empty() : boolean
+peek() : Object
+pop() : Object
+push(element: Object) : void
+search(element: Object) : int

The Stack class represents a last-in-first-out stack of objects. The elements are accessed only from the top of the stack. You can retrieve, insert, or remove an element from the top of the stack.

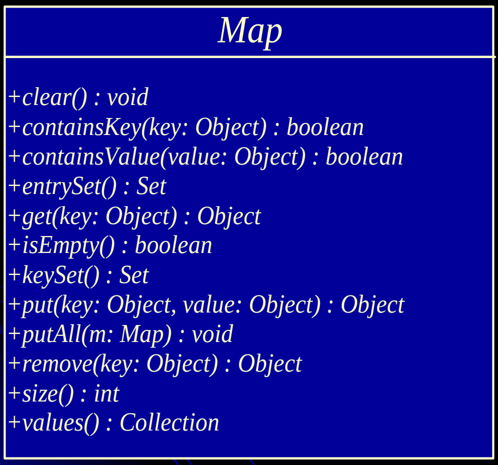
Using Vector and Stack

The program reads student scores from the keyboard, stores the scores in the vector, finds the best scores, and then assigns grades for all the students. A negative score signals the end of the input.

AssignGradeUsingVector

Run

The Map Interface



Map

+clear() : void
+containsKey(key: Object) : boolean
+containsValue(value: Object) : boolean
+entrySet() : Set
+get(key: Object) : Object
+isEmpty() : boolean
+keySet() : Set
+put(key: Object, value: Object) : Object
+putAll(m: Map) : void
+remove(key: Object) : Object
+size() : int
+values() : Collection

The Map interface maps keys to the elements. The keys are like indexes. In List, the indexes are integer. In Map, the keys can be any objects.

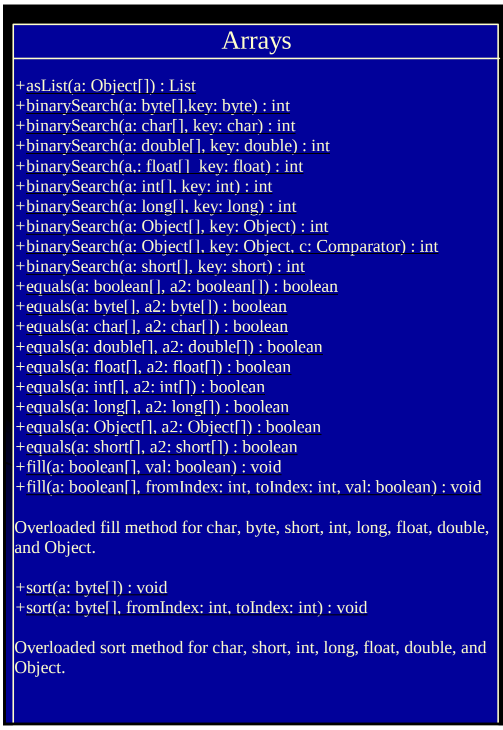
The HashMap and TreeMap classes are two concrete implementations of the Map interface. The HashMap class is efficient for locating a value, inserting a mapping, and deleting a mapping. The TreeMap class, implementing SortedMap, is efficient for traversing the keys in a sorted order.

Using HashMap and TreeMap

This example creates a hash map that maps borrowers to mortgages. The program first creates a hash map with the borrower’s name as its key and mortgage as its value. The program then creates a tree map from the hash map, and displays the mappings in ascending order of the keys.

TestMap

Run



Arrays

```
+asList(a: Object[]) : List
+binarySearch(a: byte[],key: byte) : int
+binarySearch(a: char[], key: char) : int
+binarySearch(a: double[], key: double) : int
+binarySearch(a,: float[]  key: float) : int
+binarySearch(a: int[], key: int) : int
+binarySearch(a: long[], key: long) : int
+binarySearch(a: Object[], key: Object) : int
+binarySearch(a: Object[], key: Object, c: Comparator) : int
+binarySearch(a: short[], key: short) : int
+equals(a: boolean[], a2: boolean[]) : boolean
+equals(a: byte[], a2: byte[]) : boolean
+equals(a: char[], a2: char[]) : boolean
+equals(a: double[], a2: double[]) : boolean
+equals(a: float[], a2: float[]) : boolean
+equals(a: int[], a2: int[]) : boolean
+equals(a: long[], a2: long[]) : boolean
+equals(a: Object[], a2: Object[]) : boolean
+equals(a: short[], a2: short[]) : boolean
+fill(a: boolean[], val: boolean) : void
+fill(a: boolean[], fromIndex: int, toIndex: int, val: boolean) : void
```

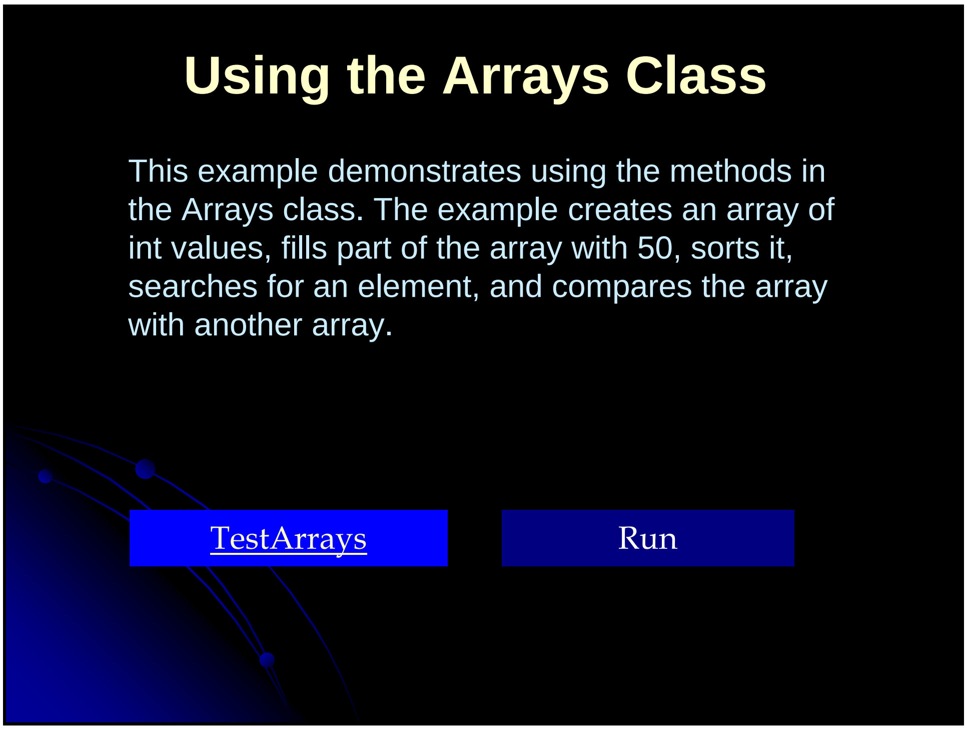
Overloaded fill method for char, byte, short, int, long, float, double, and Object.

```
+sort(a: byte[]) : void
+sort(a: byte[], fromIndex: int, toIndex: int) : void
```

Overloaded sort method for char, short, int, long, float, double, and Object.

The Arrays Class

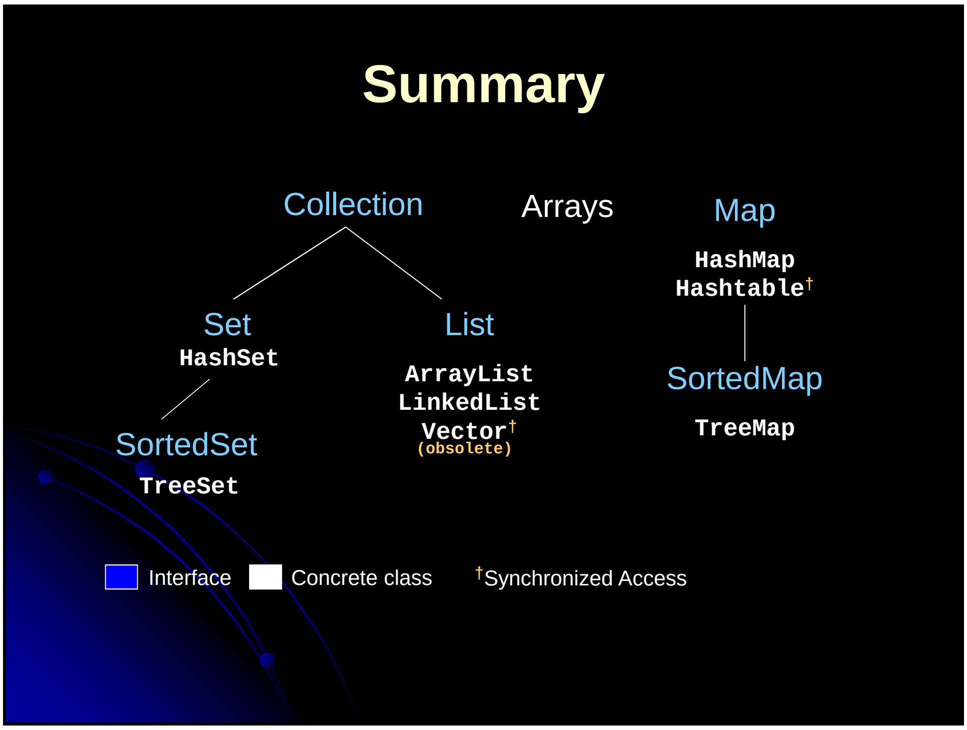
The Arrays class contains various static methods for sorting and searching arrays, for comparing arrays, and for filling array elements. It also contains a method for converting an array to a list.



Using the Arrays Class

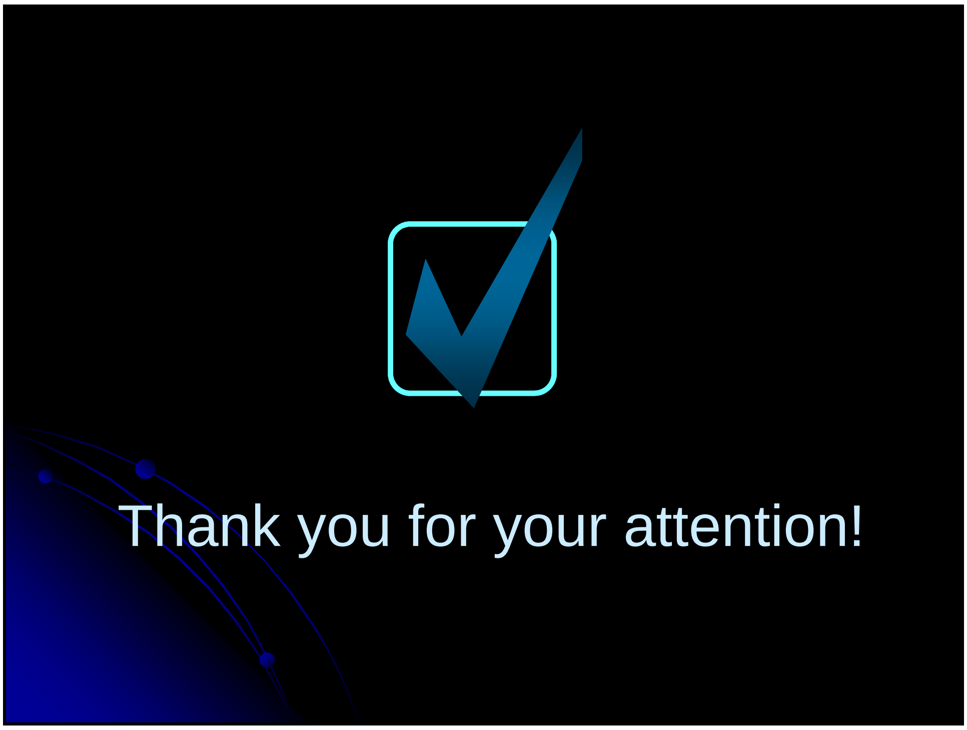
This example demonstrates using the methods in the Arrays class. The example creates an array of int values, fills part of the array with 50, sorts it, searches for an element, and compares the array with another array.

TestArrays
Run

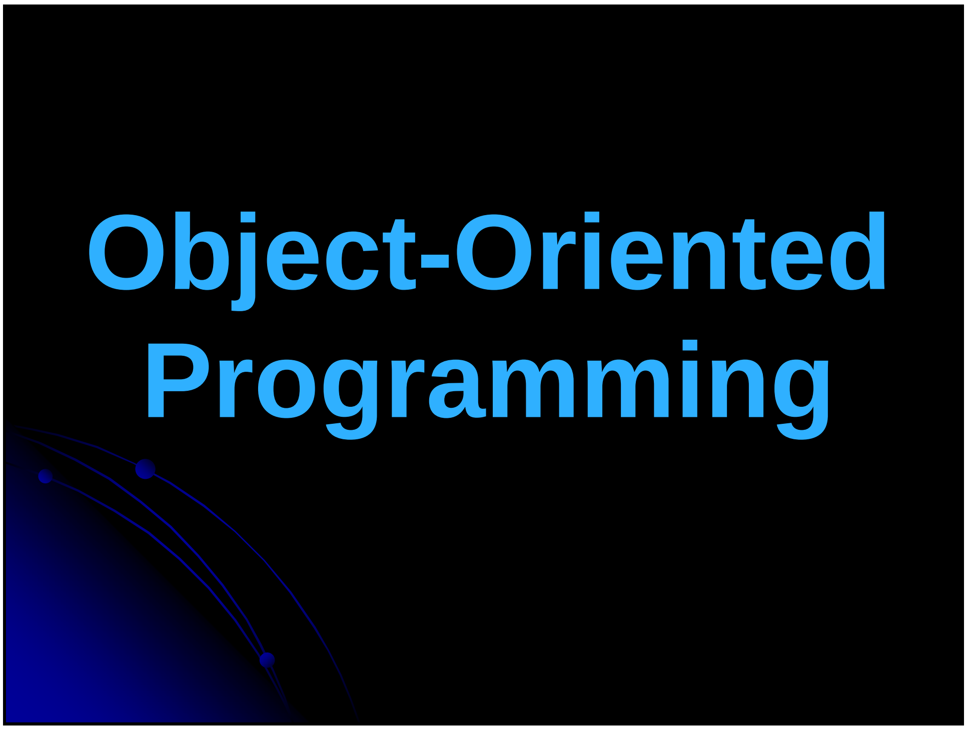


Summary

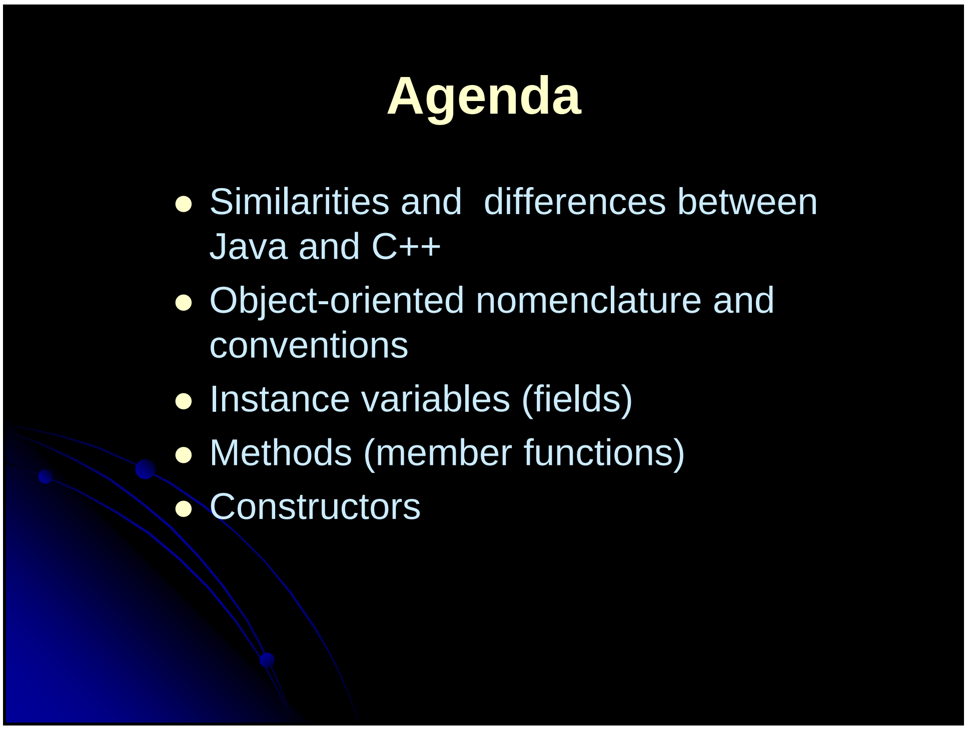
🟦 Interface ⬜ Concrete class †Synchronized Access



Thank you for your attention!

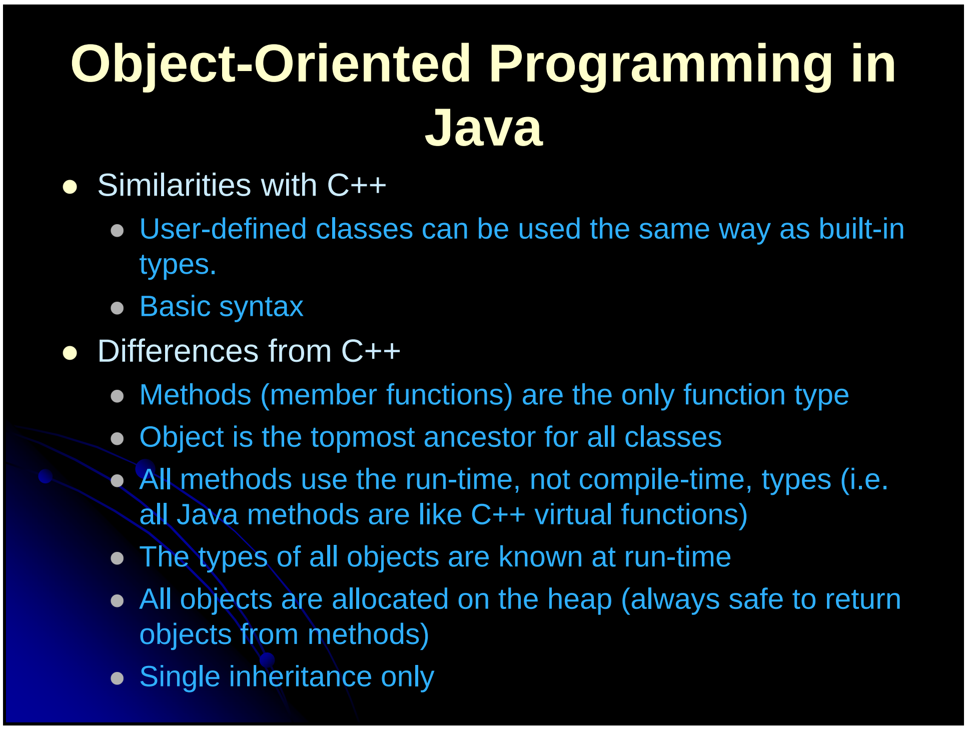


Object-Oriented Programming



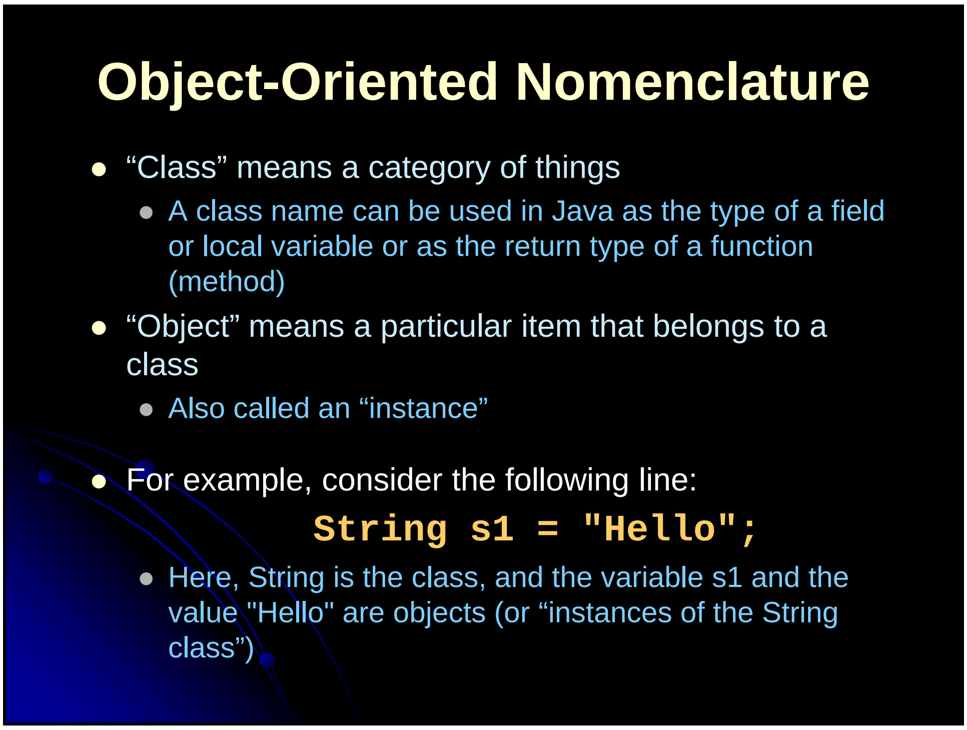
Agenda

- Similarities and differences between Java and C++
- Object-oriented nomenclature and conventions
- Instance variables (fields)
- Methods (member functions)
- Constructors



Object-Oriented Programming in Java

- Similarities with C++
  - User-defined classes can be used the same way as built-in types.
  - Basic syntax
- Differences from C++
  - Methods (member functions) are the only function type
  - Object is the topmost ancestor for all classes
  - All methods use the run-time, not compile-time, types (i.e. all Java methods are like C++ virtual functions)
  - The types of all objects are known at run-time
  - All objects are allocated on the heap (always safe to return objects from methods)
  - Single inheritance only

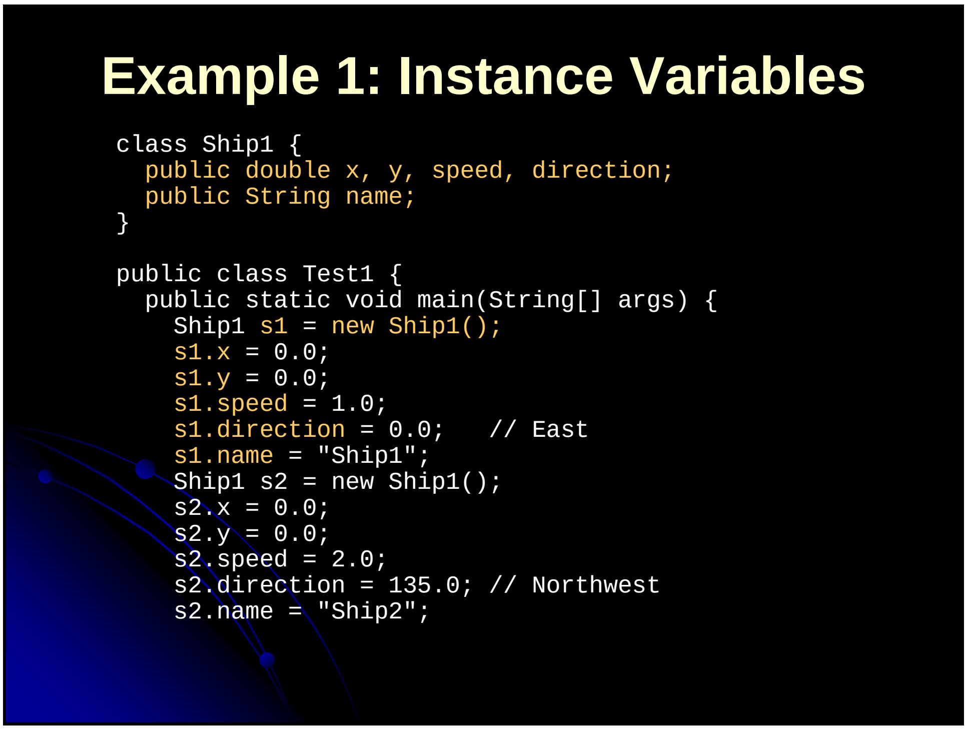


Object-Oriented Nomenclature

- “Class” means a category of things
  - A class name can be used in Java as the type of a field or local variable or as the return type of a function (method)
- “Object” means a particular item that belongs to a class
  - Also called an “instance”
- For example, consider the following line:

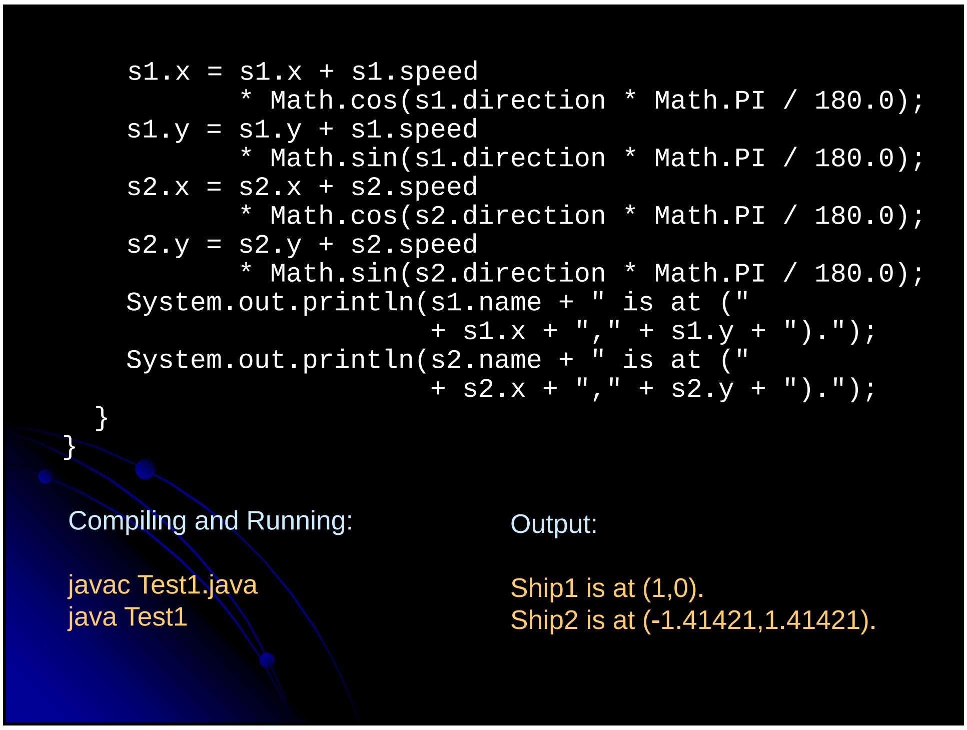
```
String s1 = "Hello";
```

  - Here, String is the class, and the variable s1 and the value "Hello" are objects (or “instances of the String class”)



Example 1: Instance Variables

```
class Ship1 {
  public double x, y, speed, direction;
  public String name;
}

public class Test1 {
  public static void main(String[] args) {
    Ship1 s1 = new Ship1();
    s1.x = 0.0;
    s1.y = 0.0;
    s1.speed = 1.0;
    s1.direction = 0.0;   // East
    s1.name = "Ship1";
    Ship1 s2 = new Ship1();
    s2.x = 0.0;
    s2.y = 0.0;
    s2.speed = 2.0;
    s2.direction = 135.0; // Northwest
    s2.name = "Ship2";
```



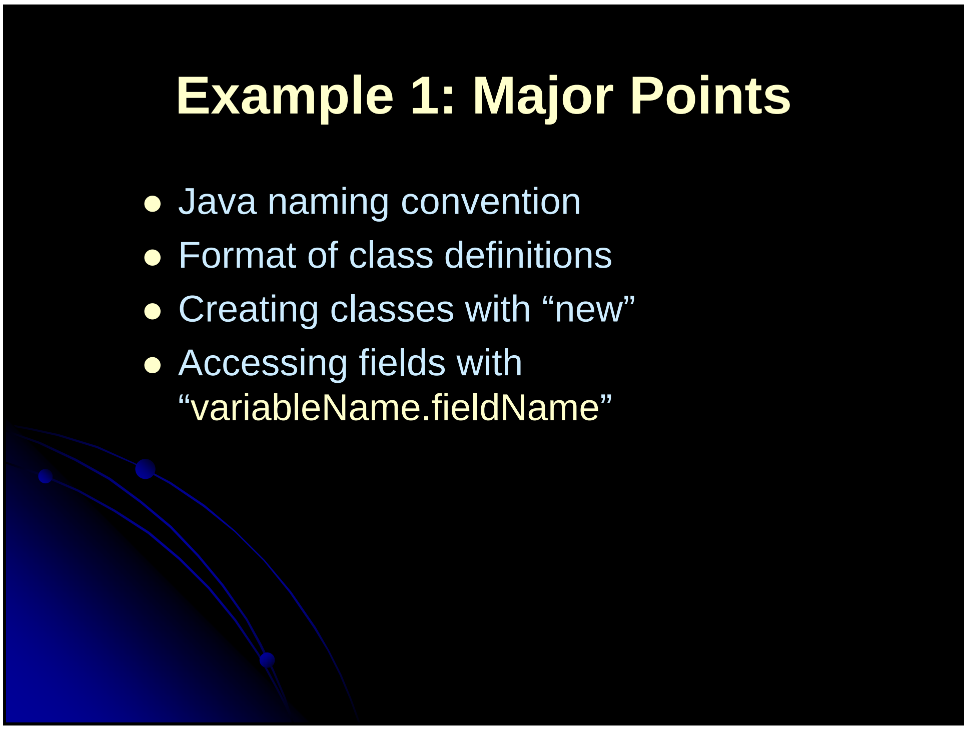
```
    s1.x = s1.x + s1.speed
           * Math.cos(s1.direction * Math.PI / 180.0);
    s1.y = s1.y + s1.speed
           * Math.sin(s1.direction * Math.PI / 180.0);
    s2.x = s2.x + s2.speed
           * Math.cos(s2.direction * Math.PI / 180.0);
    s2.y = s2.y + s2.speed
           * Math.sin(s2.direction * Math.PI / 180.0);
    System.out.println(s1.name + " is at ("
                       + s1.x + "," + s1.y + ").");
    System.out.println(s2.name + " is at ("
                       + s2.x + "," + s2.y + ").");
  }
}
```

Compiling and Running:

```
javac Test1.java
java Test1
```

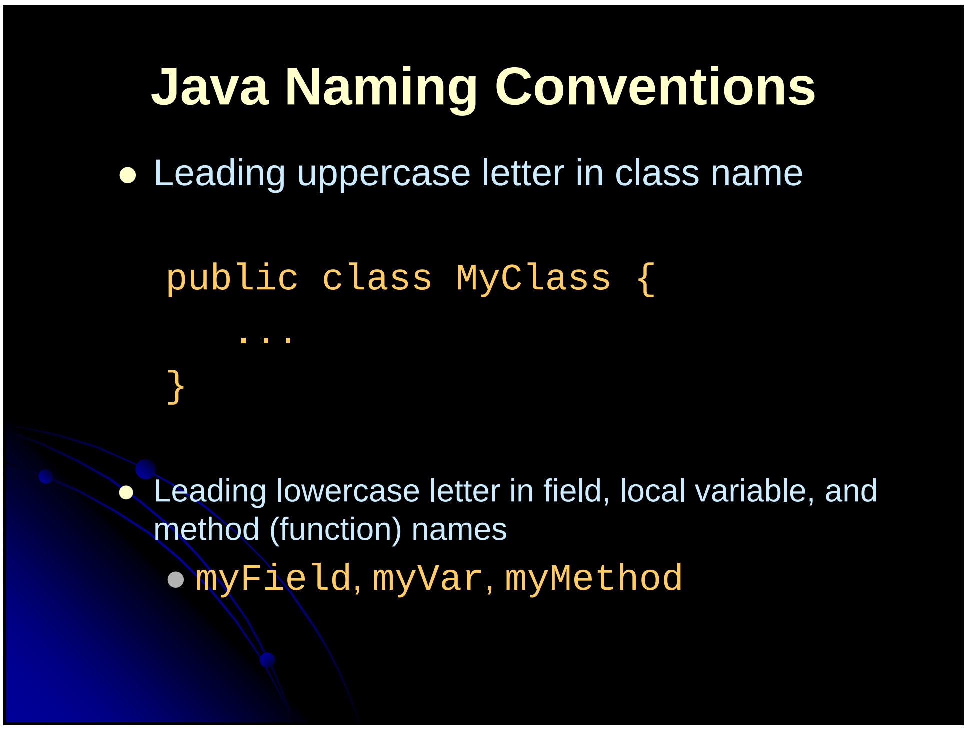
Output:

```
Ship1 is at (1,0).
Ship2 is at (-1.41421,1.41421).
```



Example 1: Major Points

- Java naming convention
- Format of class definitions
- Creating classes with “new”
- Accessing fields with “variableName.fieldName”

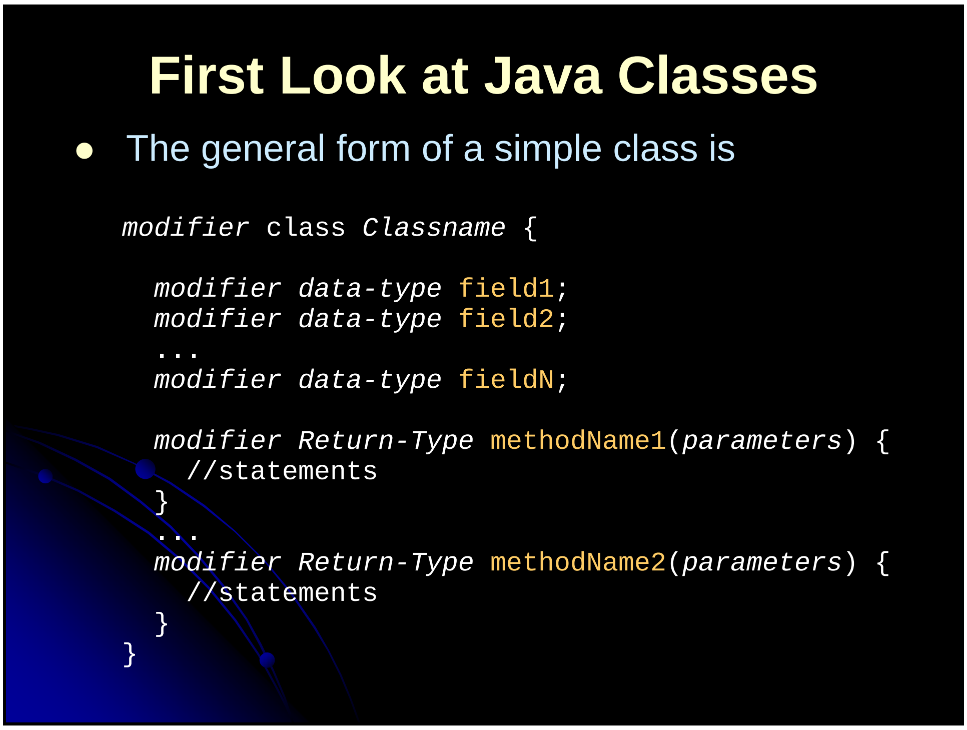


Java Naming Conventions

- Leading uppercase letter in class name

```
public class MyClass {
   ...
}
```

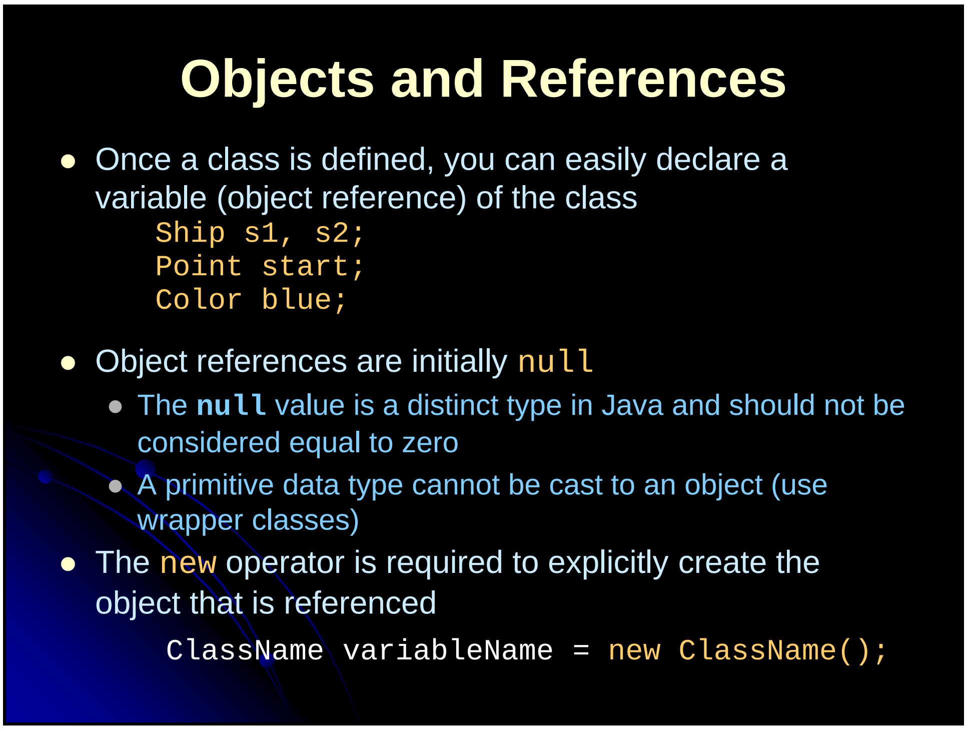
- Leading lowercase letter in field, local variable, and method (function) names
  - `myField`, `myVar`, `myMethod`



First Look at Java Classes

- The general form of a simple class is

```
modifier class Classname {

  modifier data-type field1;
  modifier data-type field2;
  ...
  modifier data-type fieldN;

  modifier Return-Type methodName1(parameters) {
    //statements
  }
  ...
  modifier Return-Type methodName2(parameters) {
    //statements
  }
}
```



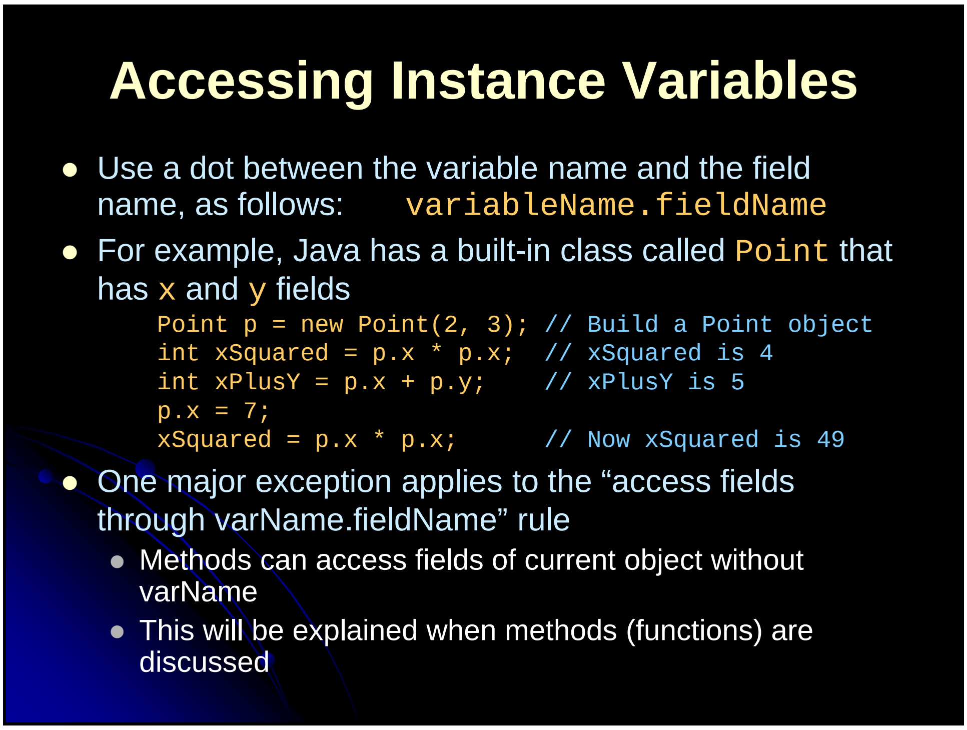
Objects and References

- Once a class is defined, you can easily declare a variable (object reference) of the class

```
    Ship s1, s2;
    Point start;
    Color blue;
```

- Object references are initially `null`
  - The **`null`** value is a distinct type in Java and should not be considered equal to zero
  - A primitive data type cannot be cast to an object (use wrapper classes)
- The new operator is required to explicitly create the object that is referenced

```
     ClassName variableName = new ClassName();
```

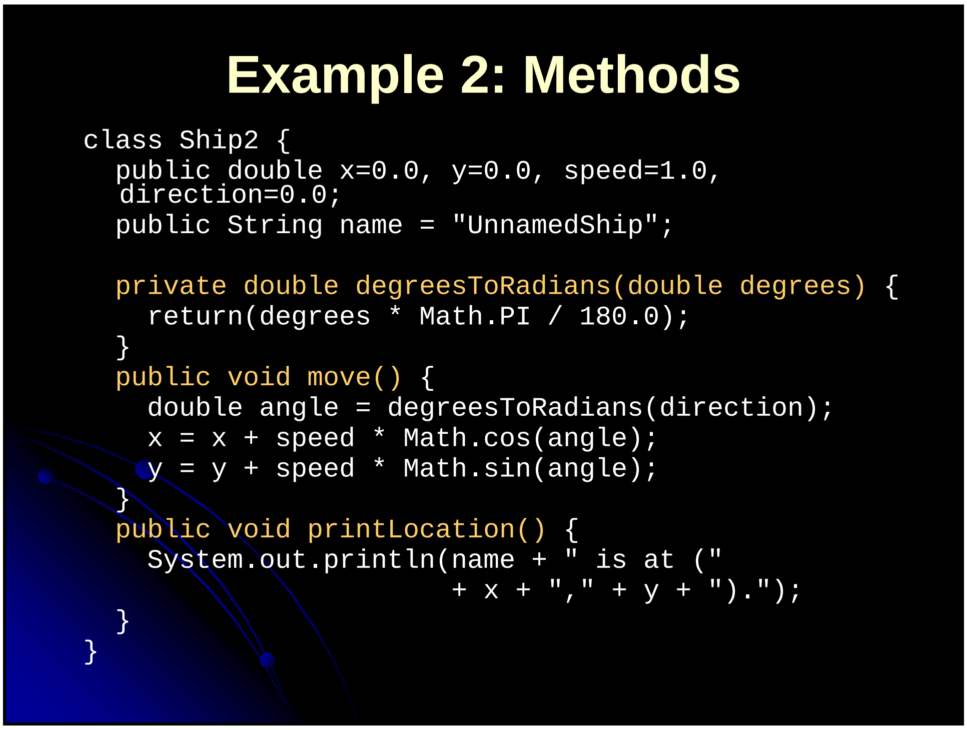


Accessing Instance Variables

- Use a dot between the variable name and the field name, as follows: `variableName.fieldName`
- For example, Java has a built-in class called `Point` that has `x` and `y` fields

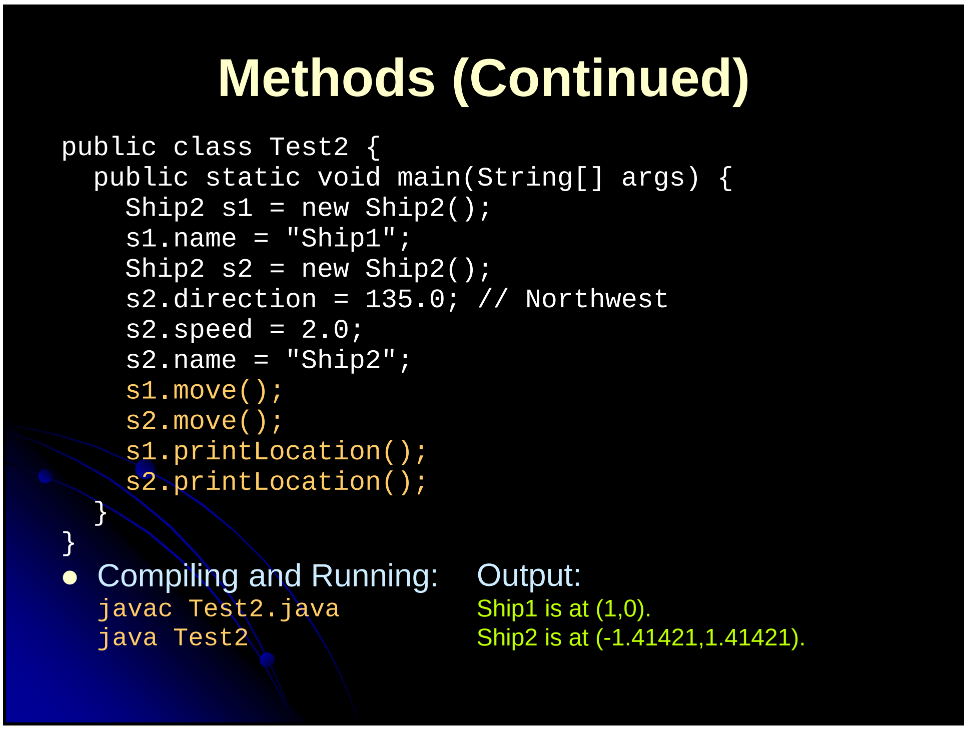
```
Point p = new Point(2, 3); // Build a Point object
int xSquared = p.x * p.x;  // xSquared is 4
int xPlusY = p.x + p.y;    // xPlusY is 5
p.x = 7;
xSquared = p.x * p.x;      // Now xSquared is 49
```

- One major exception applies to the “access fields through varName.fieldName” rule
  - Methods can access fields of current object without varName
  - This will be explained when methods (functions) are discussed



Example 2: Methods

```
class Ship2 {
  public double x=0.0, y=0.0, speed=1.0,
  direction=0.0;
  public String name = "UnnamedShip";

  private double degreesToRadians(double degrees) {
    return(degrees * Math.PI / 180.0);
  }
  public void move() {
    double angle = degreesToRadians(direction);
    x = x + speed * Math.cos(angle);
    y = y + speed * Math.sin(angle);
  }
  public void printLocation() {
    System.out.println(name + " is at ("
                       + x + "," + y + ").");
  }
}
```



Methods (Continued)

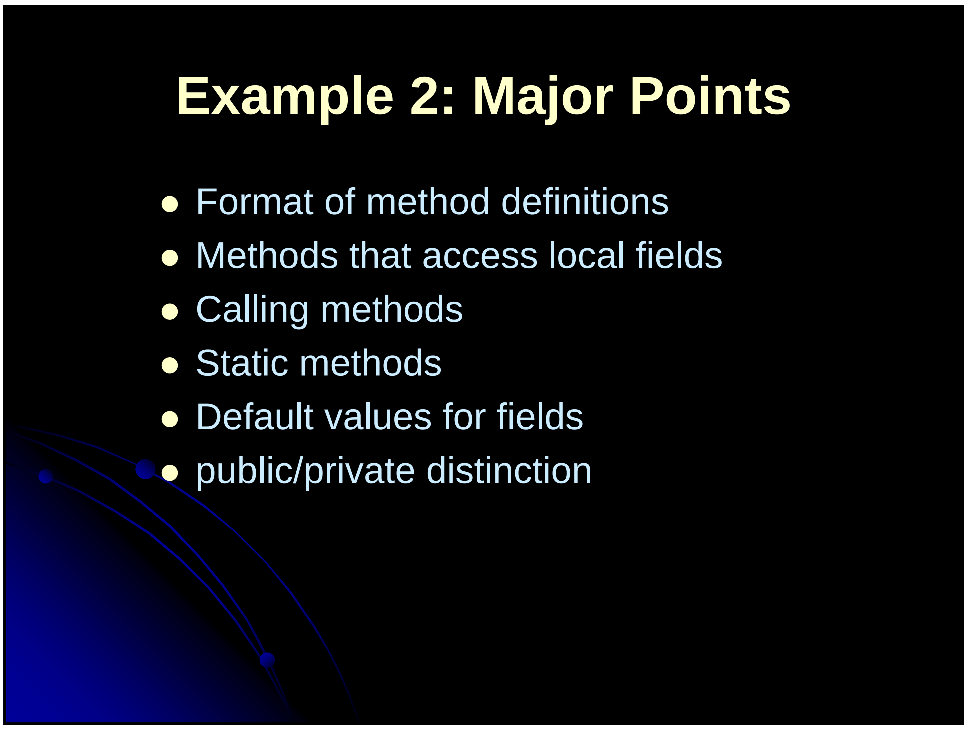
```
public class Test2 {
  public static void main(String[] args) {
    Ship2 s1 = new Ship2();
    s1.name = "Ship1";
    Ship2 s2 = new Ship2();
    s2.direction = 135.0; // Northwest
    s2.speed = 2.0;
    s2.name = "Ship2";
    s1.move();
    s2.move();
    s1.printLocation();
    s2.printLocation();
  }
}
```

- Compiling and Running:

```
javac Test2.java
java Test2
```

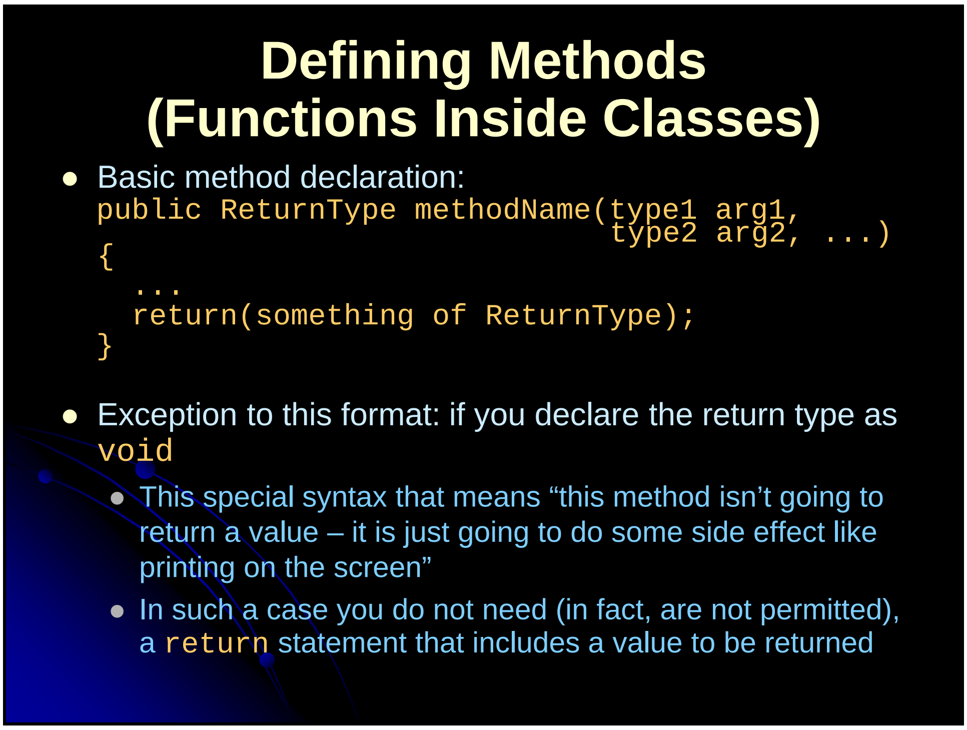
Output:

```
Ship1 is at (1,0).
Ship2 is at (-1.41421,1.41421).
```



Example 2: Major Points

- Format of method definitions
- Methods that access local fields
- Calling methods
- Static methods
- Default values for fields
- public/private distinction

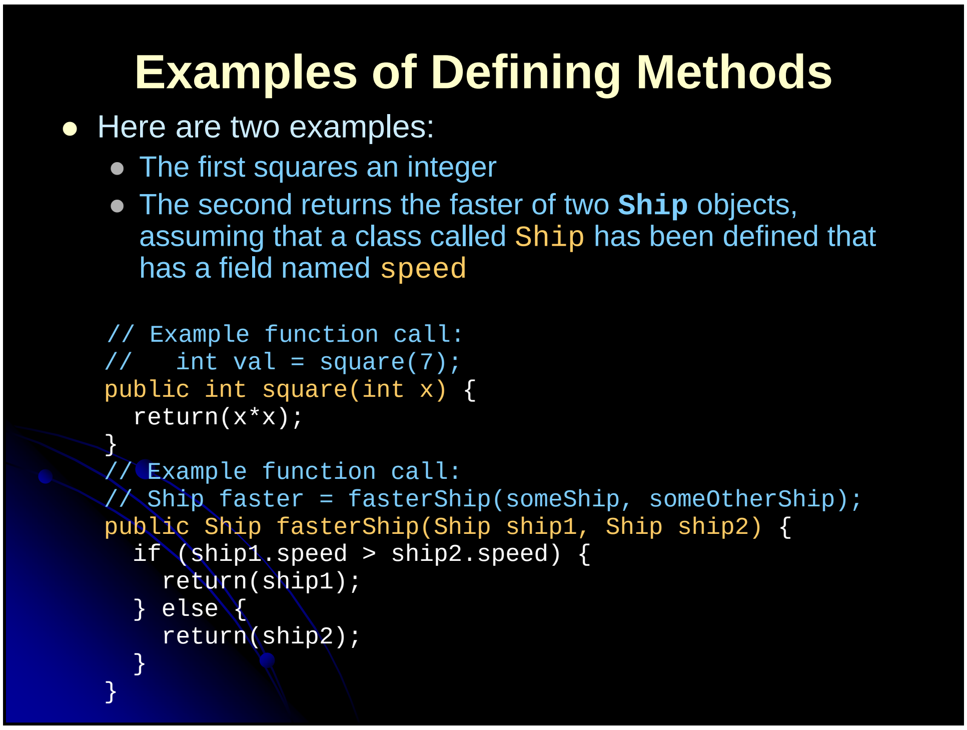


Defining Methods (Functions Inside Classes)

- Basic method declaration:

```
public ReturnType methodName(type1 arg1,
                             type2 arg2, ...)
{
  ...
  return(something of ReturnType);
}
```

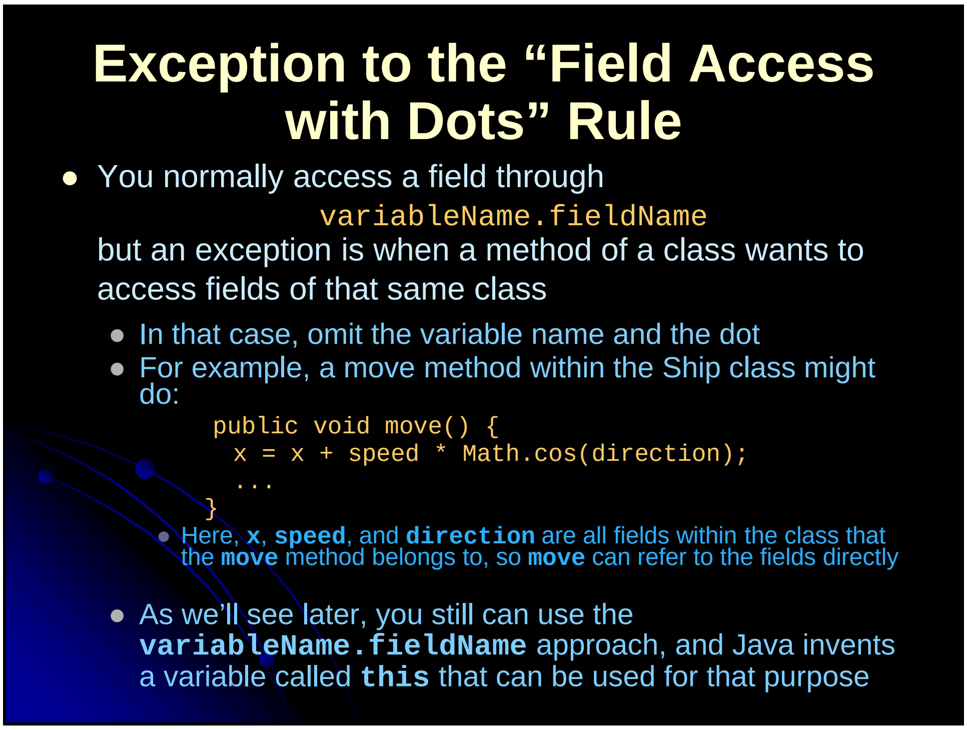
- Exception to this format: if you declare the return type as `void`
  - This special syntax that means “this method isn’t going to return a value – it is just going to do some side effect like printing on the screen”
  - In such a case you do not need (in fact, are not permitted), a `return` statement that includes a value to be returned



Examples of Defining Methods

- Here are two examples:
  - The first squares an integer
  - The second returns the faster of two **`Ship`** objects, assuming that a class called `Ship` has been defined that has a field named `speed`

```
// Example function call:
//   int val = square(7);
public int square(int x) {
  return(x*x);
}
// Example function call:
// Ship faster = fasterShip(someShip, someOtherShip);
public Ship fasterShip(Ship ship1, Ship ship2) {
  if (ship1.speed > ship2.speed) {
    return(ship1);
  } else {
    return(ship2);
  }
}
```



Exception to the “Field Access with Dots” Rule

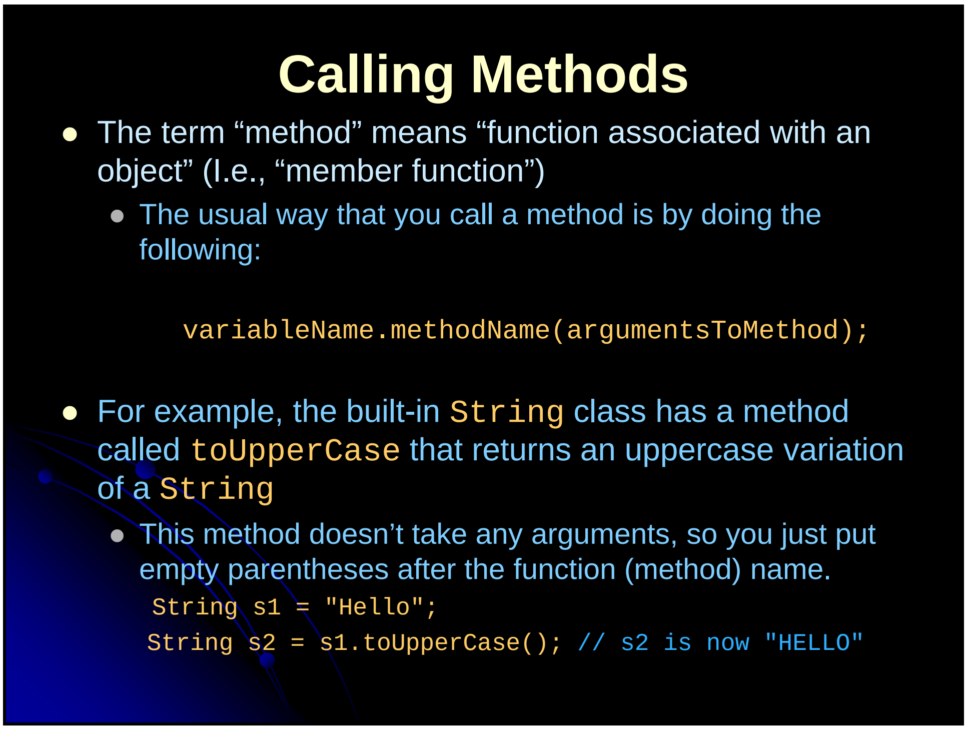
- You normally access a field through

  `variableName.fieldName`

  but an exception is when a method of a class wants to access fields of that same class
  - In that case, omit the variable name and the dot
  - For example, a move method within the Ship class might do:

    ```
    public void move() {
      x = x + speed * Math.cos(direction);
      ...
    }
    ```

    - Here, **`x`**, **`speed`**, and **`direction`** are all fields within the class that the **`move`** method belongs to, so **`move`** can refer to the fields directly
  - As we’ll see later, you still can use the **`variableName.fieldName`** approach, and Java invents a variable called **`this`** that can be used for that purpose



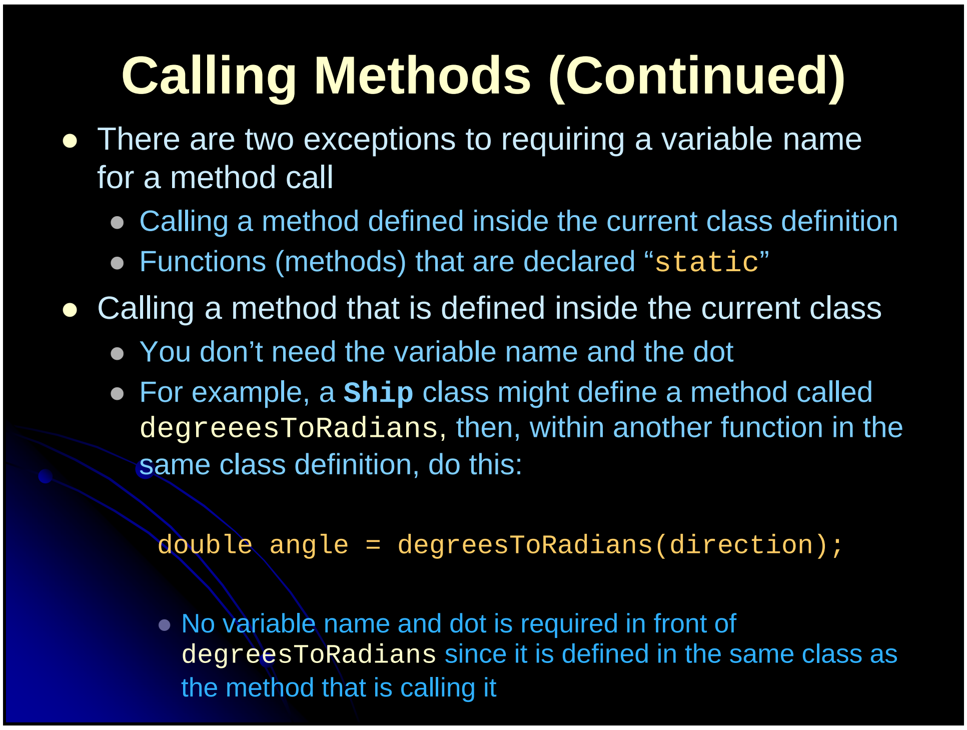
Calling Methods

- The term “method” means “function associated with an object” (I.e., “member function”)
  - The usual way that you call a method is by doing the following:

    ```
    variableName.methodName(argumentsToMethod);
    ```

- For example, the built-in `String` class has a method called `toUpperCase` that returns an uppercase variation of a `String`
  - This method doesn’t take any arguments, so you just put empty parentheses after the function (method) name.

    ```
    String s1 = "Hello";
    String s2 = s1.toUpperCase(); // s2 is now "HELLO"
    ```

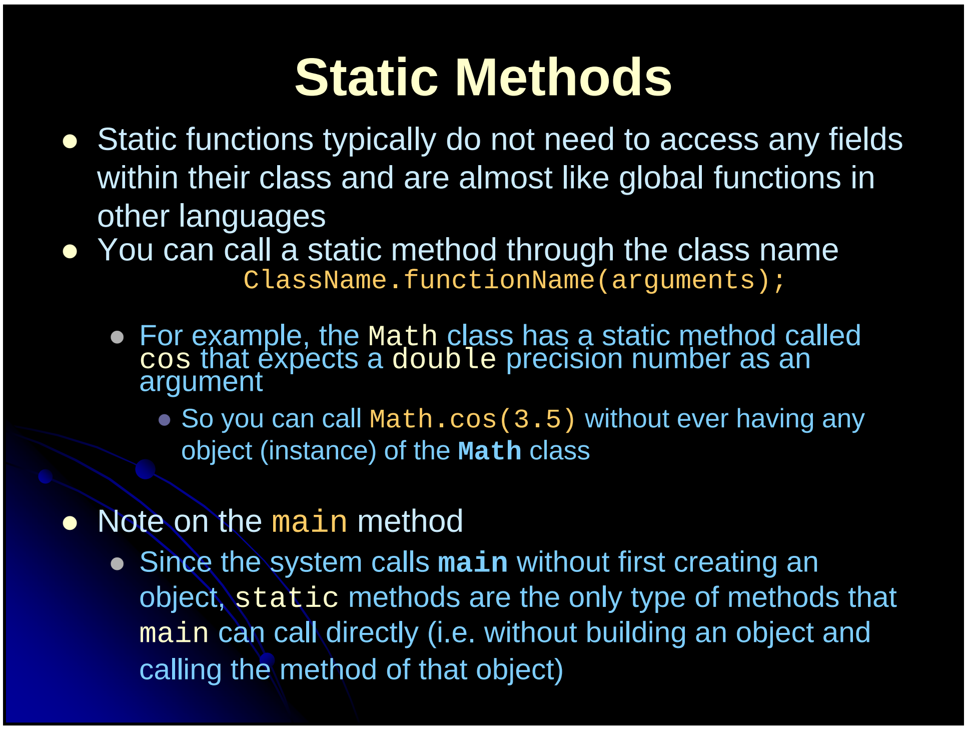


Calling Methods (Continued)

- There are two exceptions to requiring a variable name for a method call
  - Calling a method defined inside the current class definition
  - Functions (methods) that are declared “`static`”
- Calling a method that is defined inside the current class
  - You don’t need the variable name and the dot
  - For example, a **`Ship`** class might define a method called `degreeesToRadians`, then, within another function in the same class definition, do this:

```
double angle = degreesToRadians(direction);
```

    - No variable name and dot is required in front of `degreesToRadians` since it is defined in the same class as the method that is calling it

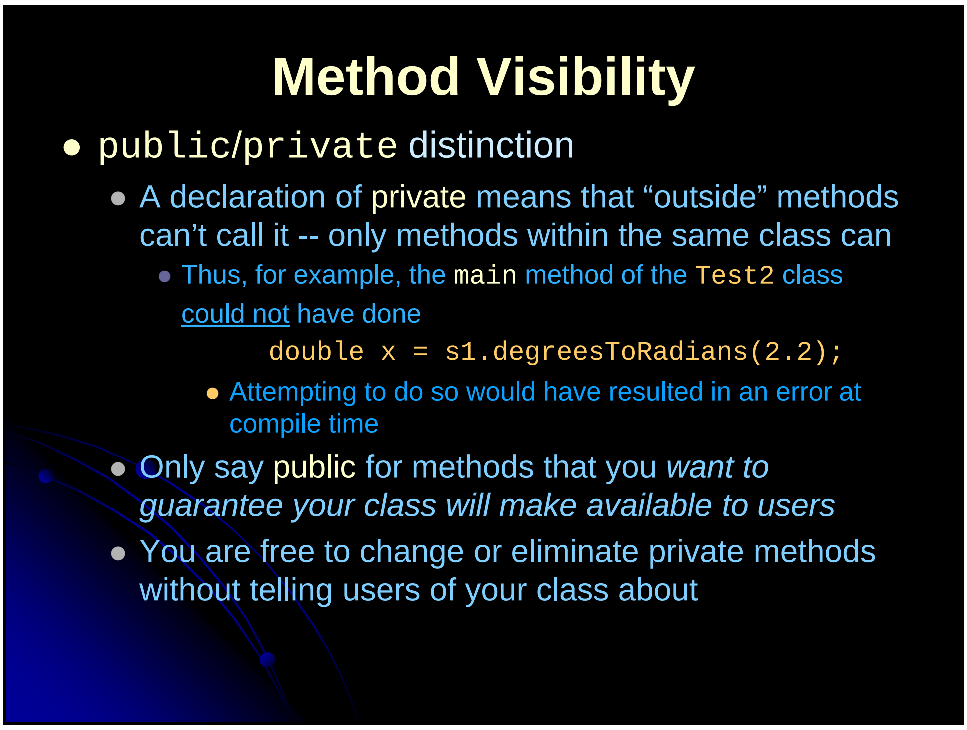


Static Methods

- Static functions typically do not need to access any fields within their class and are almost like global functions in other languages
- You can call a static method through the class name

```
ClassName.functionName(arguments);
```

  - For example, the `Math` class has a static method called `cos` that expects a `double` precision number as an argument
    - So you can call `Math.cos(3.5)` without ever having any object (instance) of the **`Math`** class
- Note on the `main` method
  - Since the system calls **`main`** without first creating an object, `static` methods are the only type of methods that `main` can call directly (i.e. without building an object and calling the method of that object)

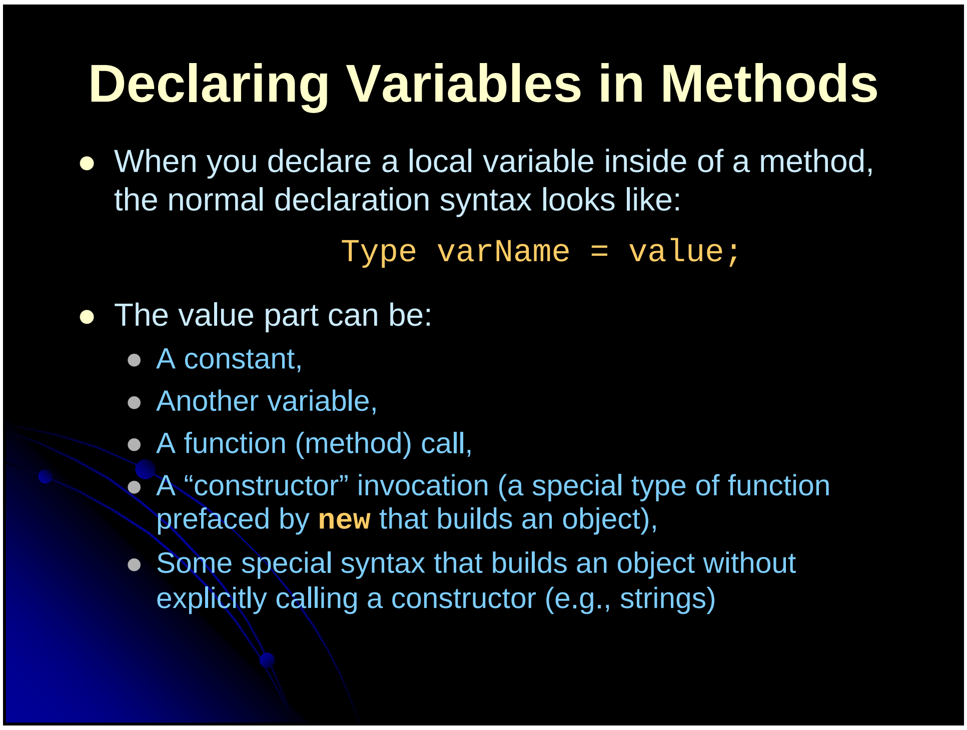


Method Visibility

- `public/private` distinction
  - A declaration of private means that “outside” methods can’t call it -- only methods within the same class can
    - Thus, for example, the `main` method of the `Test2` class <u>could not</u> have done

      ```
      double x = s1.degreesToRadians(2.2);
      ```

      - Attempting to do so would have resulted in an error at compile time
  - Only say public for methods that you *want to guarantee your class will make available to users*
  - You are free to change or eliminate private methods without telling users of your class about

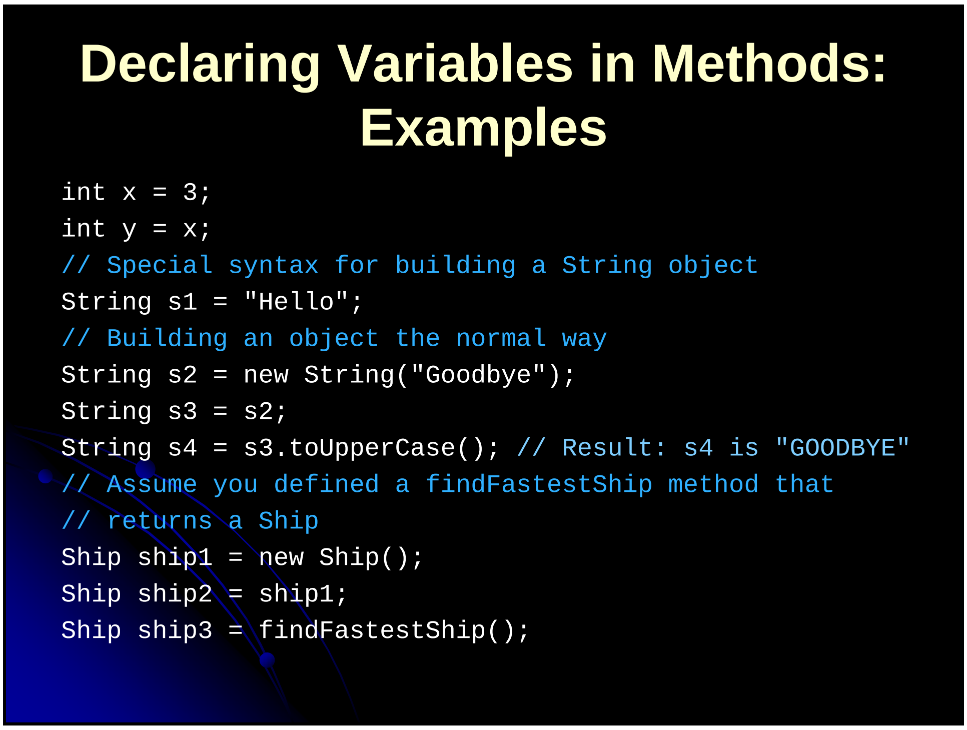


Declaring Variables in Methods

- When you declare a local variable inside of a method, the normal declaration syntax looks like:

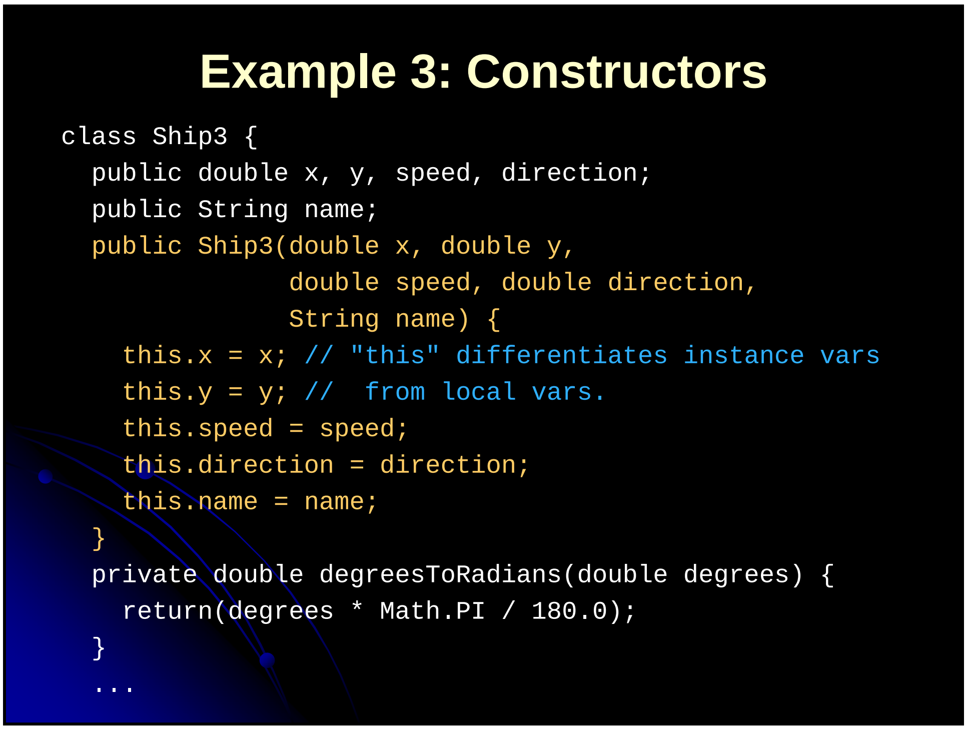
  ```
  Type varName = value;
  ```

- The value part can be:
  - A constant,
  - Another variable,
  - A function (method) call,
  - A “constructor” invocation (a special type of function prefaced by **new** that builds an object),
  - Some special syntax that builds an object without explicitly calling a constructor (e.g., strings)



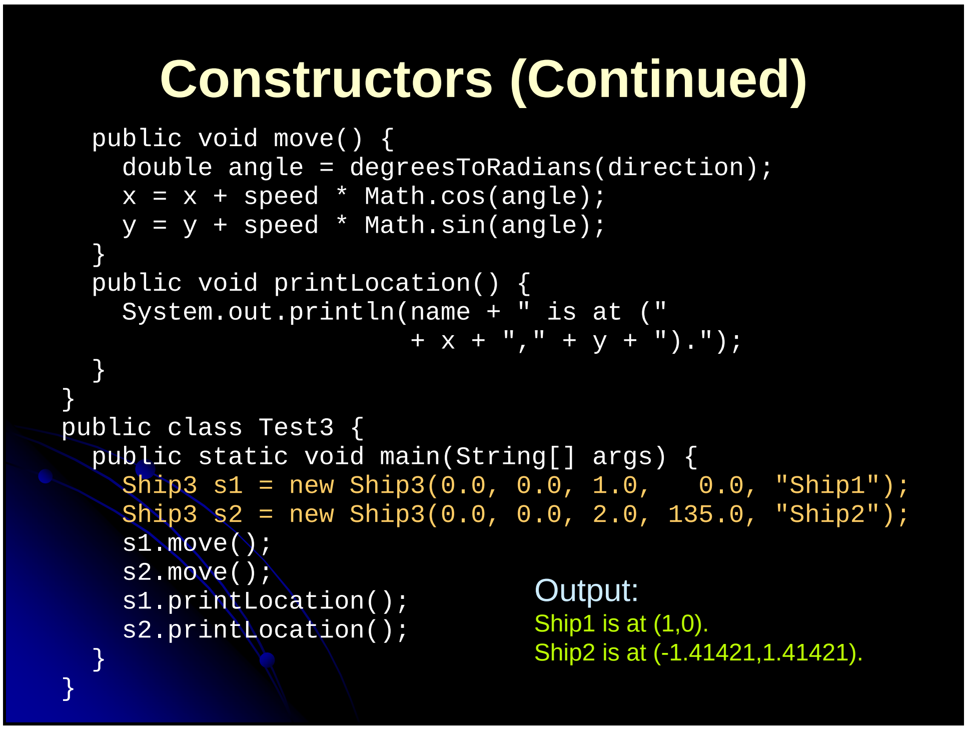
Declaring Variables in Methods: Examples

```
int x = 3;
int y = x;
// Special syntax for building a String object
String s1 = "Hello";
// Building an object the normal way
String s2 = new String("Goodbye");
String s3 = s2;
String s4 = s3.toUpperCase(); // Result: s4 is "GOODBYE"
// Assume you defined a findFastestShip method that
// returns a Ship
Ship ship1 = new Ship();
Ship ship2 = ship1;
Ship ship3 = findFastestShip();
```



Example 3: Constructors

```
class Ship3 {
  public double x, y, speed, direction;
  public String name;
  public Ship3(double x, double y,
               double speed, double direction,
               String name) {
    this.x = x; // "this" differentiates instance vars
    this.y = y; //  from local vars.
    this.speed = speed;
    this.direction = direction;
    this.name = name;
  }
  private double degreesToRadians(double degrees) {
    return(degrees * Math.PI / 180.0);
  }
  ...
```

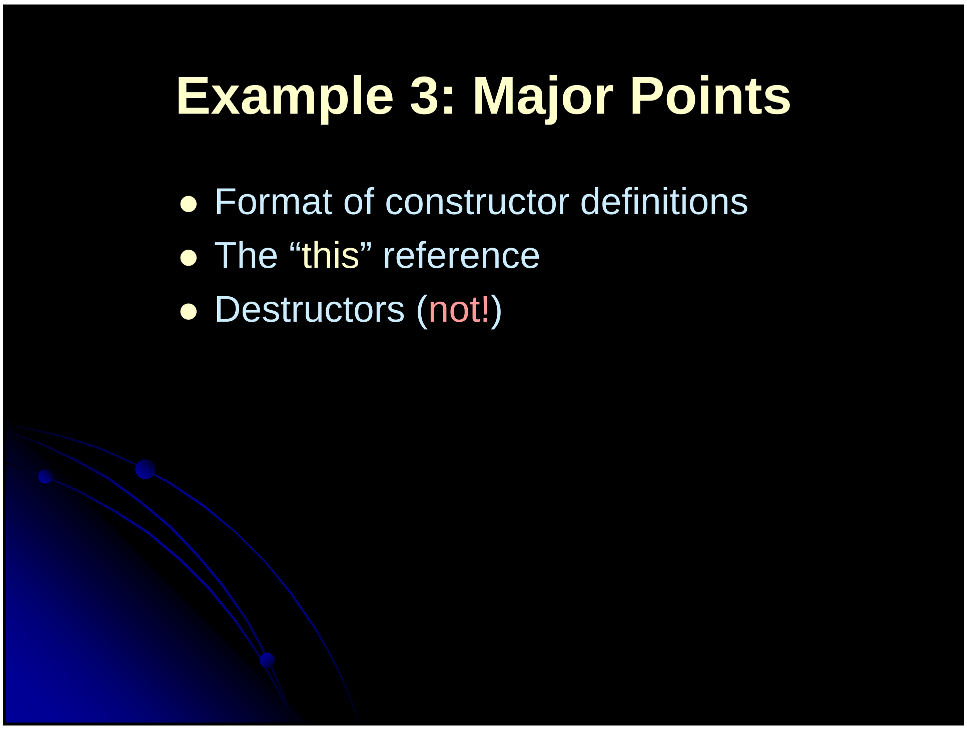


Constructors (Continued)

```
  public void move() {
    double angle = degreesToRadians(direction);
    x = x + speed * Math.cos(angle);
    y = y + speed * Math.sin(angle);
  }
  public void printLocation() {
    System.out.println(name + " is at ("
                       + x + "," + y + ").");
  }
}
public class Test3 {
  public static void main(String[] args) {
    Ship3 s1 = new Ship3(0.0, 0.0, 1.0,   0.0, "Ship1");
    Ship3 s2 = new Ship3(0.0, 0.0, 2.0, 135.0, "Ship2");
    s1.move();
    s2.move();
    s1.printLocation();
    s2.printLocation();
  }
}
```

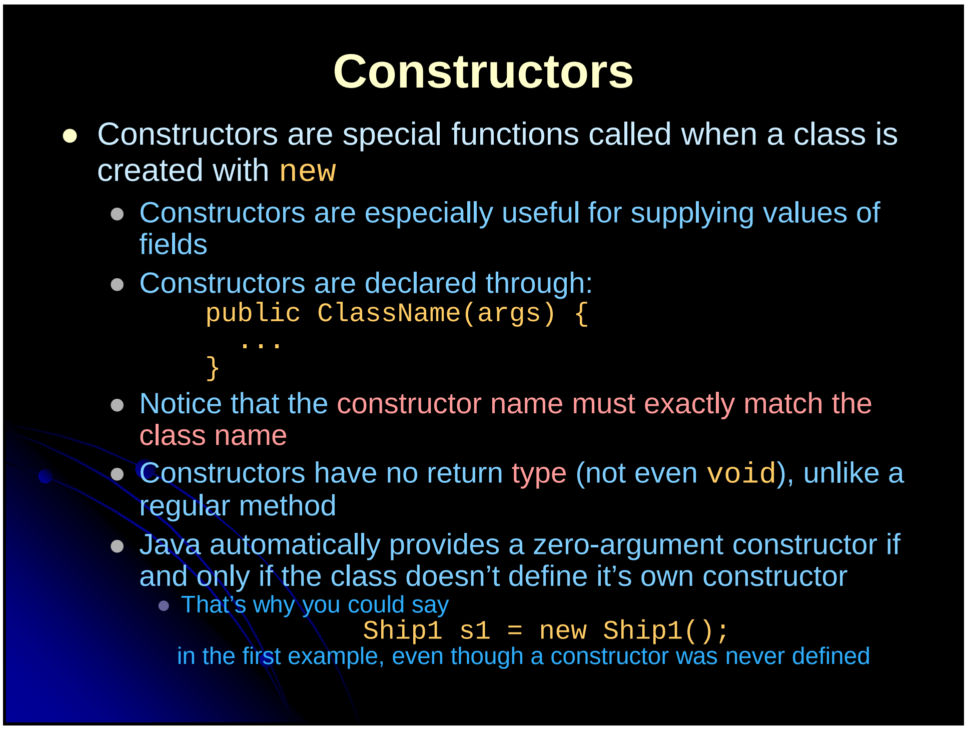
Output:

```
Ship1 is at (1,0).
Ship2 is at (-1.41421,1.41421).
```



Example 3: Major Points

- Format of constructor definitions
- The “this” reference
- Destructors (not!)



Constructors

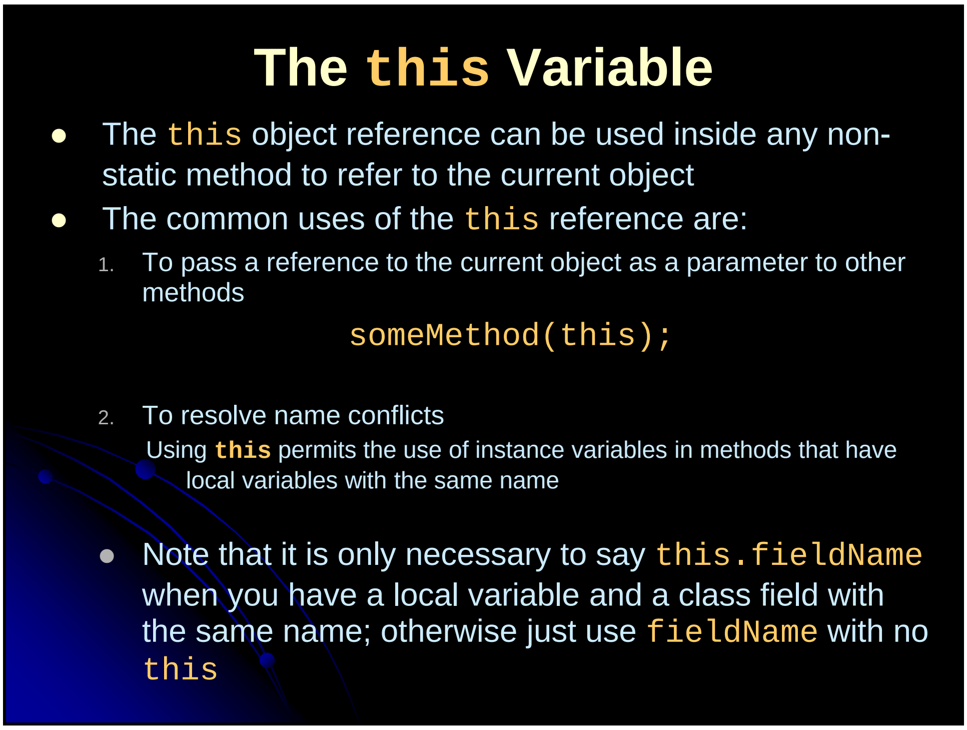
- Constructors are special functions called when a class is created with `new`
  - Constructors are especially useful for supplying values of fields
  - Constructors are declared through:

    ```
    public ClassName(args) {
      ...
    }
    ```

  - Notice that the constructor name must exactly match the class name
  - Constructors have no return type (not even `void`), unlike a regular method
  - Java automatically provides a zero-argument constructor if and only if the class doesn't define it's own constructor
    - That's why you could say

      ```
      Ship1 s1 = new Ship1();
      ```

      in the first example, even though a constructor was never defined



The `this` Variable

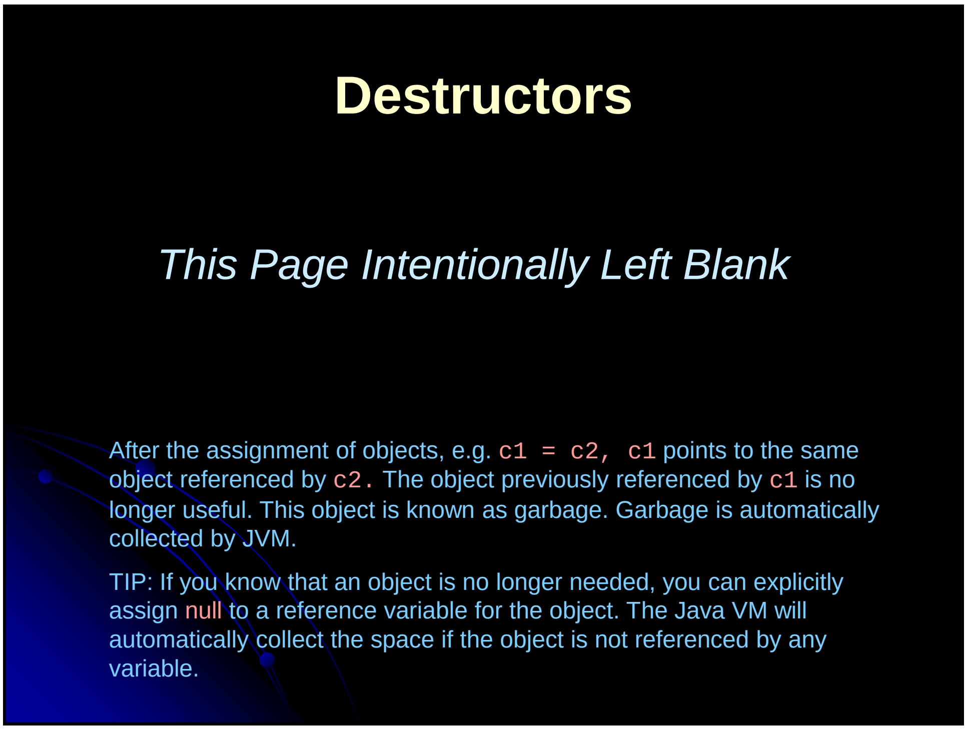
- The `this` object reference can be used inside any non-static method to refer to the current object
- The common uses of the `this` reference are:
  1. To pass a reference to the current object as a parameter to other methods

     ```
     someMethod(this);
     ```

  2. To resolve name conflicts

     Using **`this`** permits the use of instance variables in methods that have local variables with the same name

  - Note that it is only necessary to say `this.fieldName` when you have a local variable and a class field with the same name; otherwise just use `fieldName` with no `this`

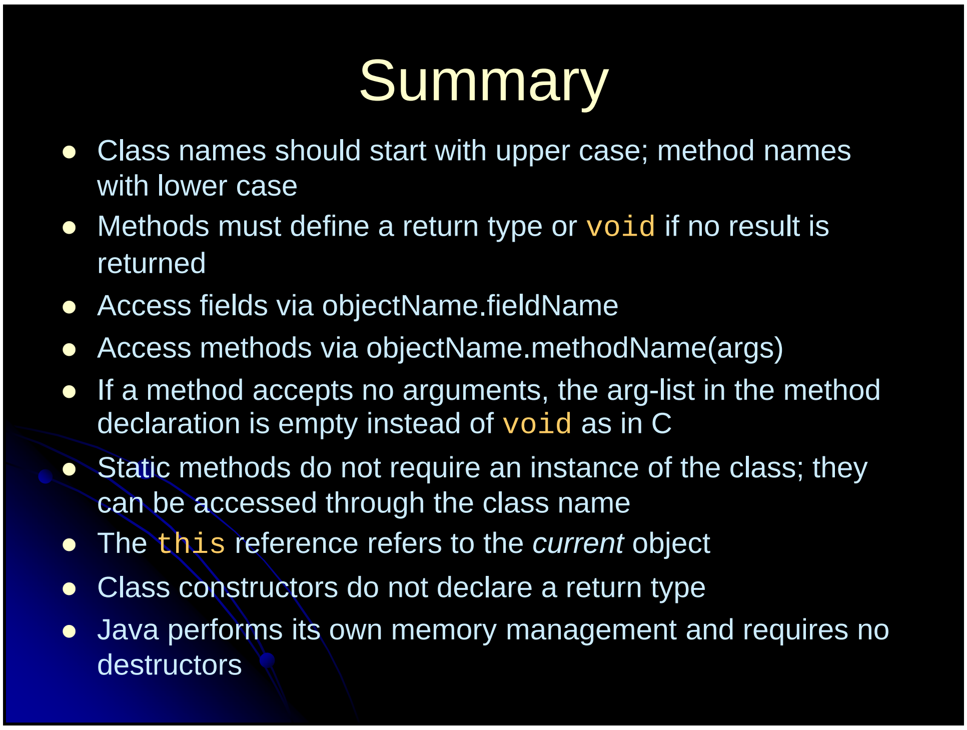


Destructors

This Page Intentionally Left Blank

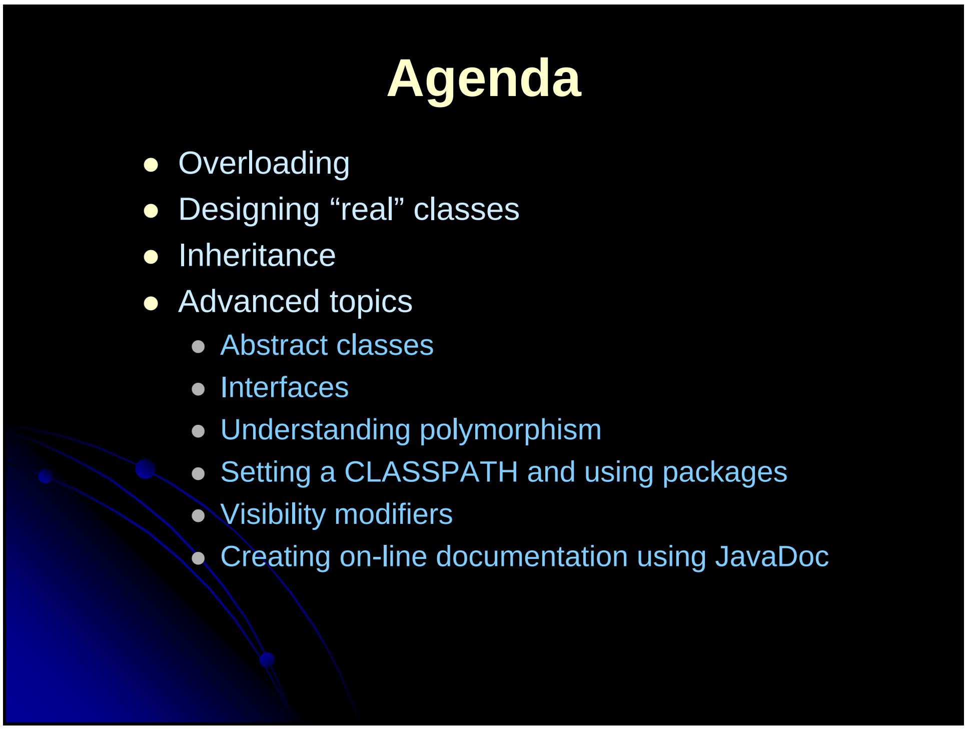
After the assignment of objects, e.g. `c1 = c2, c1` points to the same object referenced by `c2.` The object previously referenced by `c1` is no longer useful. This object is known as garbage. Garbage is automatically collected by JVM.

TIP: If you know that an object is no longer needed, you can explicitly assign null to a reference variable for the object. The Java VM will automatically collect the space if the object is not referenced by any variable.



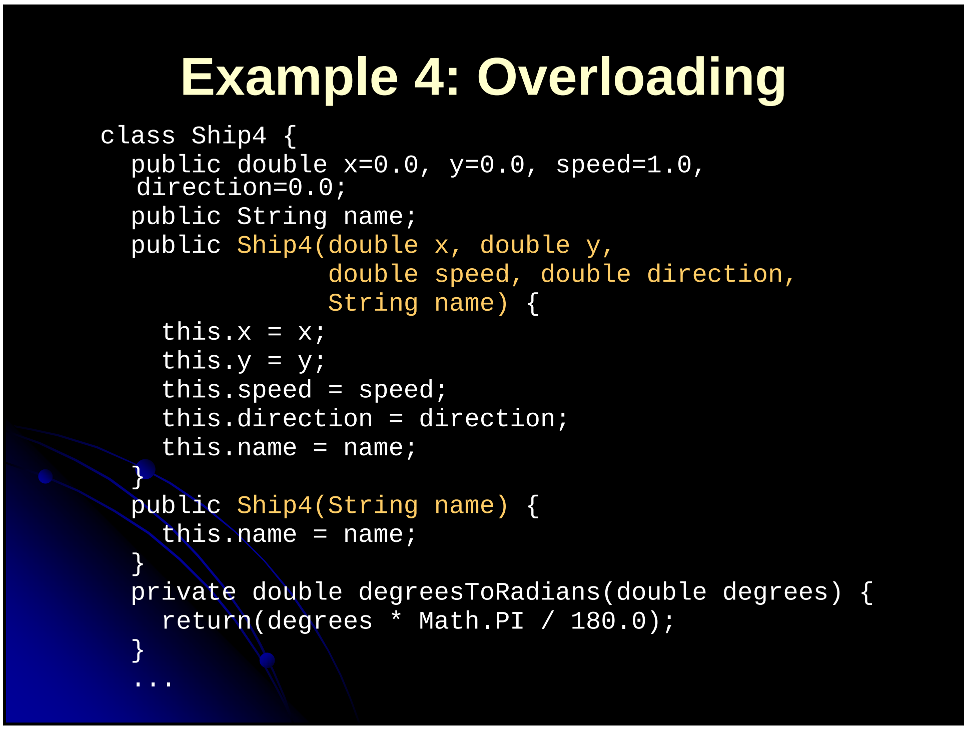
Summary

- Class names should start with upper case; method names with lower case
- Methods must define a return type or `void` if no result is returned
- Access fields via objectName.fieldName
- Access methods via objectName.methodName(args)
- If a method accepts no arguments, the arg-list in the method declaration is empty instead of `void` as in C
- Static methods do not require an instance of the class; they can be accessed through the class name
- The `this` reference refers to the *current* object
- Class constructors do not declare a return type
- Java performs its own memory management and requires no destructors



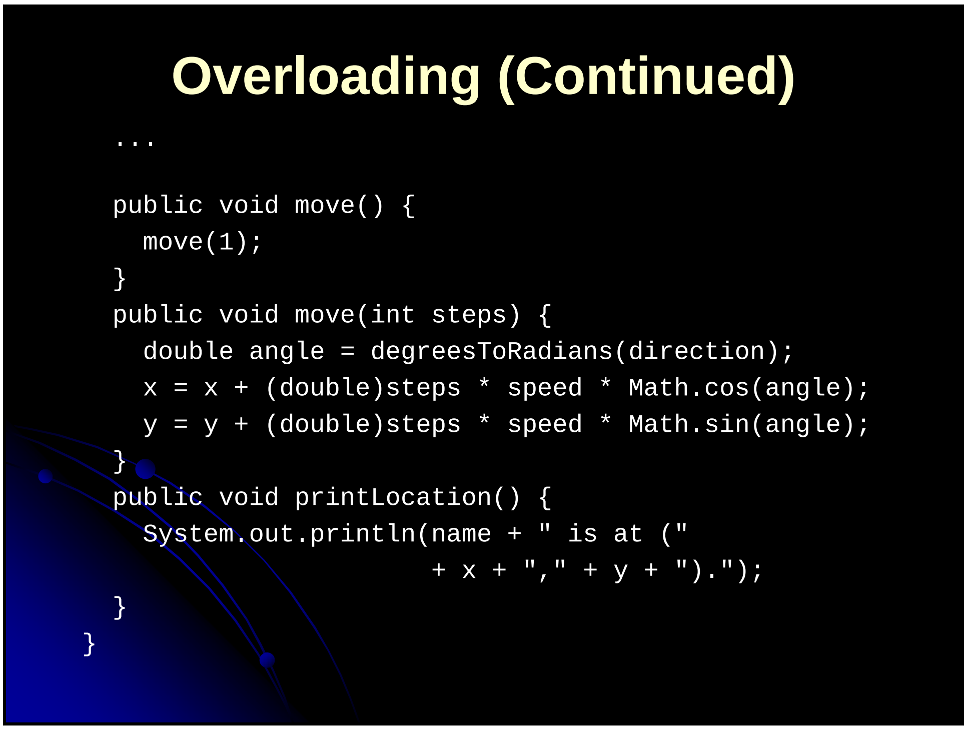
Agenda

- Overloading
- Designing “real” classes
- Inheritance
- Advanced topics
  - Abstract classes
  - Interfaces
  - Understanding polymorphism
  - Setting a CLASSPATH and using packages
  - Visibility modifiers
  - Creating on-line documentation using JavaDoc



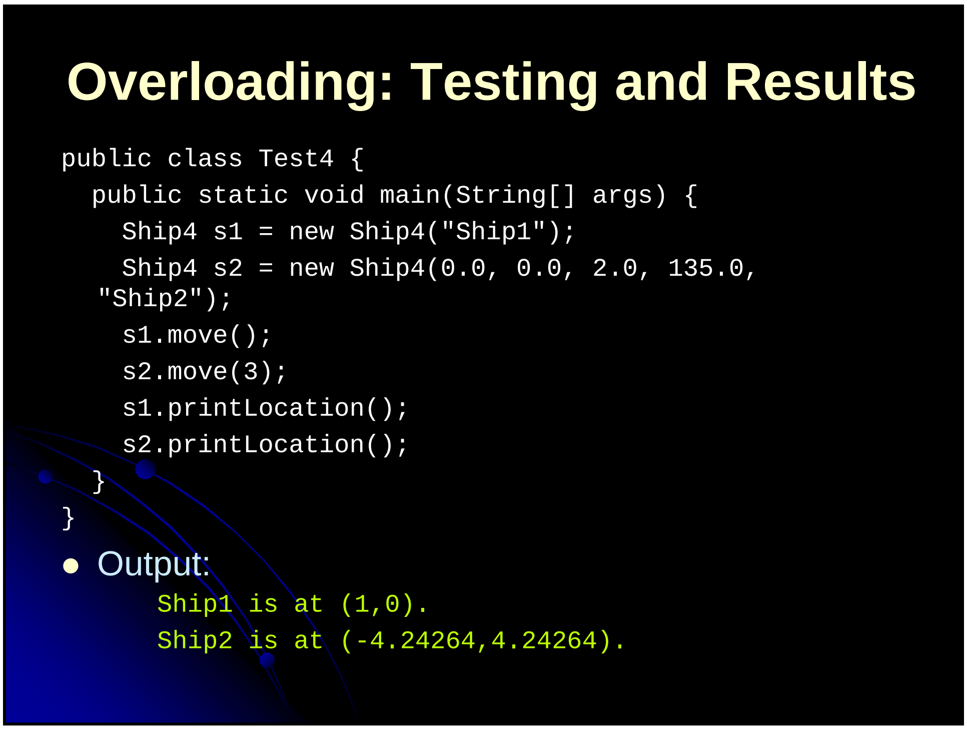
Example 4: Overloading

```
class Ship4 {
  public double x=0.0, y=0.0, speed=1.0,
  direction=0.0;
  public String name;
  public Ship4(double x, double y,
               double speed, double direction,
               String name) {
    this.x = x;
    this.y = y;
    this.speed = speed;
    this.direction = direction;
    this.name = name;
  }
  public Ship4(String name) {
    this.name = name;
  }
  private double degreesToRadians(double degrees) {
    return(degrees * Math.PI / 180.0);
  }
  ...
```



Overloading (Continued)

```
    ...

    public void move() {
      move(1);
    }
    public void move(int steps) {
      double angle = degreesToRadians(direction);
      x = x + (double)steps * speed * Math.cos(angle);
      y = y + (double)steps * speed * Math.sin(angle);
    }
    public void printLocation() {
      System.out.println(name + " is at ("
                         + x + "," + y + ").");
    }
  }
```

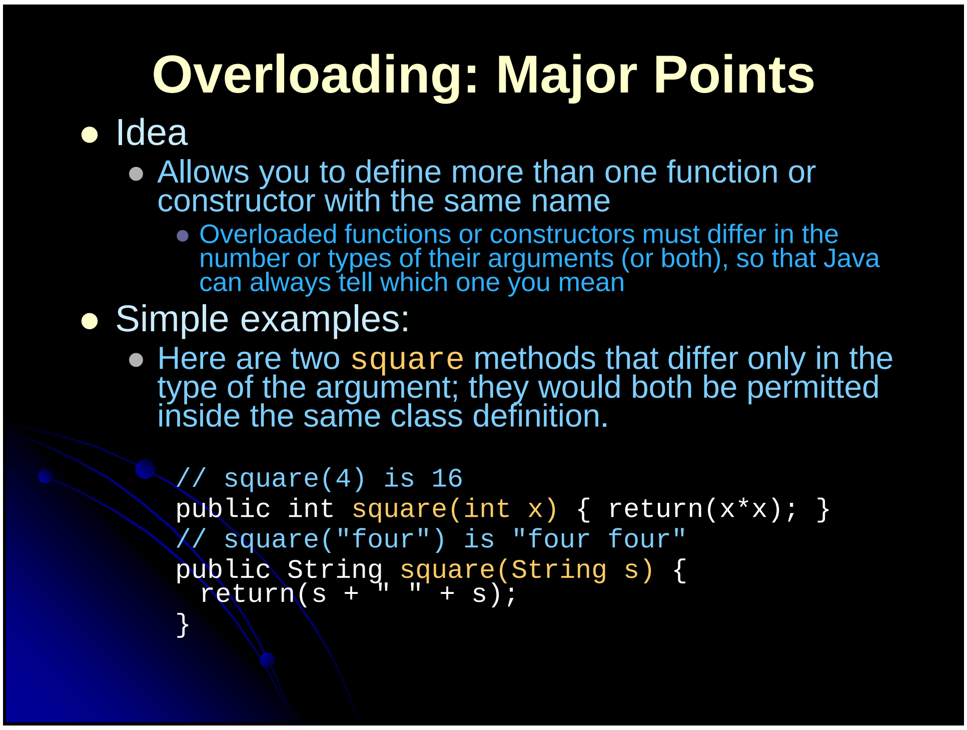


Overloading: Testing and Results

```
public class Test4 {
  public static void main(String[] args) {
    Ship4 s1 = new Ship4("Ship1");
    Ship4 s2 = new Ship4(0.0, 0.0, 2.0, 135.0,
   "Ship2");
    s1.move();
    s2.move(3);
    s1.printLocation();
    s2.printLocation();
  }
}
```

- Output:

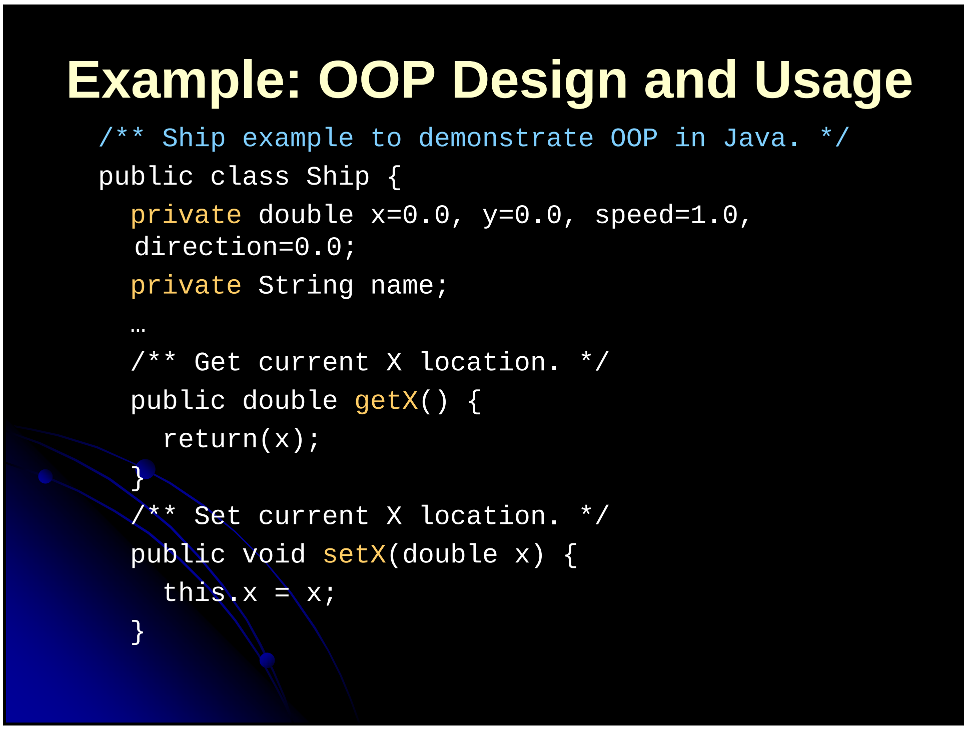
```
Ship1 is at (1,0).
Ship2 is at (-4.24264,4.24264).
```



Overloading: Major Points

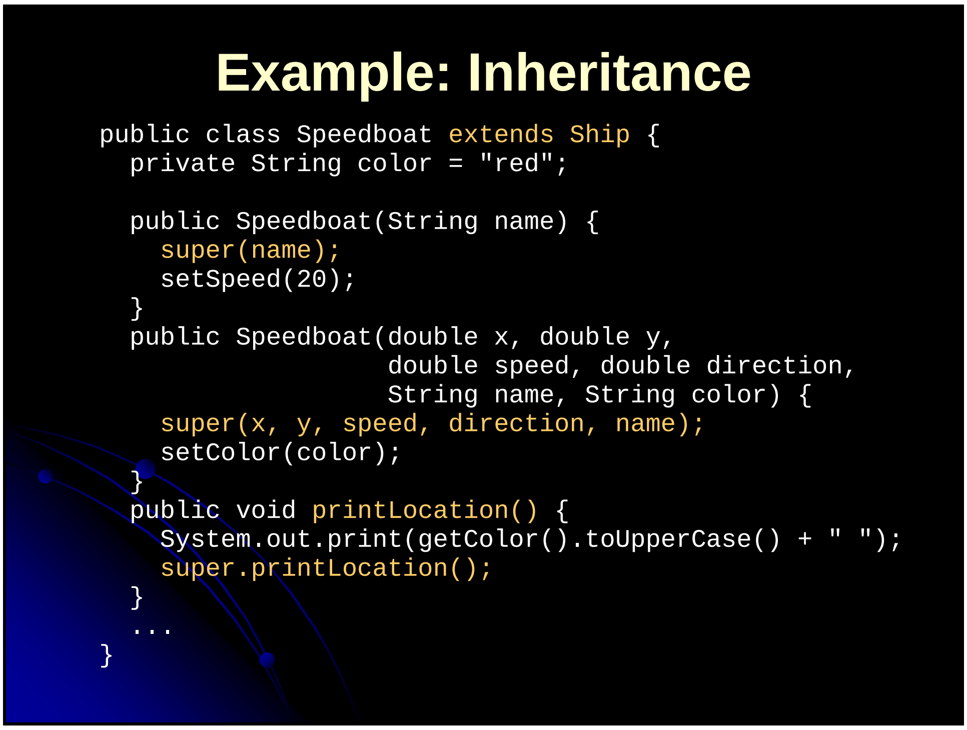
- Idea
  - Allows you to define more than one function or constructor with the same name
    - Overloaded functions or constructors must differ in the number or types of their arguments (or both), so that Java can always tell which one you mean
- Simple examples:
  - Here are two `square` methods that differ only in the type of the argument; they would both be permitted inside the same class definition.

```
// square(4) is 16
public int square(int x) { return(x*x); }
// square("four") is "four four"
public String square(String s) {
  return(s + " " + s);
}
```



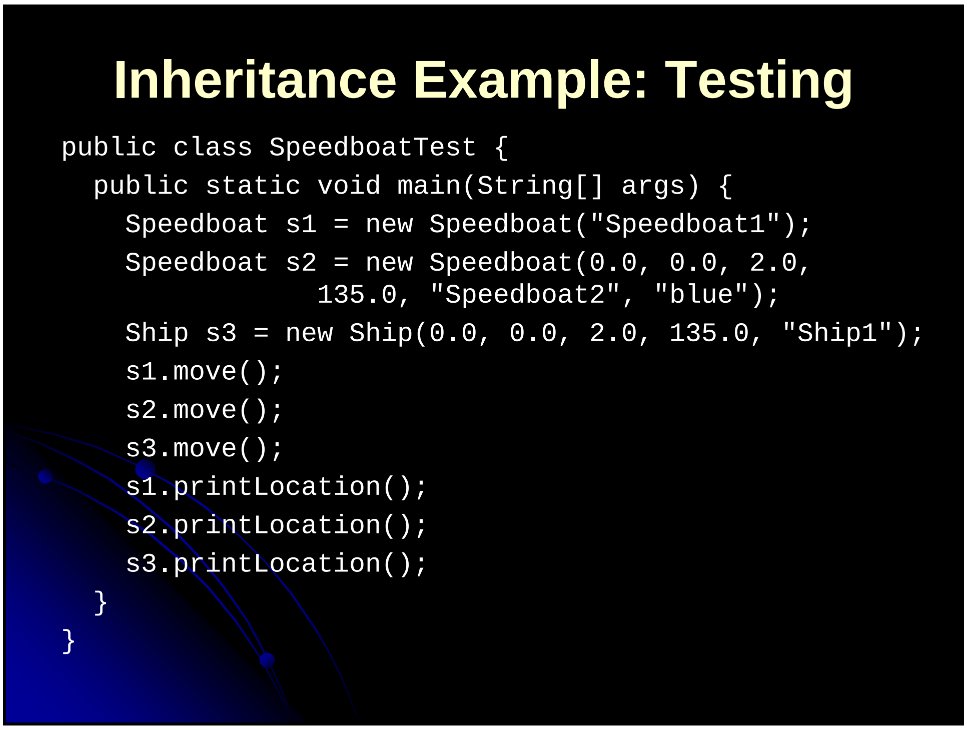
Example: OOP Design and Usage

```
/** Ship example to demonstrate OOP in Java. */
public class Ship {
  private double x=0.0, y=0.0, speed=1.0,
  direction=0.0;
  private String name;
  …
  /** Get current X location. */
  public double getX() {
    return(x);
  }
  /** Set current X location. */
  public void setX(double x) {
    this.x = x;
  }
```



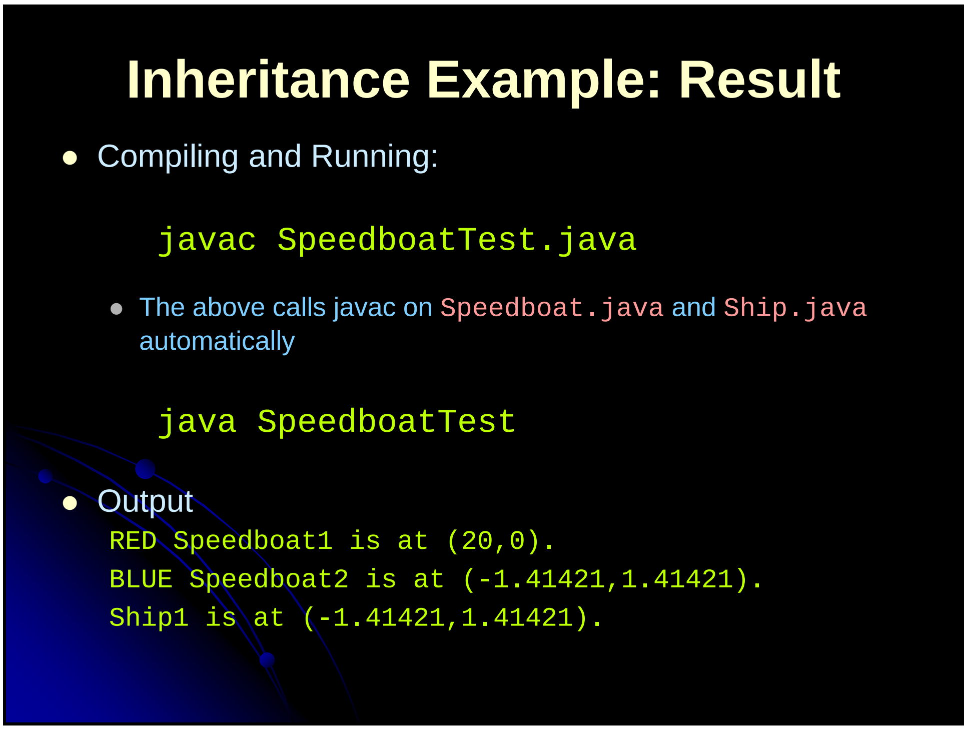
Example: Inheritance

```
public class Speedboat extends Ship {
  private String color = "red";

  public Speedboat(String name) {
    super(name);
    setSpeed(20);
  }
  public Speedboat(double x, double y,
                   double speed, double direction,
                   String name, String color) {
    super(x, y, speed, direction, name);
    setColor(color);
  }
  public void printLocation() {
    System.out.print(getColor().toUpperCase() + " ");
    super.printLocation();
  }
  ...
}
```



Inheritance Example: Testing

```
public class SpeedboatTest {
  public static void main(String[] args) {
    Speedboat s1 = new Speedboat("Speedboat1");
    Speedboat s2 = new Speedboat(0.0, 0.0, 2.0,
                135.0, "Speedboat2", "blue");
    Ship s3 = new Ship(0.0, 0.0, 2.0, 135.0, "Ship1");
    s1.move();
    s2.move();
    s3.move();
    s1.printLocation();
    s2.printLocation();
    s3.printLocation();
  }
}
```



Inheritance Example: Result

- Compiling and Running:

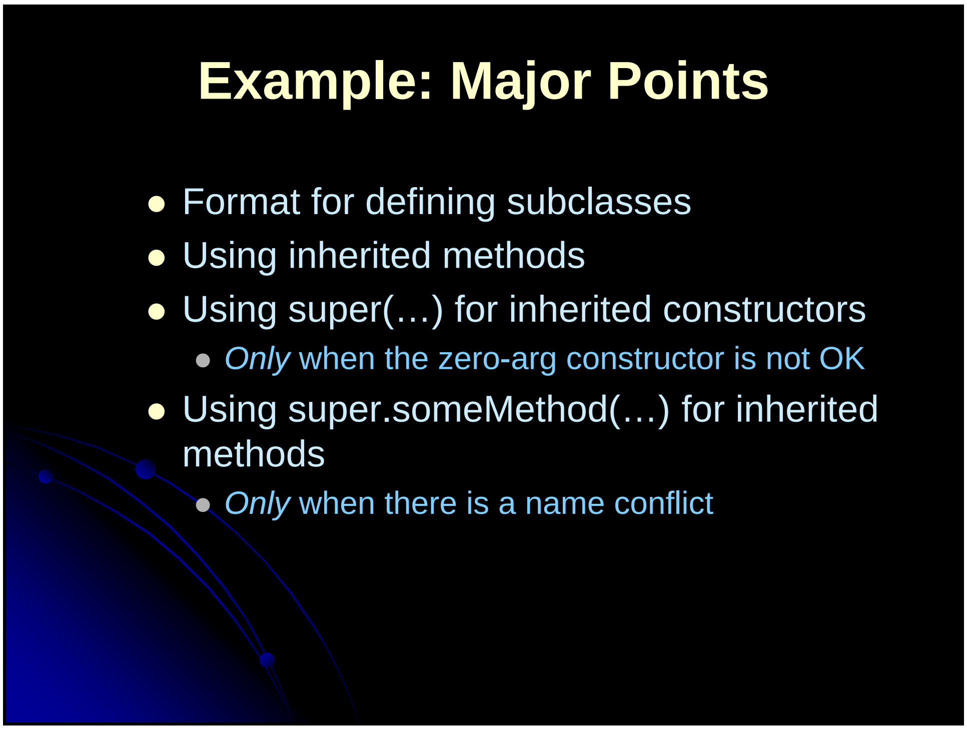
```
javac SpeedboatTest.java
```

  - The above calls javac on `Speedboat.java` and `Ship.java` automatically

```
java SpeedboatTest
```

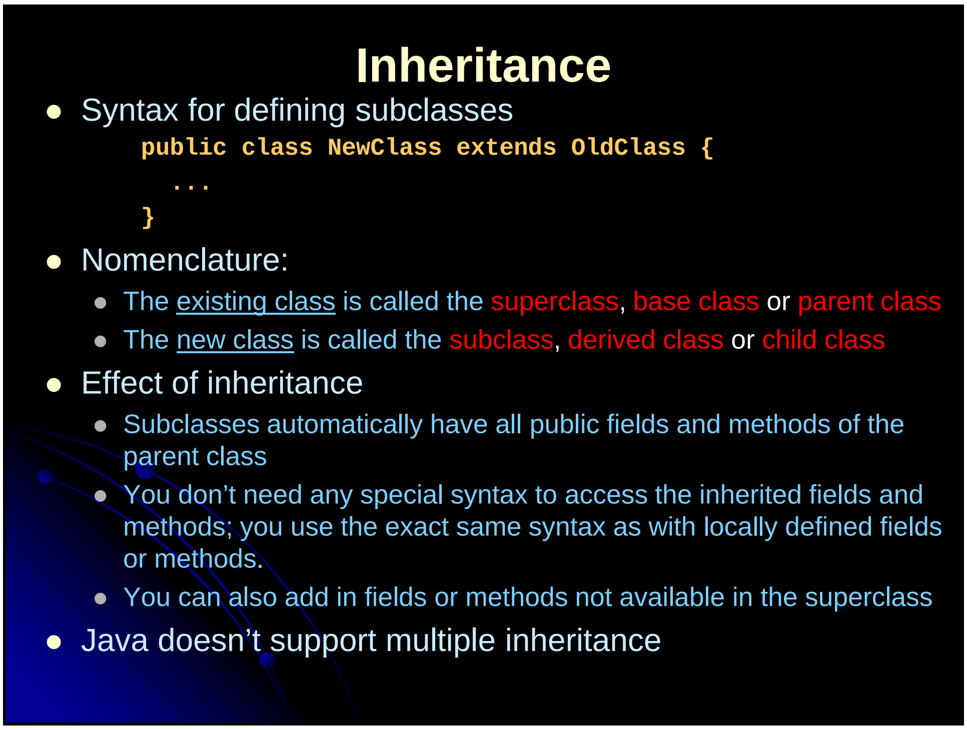
- Output

```
RED Speedboat1 is at (20,0).
BLUE Speedboat2 is at (-1.41421,1.41421).
Ship1 is at (-1.41421,1.41421).
```



Example: Major Points

- Format for defining subclasses
- Using inherited methods
- Using super(…) for inherited constructors
  - *Only* when the zero-arg constructor is not OK
- Using super.someMethod(…) for inherited methods
  - *Only* when there is a name conflict

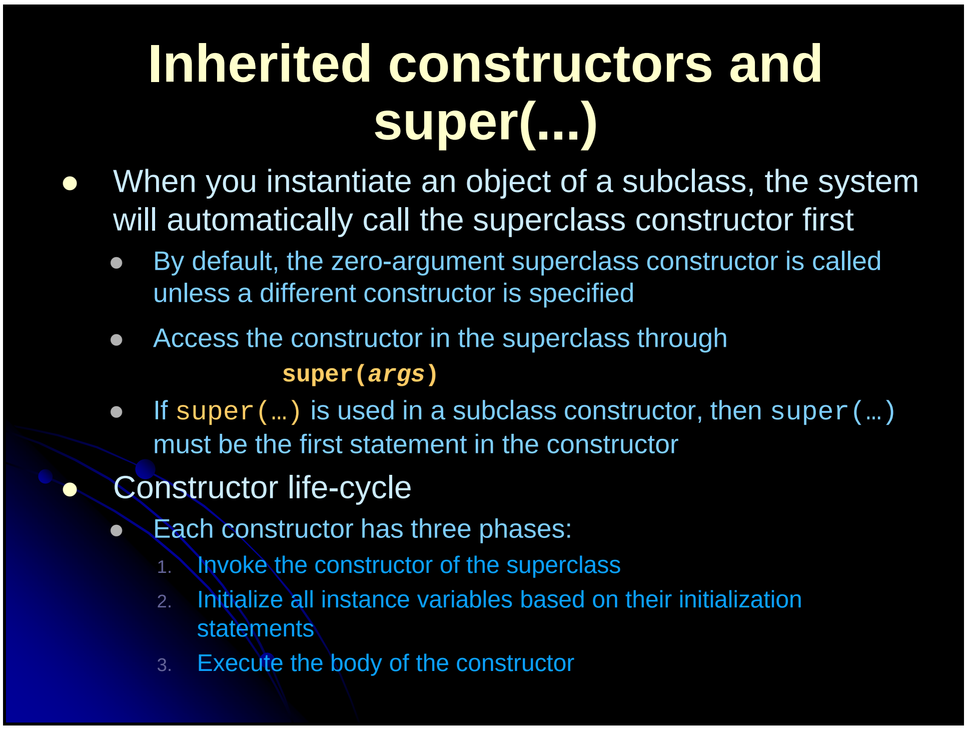


Inheritance

- Syntax for defining subclasses

```
public class NewClass extends OldClass {
  ...
}
```

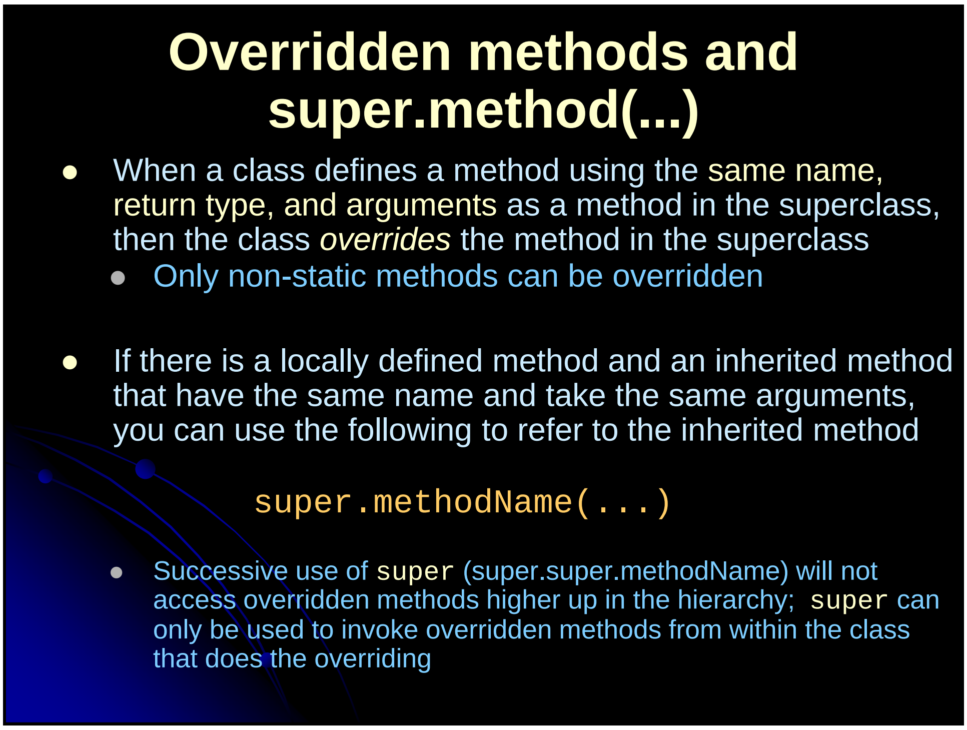
- Nomenclature:
  - The existing class is called the superclass, base class or parent class
  - The new class is called the subclass, derived class or child class
- Effect of inheritance
  - Subclasses automatically have all public fields and methods of the parent class
  - You don't need any special syntax to access the inherited fields and methods; you use the exact same syntax as with locally defined fields or methods.
  - You can also add in fields or methods not available in the superclass
- Java doesn't support multiple inheritance



Inherited constructors and super(...)

- When you instantiate an object of a subclass, the system will automatically call the superclass constructor first
  - By default, the zero-argument superclass constructor is called unless a different constructor is specified
  - Access the constructor in the superclass through

    `super(`***`args`***`)`
  - If `super(…)` is used in a subclass constructor, then `super(…)` must be the first statement in the constructor
- Constructor life-cycle
  - Each constructor has three phases:
    1. Invoke the constructor of the superclass
    2. Initialize all instance variables based on their initialization statements
    3. Execute the body of the constructor



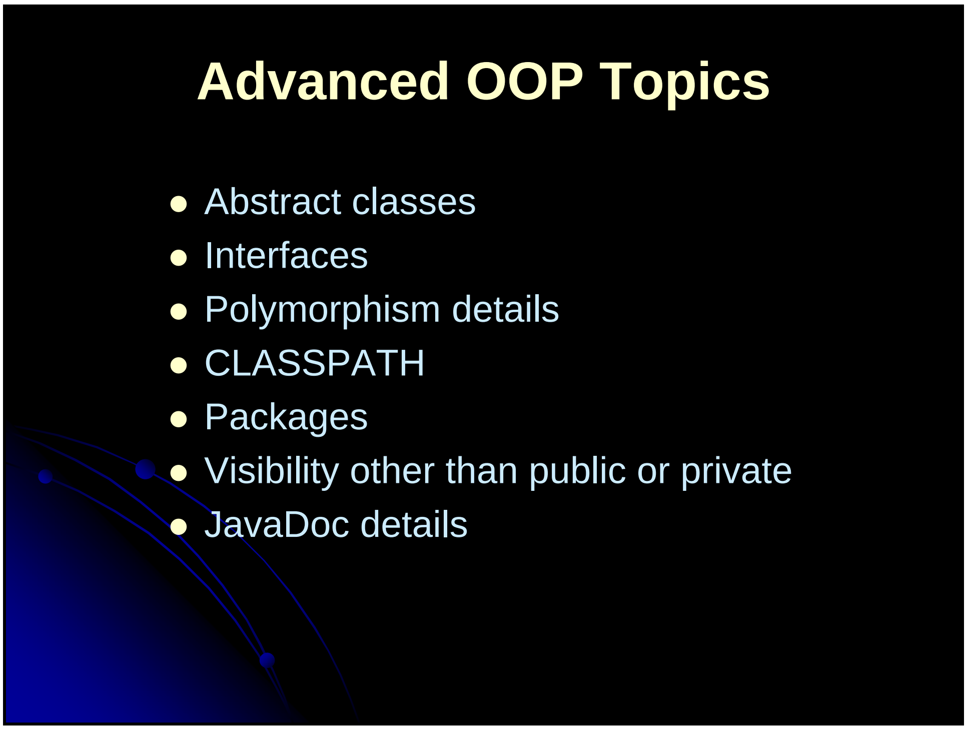
Overridden methods and super.method(...)

- When a class defines a method using the same name, return type, and arguments as a method in the superclass, then the class *overrides* the method in the superclass
  - Only non-static methods can be overridden

- If there is a locally defined method and an inherited method that have the same name and take the same arguments, you can use the following to refer to the inherited method

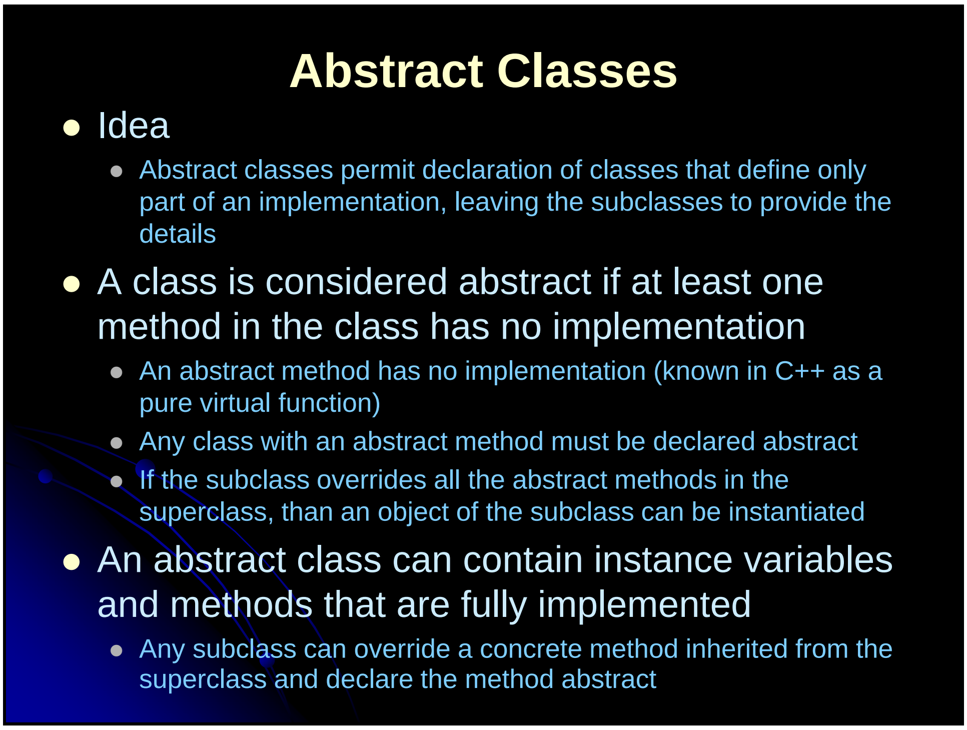
```
super.methodName(...)
```

  - Successive use of `super` (super.super.methodName) will not access overridden methods higher up in the hierarchy; `super` can only be used to invoke overridden methods from within the class that does the overriding



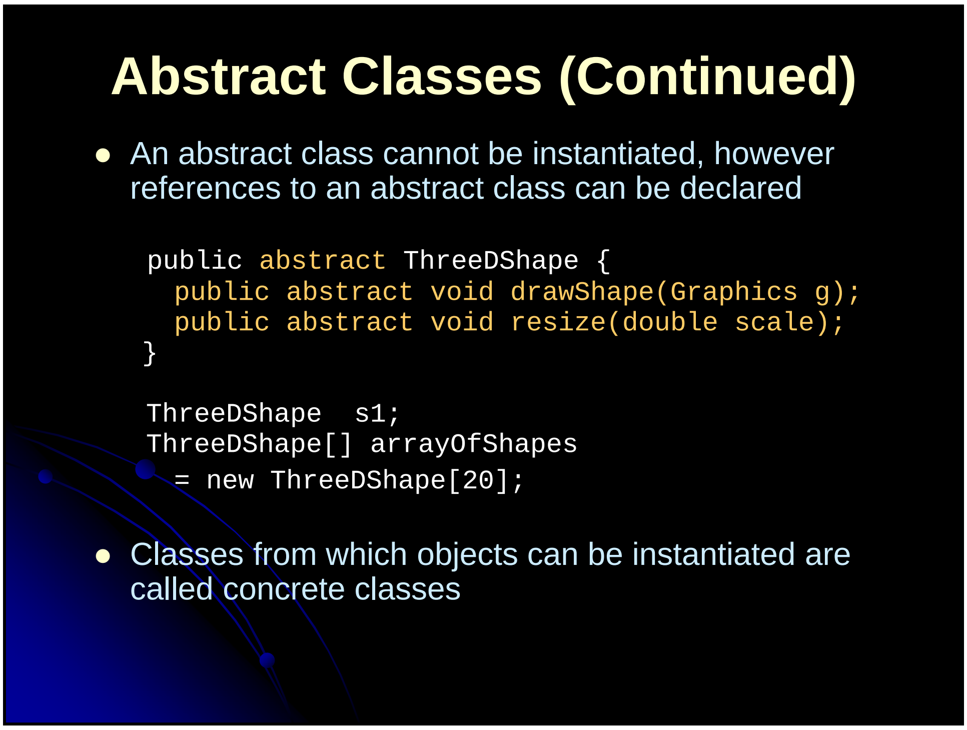
Advanced OOP Topics

- Abstract classes
- Interfaces
- Polymorphism details
- CLASSPATH
- Packages
- Visibility other than public or private
- JavaDoc details



Abstract Classes

- Idea
  - Abstract classes permit declaration of classes that define only part of an implementation, leaving the subclasses to provide the details
- A class is considered abstract if at least one method in the class has no implementation
  - An abstract method has no implementation (known in C++ as a pure virtual function)
  - Any class with an abstract method must be declared abstract
  - If the subclass overrides all the abstract methods in the superclass, than an object of the subclass can be instantiated
- An abstract class can contain instance variables and methods that are fully implemented
  - Any subclass can override a concrete method inherited from the superclass and declare the method abstract

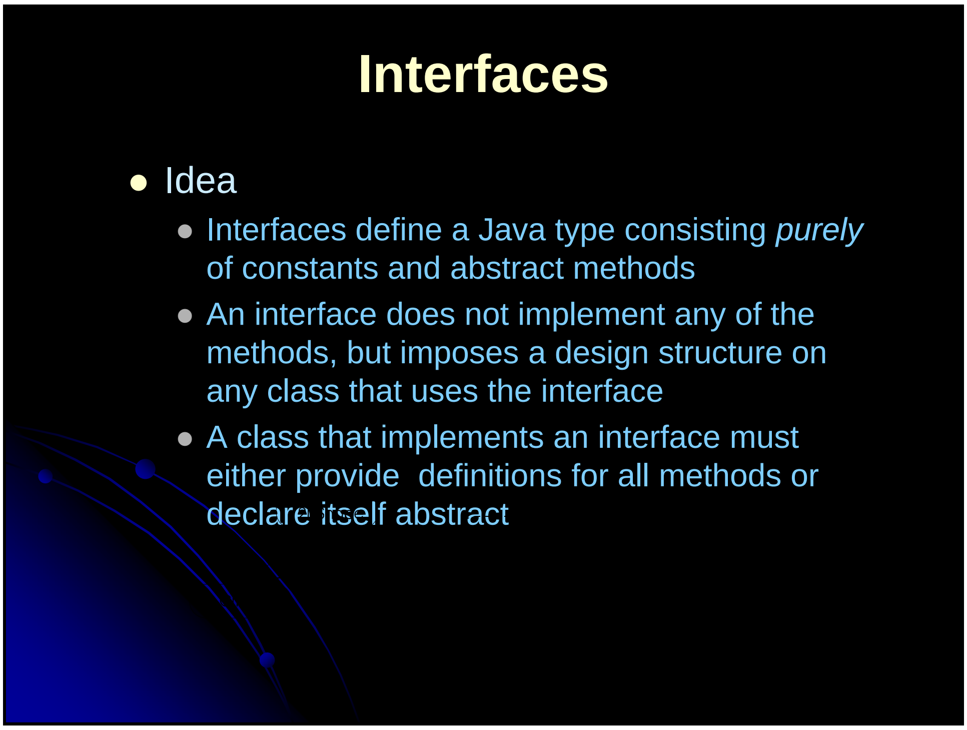


Abstract Classes (Continued)

- An abstract class cannot be instantiated, however references to an abstract class can be declared

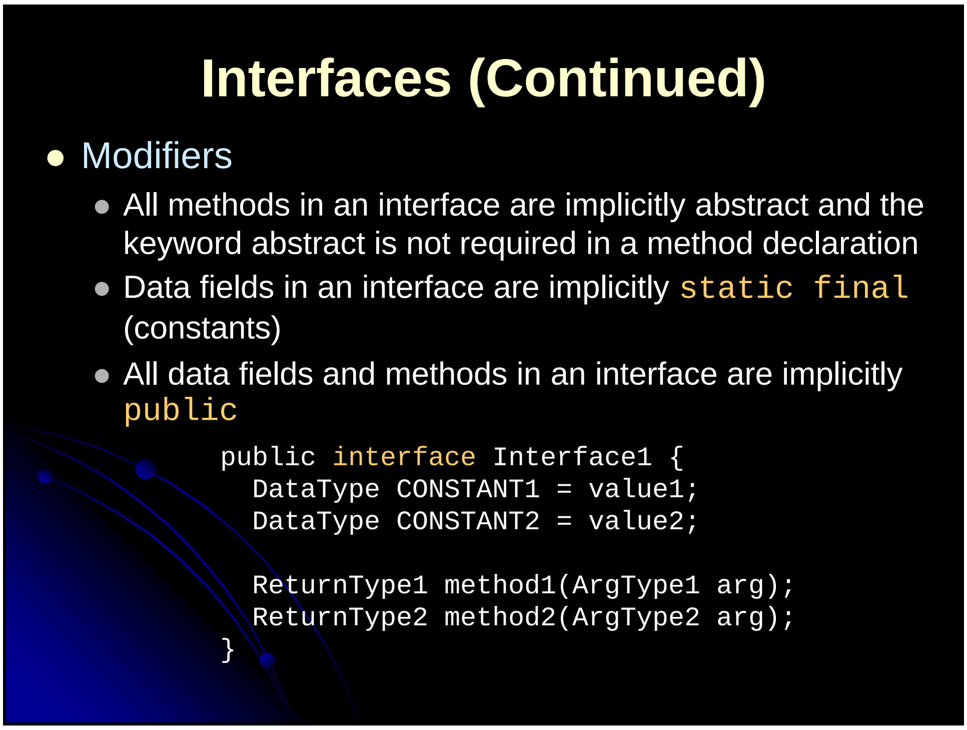
```
public abstract ThreeDShape {
  public abstract void drawShape(Graphics g);
  public abstract void resize(double scale);
}

ThreeDShape  s1;
ThreeDShape[] arrayOfShapes
  = new ThreeDShape[20];
```

- Classes from which objects can be instantiated are called concrete classes



Interfaces

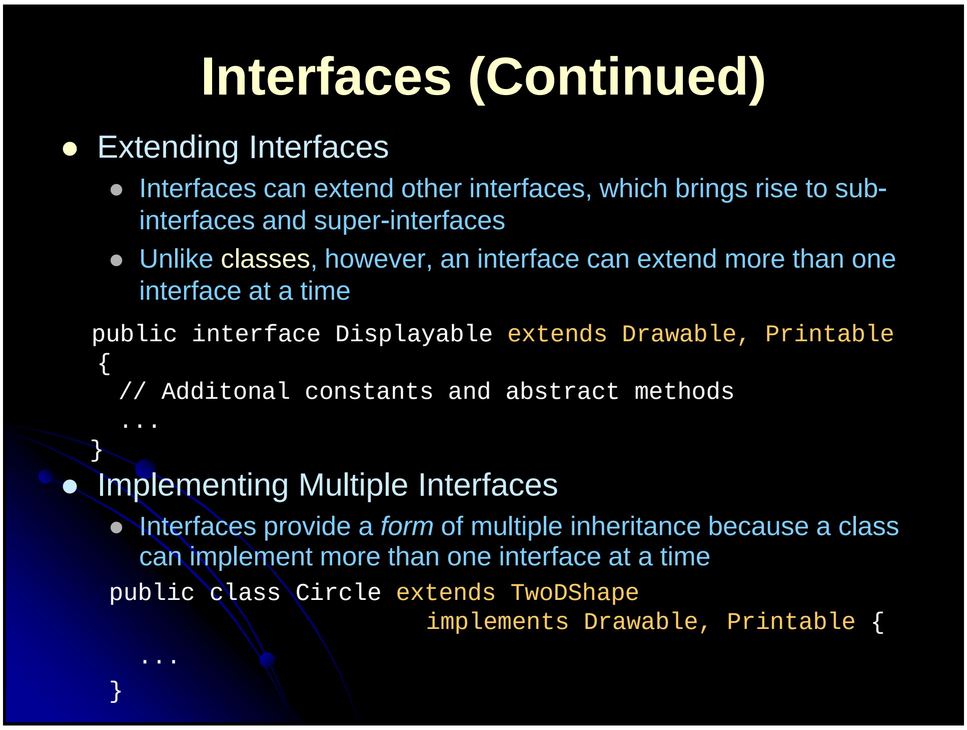
- Idea
  - Interfaces define a Java type consisting *purely* of constants and abstract methods
  - An interface does not implement any of the methods, but imposes a design structure on any class that uses the interface
  - A class that implements an interface must either provide definitions for all methods or declare itself abstract



Interfaces (Continued)

- Modifiers
  - All methods in an interface are implicitly abstract and the keyword abstract is not required in a method declaration
  - Data fields in an interface are implicitly `static final` (constants)
  - All data fields and methods in an interface are implicitly `public`

```
public interface Interface1 {
  DataType CONSTANT1 = value1;
  DataType CONSTANT2 = value2;

  ReturnType1 method1(ArgType1 arg);
  ReturnType2 method2(ArgType2 arg);
}
```



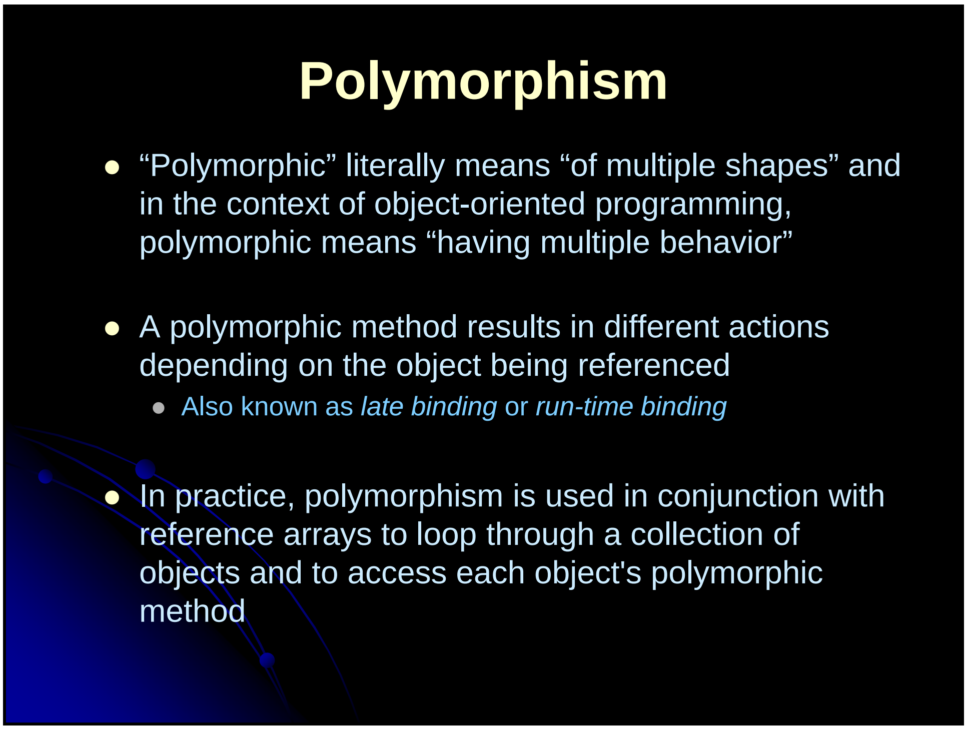
Interfaces (Continued)

- Extending Interfaces
  - Interfaces can extend other interfaces, which brings rise to sub-interfaces and super-interfaces
  - Unlike classes, however, an interface can extend more than one interface at a time

```
public interface Displayable extends Drawable, Printable
 {
   // Additonal constants and abstract methods
   ...
}
```

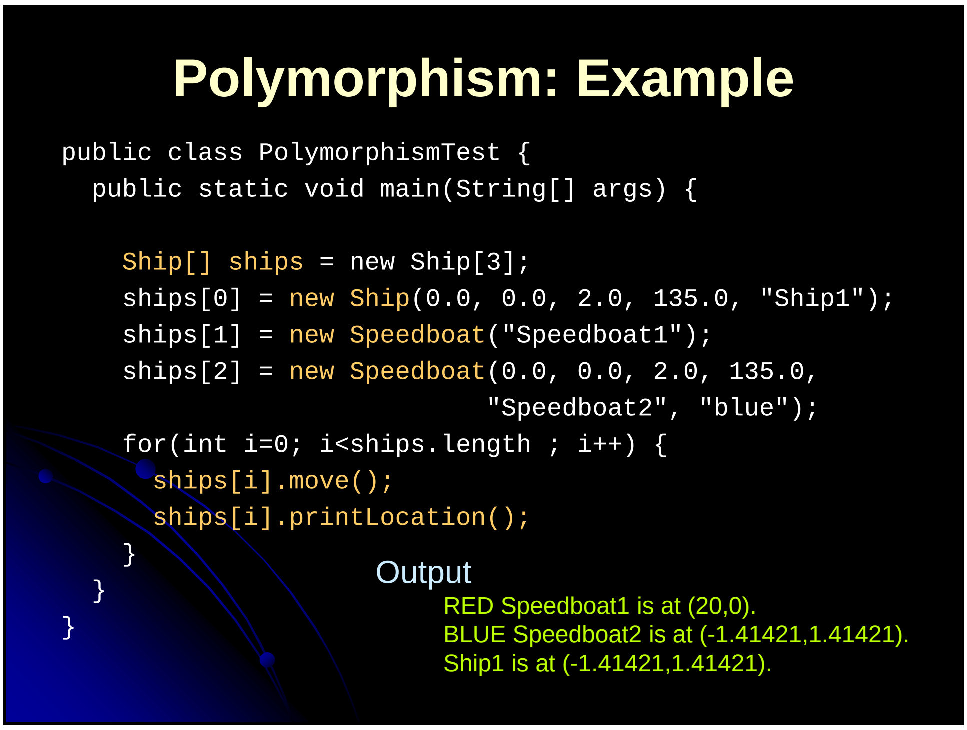
- Implementing Multiple Interfaces
  - Interfaces provide a *form* of multiple inheritance because a class can implement more than one interface at a time

```
public class Circle extends TwoDShape
                      implements Drawable, Printable {
    ...
}
```



Polymorphism

- “Polymorphic” literally means “of multiple shapes” and in the context of object-oriented programming, polymorphic means “having multiple behavior”
- A polymorphic method results in different actions depending on the object being referenced
  - Also known as *late binding* or *run-time binding*
- In practice, polymorphism is used in conjunction with reference arrays to loop through a collection of objects and to access each object's polymorphic method

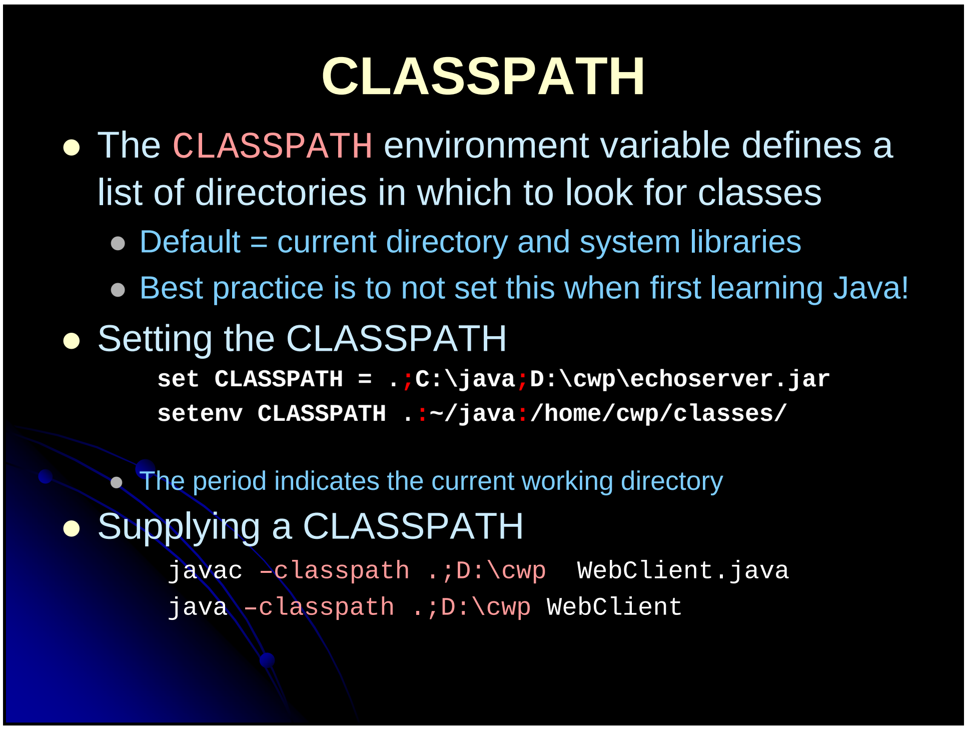


Polymorphism: Example

```
public class PolymorphismTest {
  public static void main(String[] args) {

    Ship[] ships = new Ship[3];
    ships[0] = new Ship(0.0, 0.0, 2.0, 135.0, "Ship1");
    ships[1] = new Speedboat("Speedboat1");
    ships[2] = new Speedboat(0.0, 0.0, 2.0, 135.0,
                             "Speedboat2", "blue");
    for(int i=0; i<ships.length ; i++) {
      ships[i].move();
      ships[i].printLocation();
    }
  }
}
```

Output

```
RED Speedboat1 is at (20,0).
BLUE Speedboat2 is at (-1.41421,1.41421).
Ship1 is at (-1.41421,1.41421).
```



CLASSPATH

- The CLASSPATH environment variable defines a list of directories in which to look for classes
  - Default = current directory and system libraries
  - Best practice is to not set this when first learning Java!
- Setting the CLASSPATH

```
set CLASSPATH = .;C:\java;D:\cwp\echoserver.jar
setenv CLASSPATH .:~/java:/home/cwp/classes/
```

  - The period indicates the current working directory
- Supplying a CLASSPATH

```
javac –classpath .;D:\cwp  WebClient.java
java –classpath .;D:\cwp WebClient
```

Creating Packages

- A package lets you group classes in subdirectories to avoid accidental name conflicts
  - To create a package:
    1. Create a subdirectory with the same name as the desired package and place the source files in that directory
    2. Add a package statement to each file

```
package packagename;
```

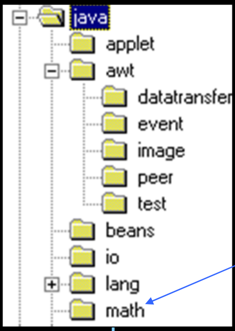
    3. Files in the main directory that want to use the package should include

```
import packagename.*;
```

- The package statement must be the first statement in the file
- If a package statement is omitted from a file, then the code is part of the default package that has no name

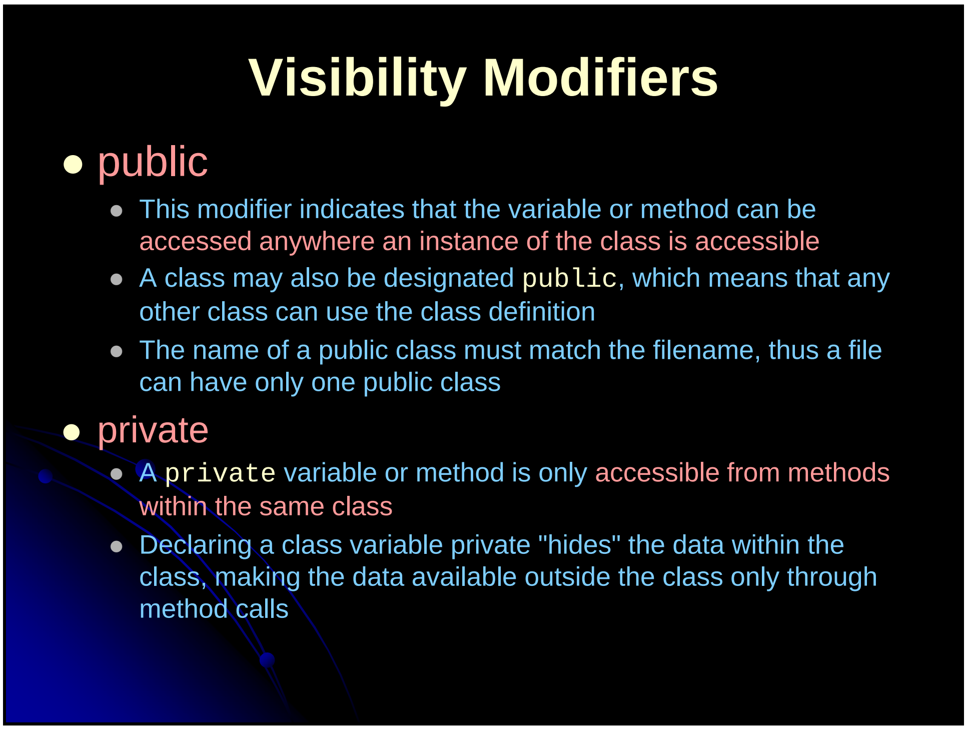
Package Directories

- The package hierarchy reflects the file system directory structure



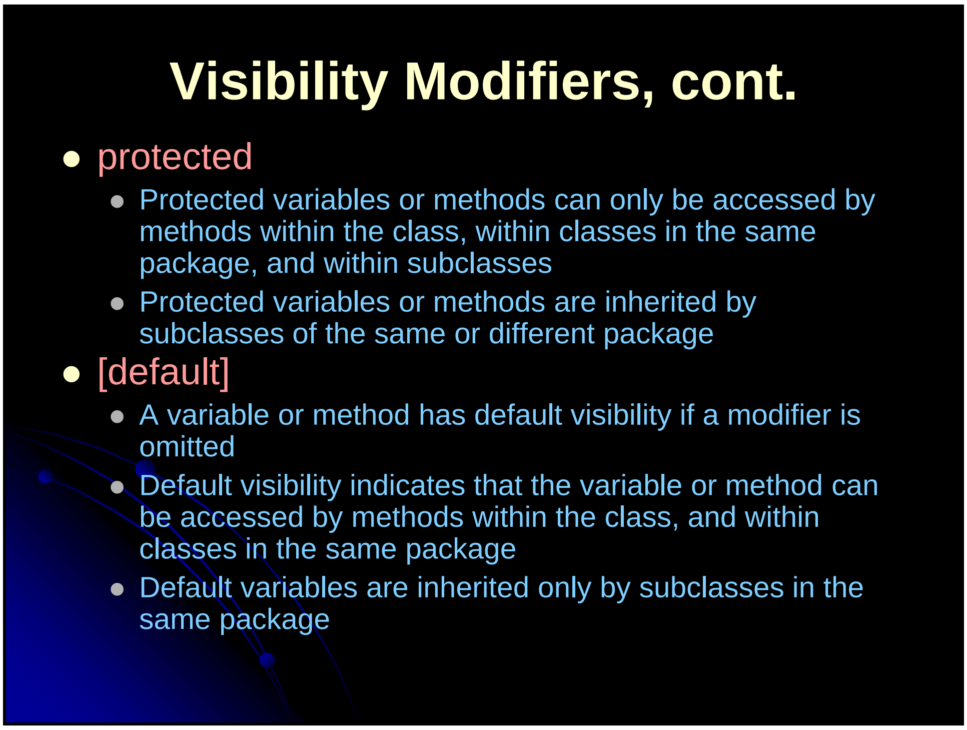
Package `java.math`

  - The root of any package must be accessible through a Java system default directory or through the `CLASSPATH` environment variable



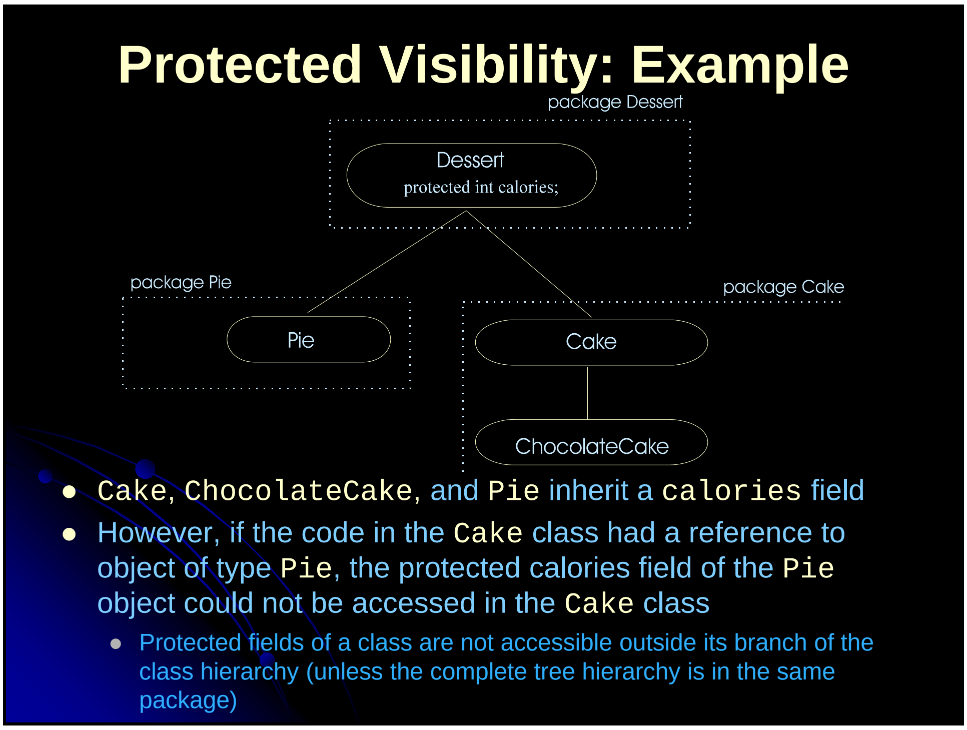
Visibility Modifiers

- public
  - This modifier indicates that the variable or method can be accessed anywhere an instance of the class is accessible
  - A class may also be designated `public`, which means that any other class can use the class definition
  - The name of a public class must match the filename, thus a file can have only one public class
- private
  - A `private` variable or method is only accessible from methods within the same class
  - Declaring a class variable private "hides" the data within the class, making the data available outside the class only through method calls



Visibility Modifiers, cont.

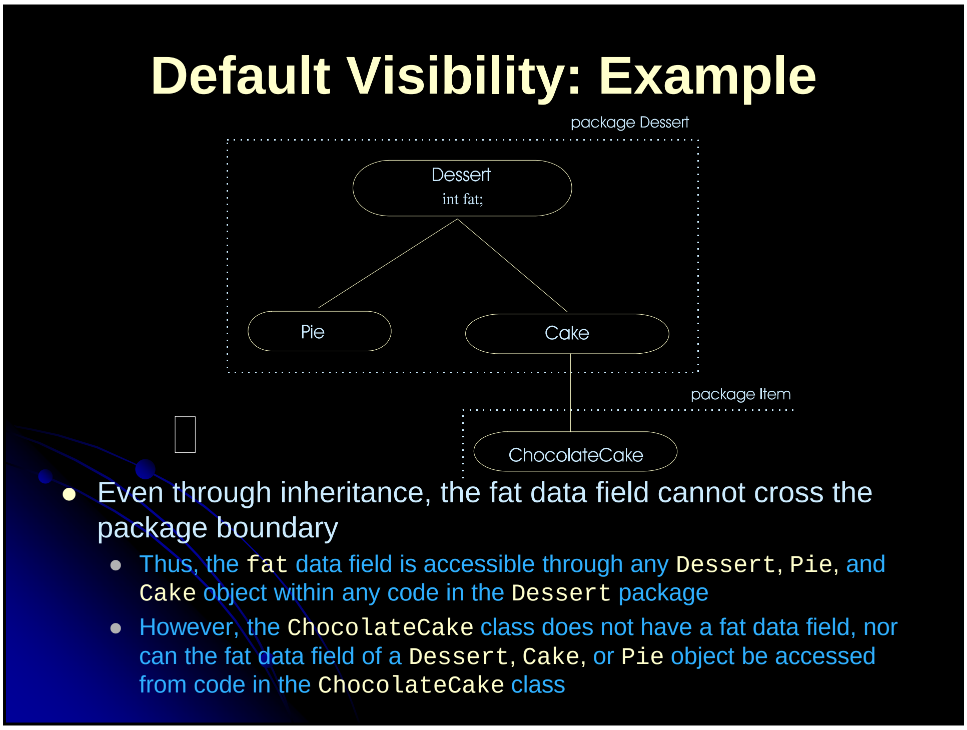
- protected
  - Protected variables or methods can only be accessed by methods within the class, within classes in the same package, and within subclasses
  - Protected variables or methods are inherited by subclasses of the same or different package
- [default]
  - A variable or method has default visibility if a modifier is omitted
  - Default visibility indicates that the variable or method can be accessed by methods within the class, and within classes in the same package
  - Default variables are inherited only by subclasses in the same package



Protected Visibility: Example

package Dessert
Dessert
protected int calories;
package Pie
Pie
package Cake
Cake
ChocolateCake

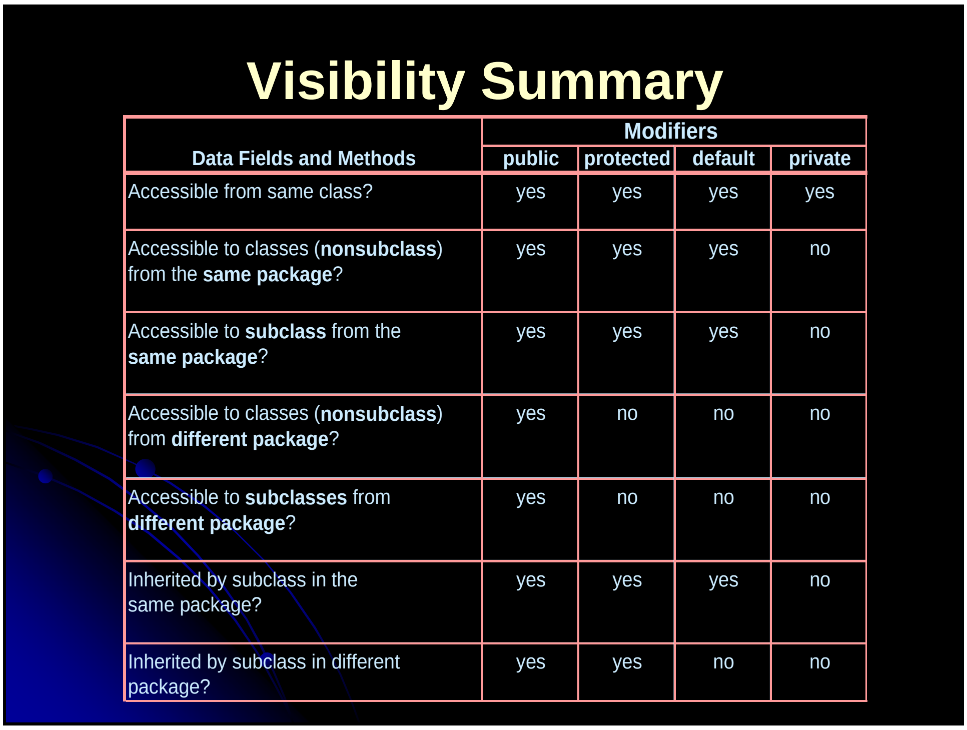
- `Cake`, `ChocolateCake`, and `Pie` inherit a `calories` field
- However, if the code in the `Cake` class had a reference to object of type `Pie`, the protected calories field of the `Pie` object could not be accessed in the `Cake` class
  - Protected fields of a class are not accessible outside its branch of the class hierarchy (unless the complete tree hierarchy is in the same package)



Default Visibility: Example

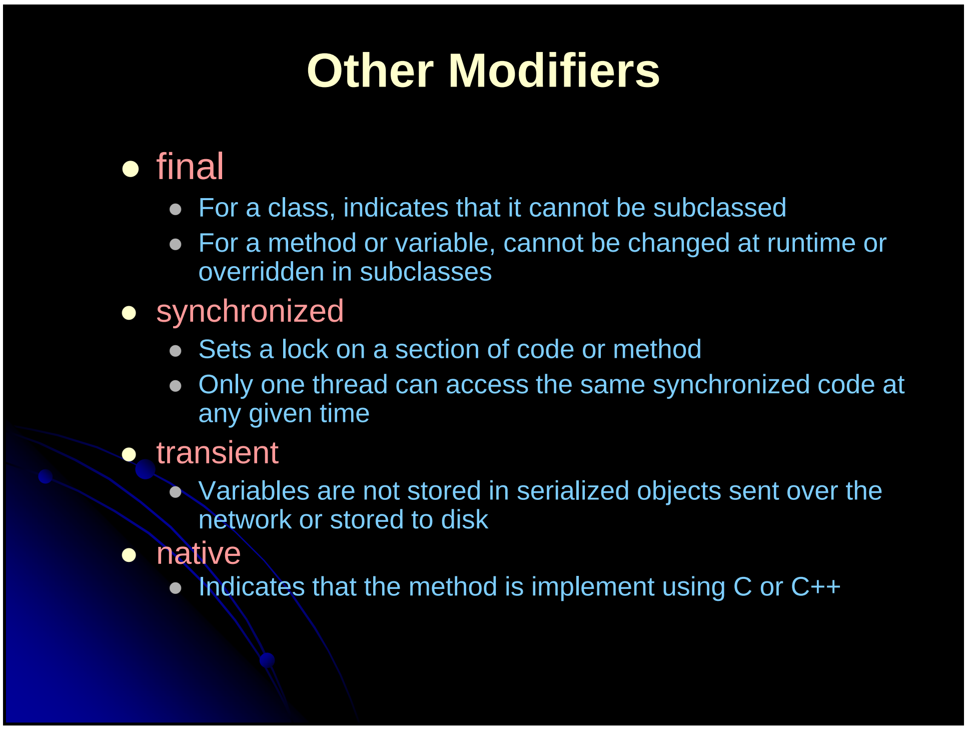
package Dessert
Dessert
int fat;
Pie
Cake
package Item
ChocolateCake

- Even through inheritance, the fat data field cannot cross the package boundary
  - Thus, the `fat` data field is accessible through any `Dessert`, `Pie`, and `Cake` object within any code in the `Dessert` package
  - However, the `ChocolateCake` class does not have a fat data field, nor can the fat data field of a `Dessert`, `Cake`, or `Pie` object be accessed from code in the `ChocolateCake` class



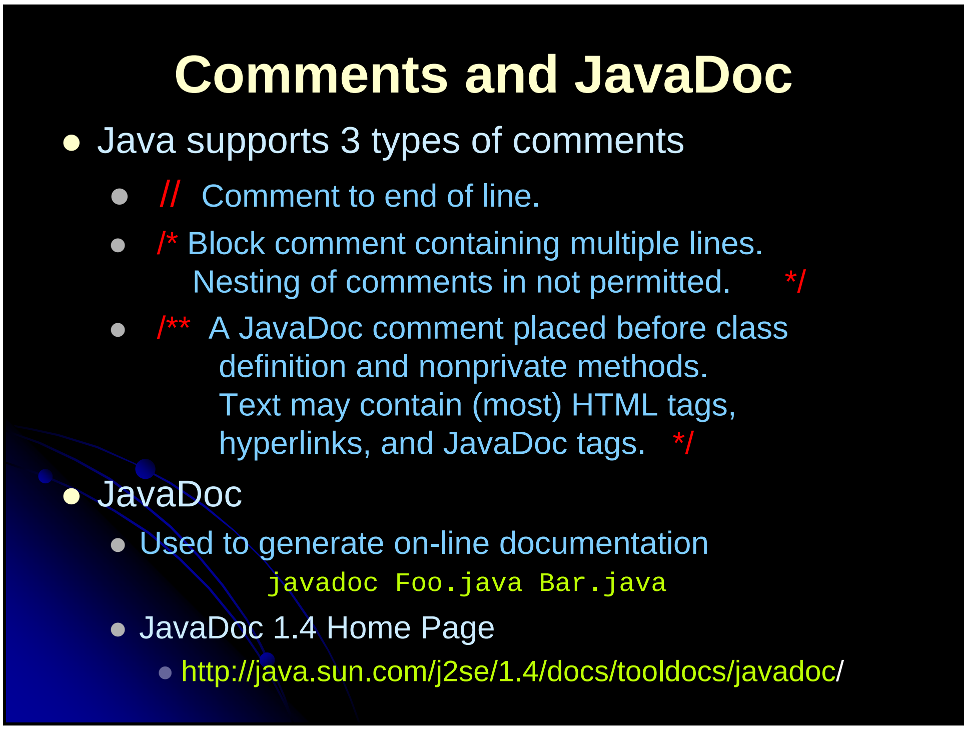
Visibility Summary

Data Fields and Methods	Modifiers			
	public	protected	default	private
Accessible from same class?	yes	yes	yes	yes
Accessible to classes (**nonsubclass**) from the **same package**?	yes	yes	yes	no
Accessible to **subclass** from the **same package**?	yes	yes	yes	no
Accessible to classes (**nonsubclass**) from **different package**?	yes	no	no	no
Accessible to **subclasses** from **different package**?	yes	no	no	no
Inherited by subclass in the same package?	yes	yes	yes	no
Inherited by subclass in different package?	yes	yes	no	no



Other Modifiers

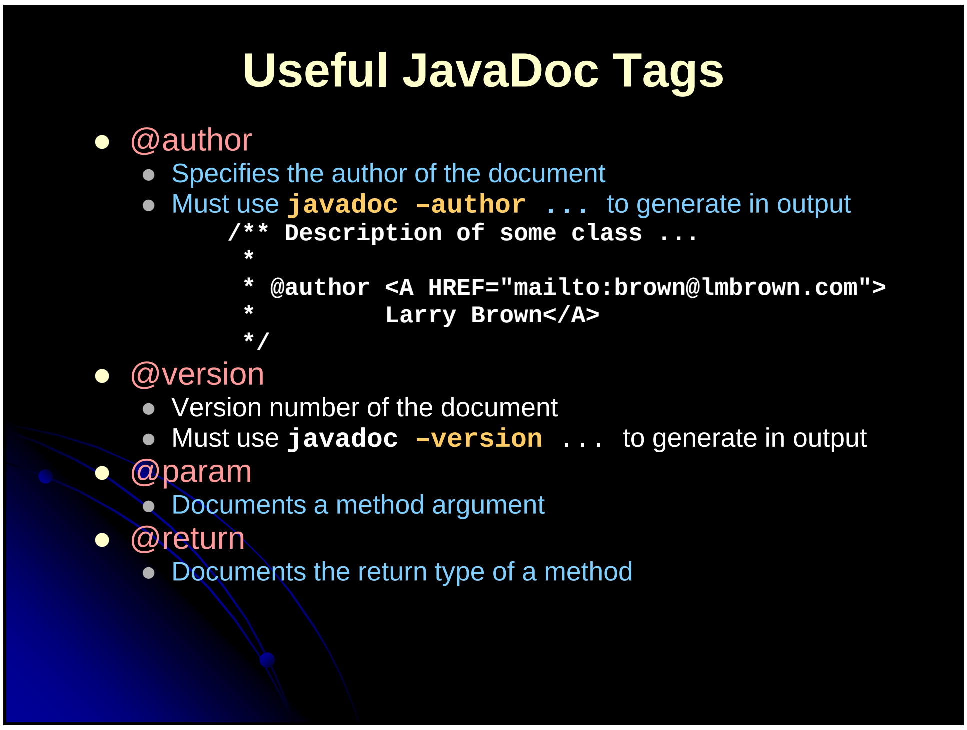
- final
  - For a class, indicates that it cannot be subclassed
  - For a method or variable, cannot be changed at runtime or overridden in subclasses
- synchronized
  - Sets a lock on a section of code or method
  - Only one thread can access the same synchronized code at any given time
- transient
  - Variables are not stored in serialized objects sent over the network or stored to disk
- native
  - Indicates that the method is implement using C or C++



Comments and JavaDoc

- Java supports 3 types of comments
  - // Comment to end of line.
  - /* Block comment containing multiple lines.
    Nesting of comments in not permitted. */
  - /** A JavaDoc comment placed before class
    definition and nonprivate methods.
    Text may contain (most) HTML tags,
    hyperlinks, and JavaDoc tags. */
- JavaDoc
  - Used to generate on-line documentation

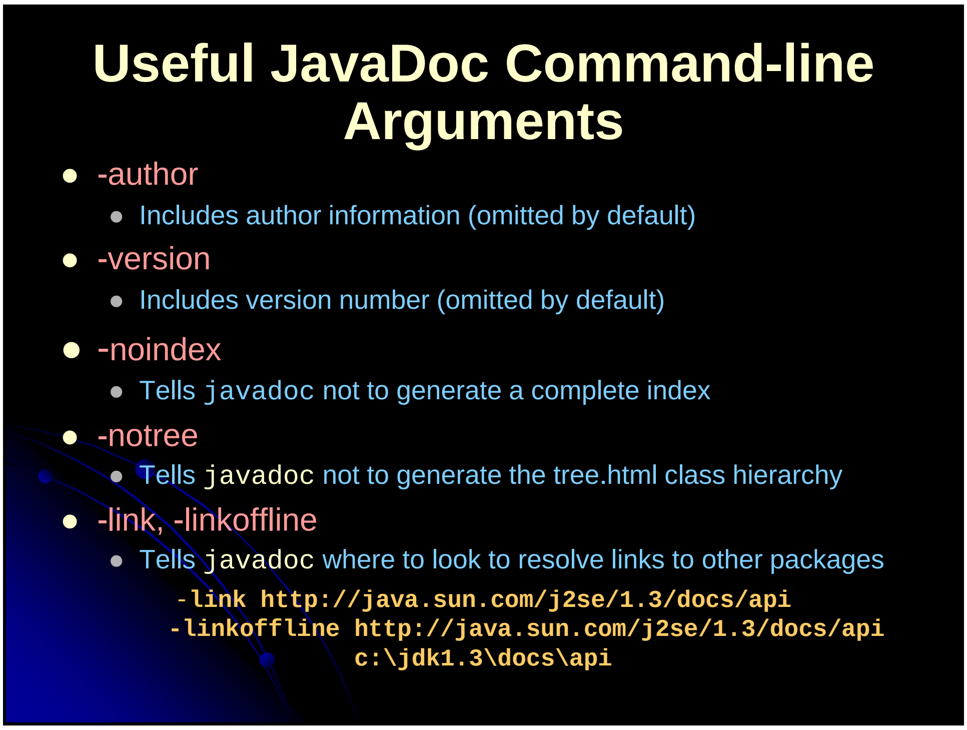
    ```
    javadoc Foo.java Bar.java
    ```
  - JavaDoc 1.4 Home Page
    - http://java.sun.com/j2se/1.4/docs/tooldocs/javadoc/



Useful JavaDoc Tags

- @author
  - Specifies the author of the document
  - Must use **`javadoc –author ...`** to generate in output

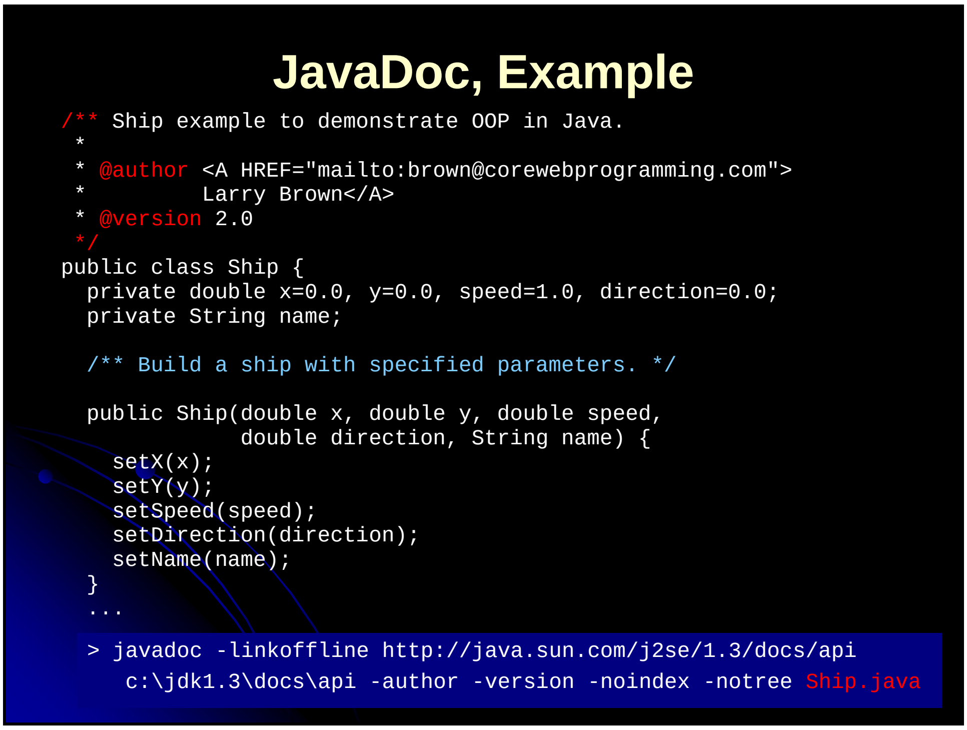
    ```
    /** Description of some class ...
     *
     * @author <A HREF="mailto:brown@lmbrown.com">
     *         Larry Brown</A>
     */
    ```
- @version
  - Version number of the document
  - Must use **`javadoc –version ...`** to generate in output
- @param
  - Documents a method argument
- @return
  - Documents the return type of a method



Useful JavaDoc Command-line Arguments

- -author
  - Includes author information (omitted by default)
- -version
  - Includes version number (omitted by default)
- -noindex
  - Tells `javadoc` not to generate a complete index
- -notree
  - Tells `javadoc` not to generate the tree.html class hierarchy
- -link, -linkoffline
  - Tells `javadoc` where to look to resolve links to other packages

```
 -link http://java.sun.com/j2se/1.3/docs/api
-linkoffline http://java.sun.com/j2se/1.3/docs/api
             c:\jdk1.3\docs\api
```

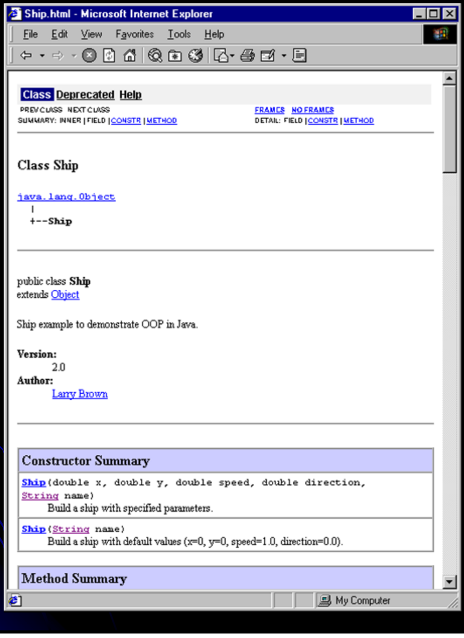


JavaDoc, Example

```
/** Ship example to demonstrate OOP in Java.
 *
 * @author <A HREF="mailto:brown@corewebprogramming.com">
 *         Larry Brown</A>
 * @version 2.0
 */
public class Ship {
  private double x=0.0, y=0.0, speed=1.0, direction=0.0;
  private String name;

  /** Build a ship with specified parameters. */

  public Ship(double x, double y, double speed,
              double direction, String name) {
    setX(x);
    setY(y);
    setSpeed(speed);
    setDirection(direction);
    setName(name);
  }
  ...
```

```
> javadoc -linkoffline http://java.sun.com/j2se/1.3/docs/api
   c:\jdk1.3\docs\api -author -version -noindex -notree Ship.java
```

JavaDoc: Result


Ship.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
My Computer

Class Deprecated Help

PREV CLASS NEXT CLASS | FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD

Class Ship

```
java.lang.Object
  |
  +--Ship
```

public class **Ship**
extends Object

Ship example to demonstrate OOP in Java.

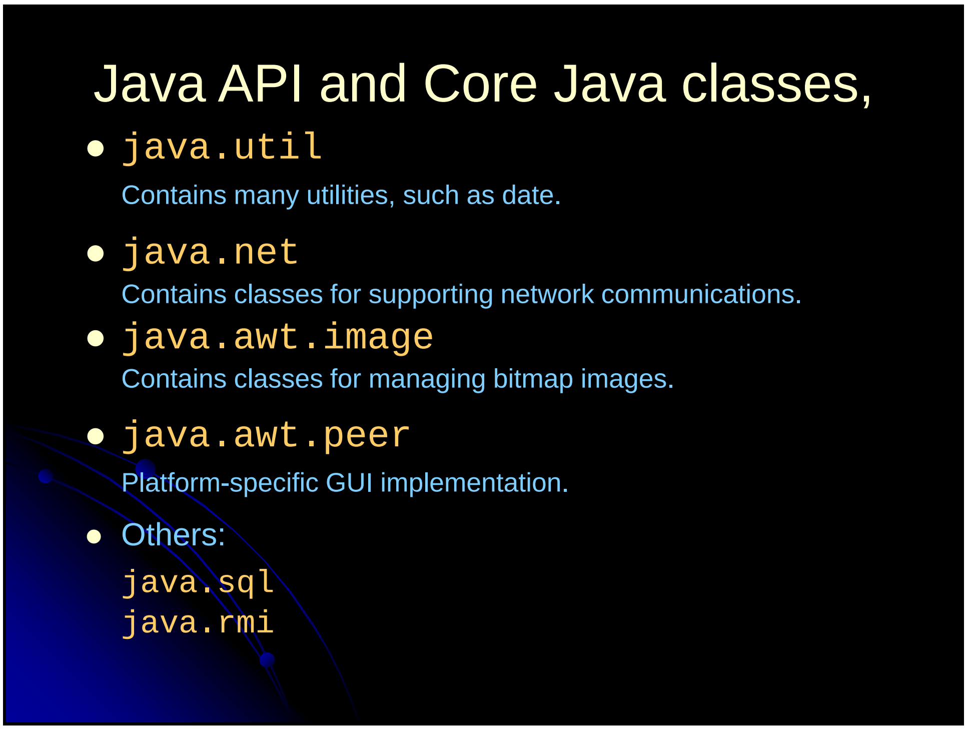
Version:
2.0
Author:
Larry Brown

Constructor Summary
`Ship(double x, double y, double speed, double direction, String name)` Build a ship with specified parameters.
`Ship(String name)` Build a ship with default values (x=0, y=0, speed=1.0, direction=0.0).

Method Summary

Java API and Core Java classes

- `java.lang`
  Contains core Java classes, such as numeric classes, strings, and objects. This package is implicitly imported to every Java program.
- `java.awt`
  Contains classes for graphics.
- `java.io`
  Contains classes for input and output streams and files.
- `java.applet`
  Contains classes for supporting applets.



Java API and Core Java classes,

- `java.util`

  Contains many utilities, such as date.

- `java.net`

  Contains classes for supporting network communications.

- `java.awt.image`

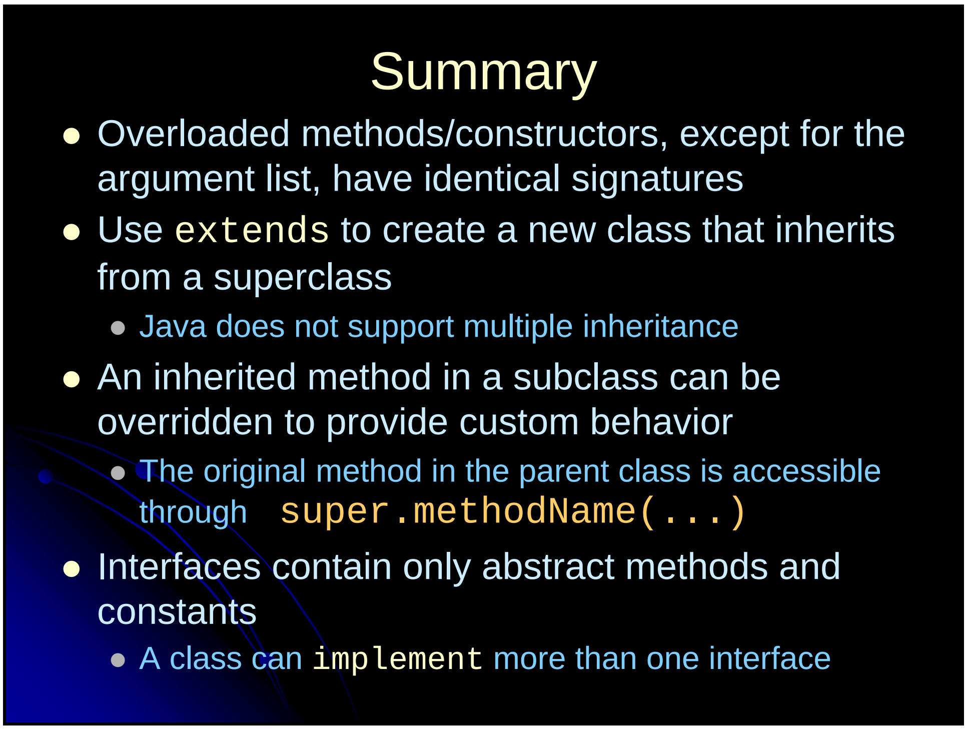
  Contains classes for managing bitmap images.

- `java.awt.peer`

  Platform-specific GUI implementation.

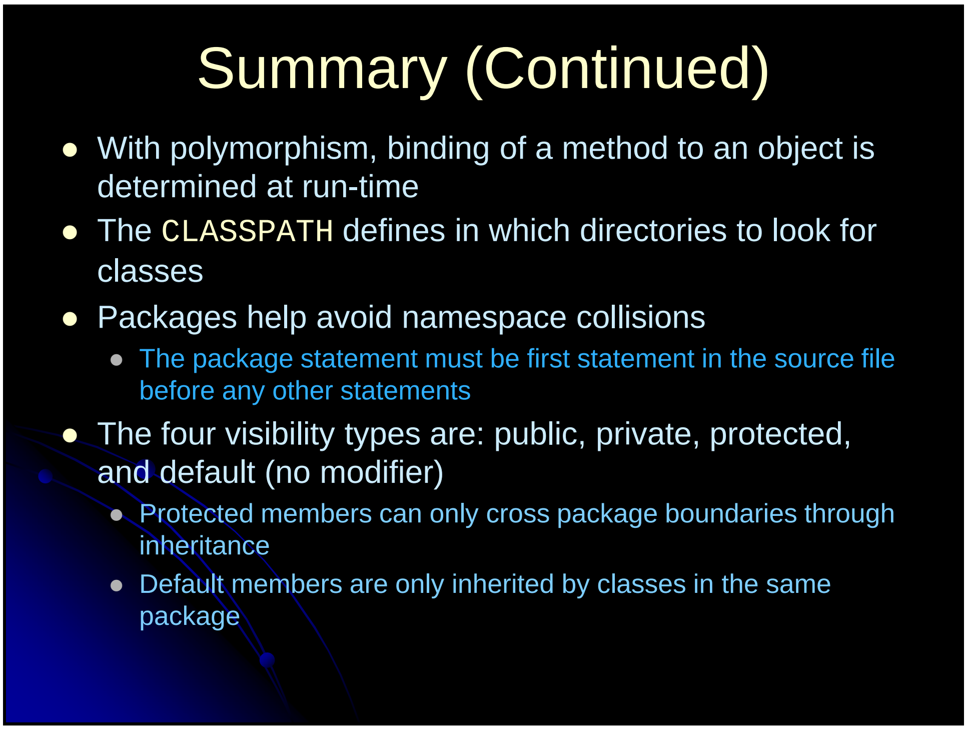
- Others:

  `java.sql`
  `java.rmi`



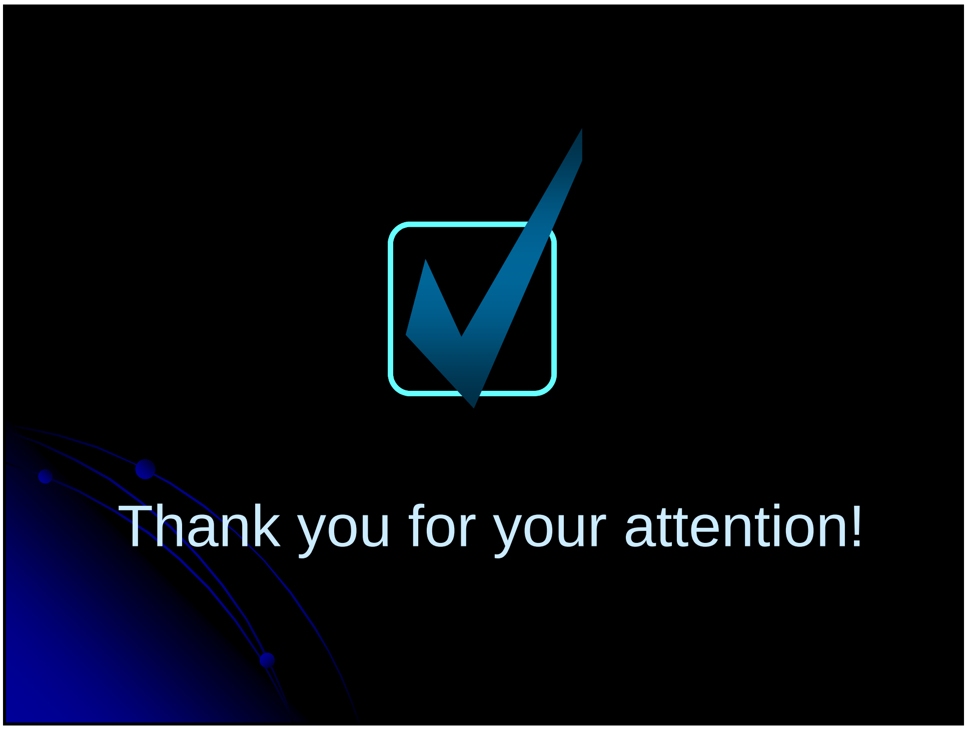
Summary

- Overloaded methods/constructors, except for the argument list, have identical signatures
- Use `extends` to create a new class that inherits from a superclass
  - Java does not support multiple inheritance
- An inherited method in a subclass can be overridden to provide custom behavior
  - The original method in the parent class is accessible through `super.methodName(...)`
- Interfaces contain only abstract methods and constants
  - A class can `implement` more than one interface

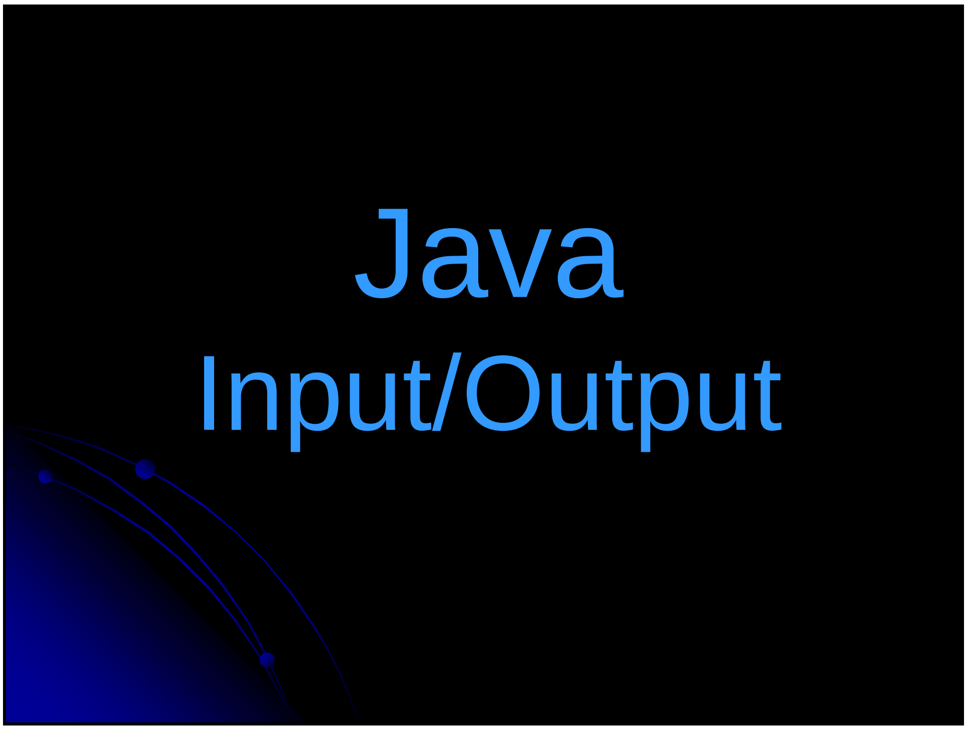


Summary (Continued)

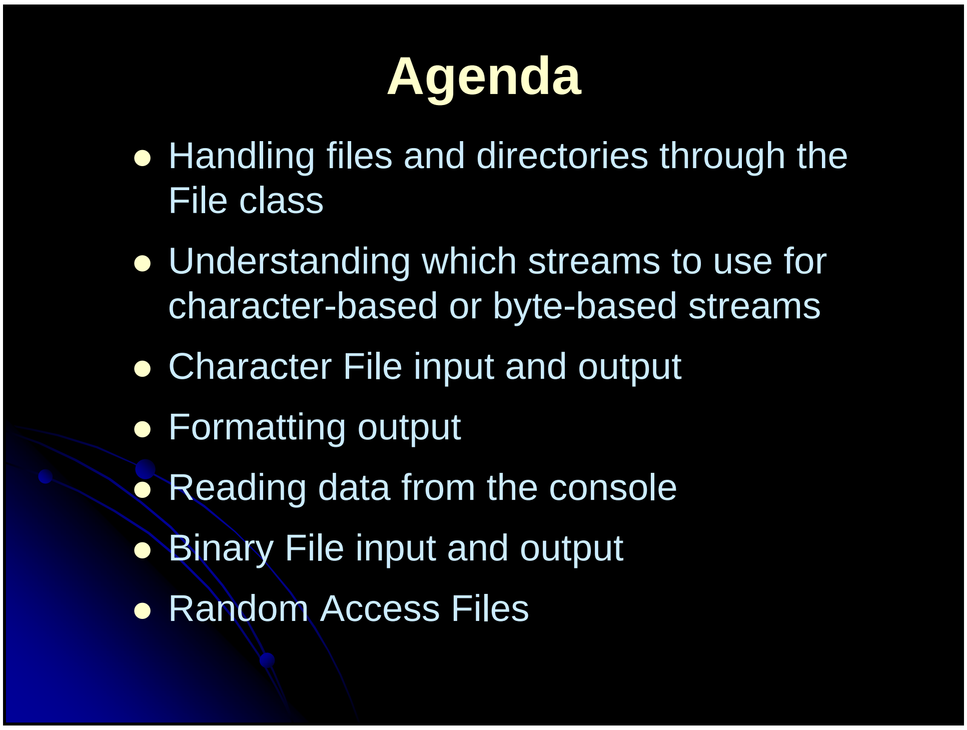
- With polymorphism, binding of a method to an object is determined at run-time
- The `CLASSPATH` defines in which directories to look for classes
- Packages help avoid namespace collisions
  - The package statement must be first statement in the source file before any other statements
- The four visibility types are: public, private, protected, and default (no modifier)
  - Protected members can only cross package boundaries through inheritance
  - Default members are only inherited by classes in the same package



Thank you for your attention!

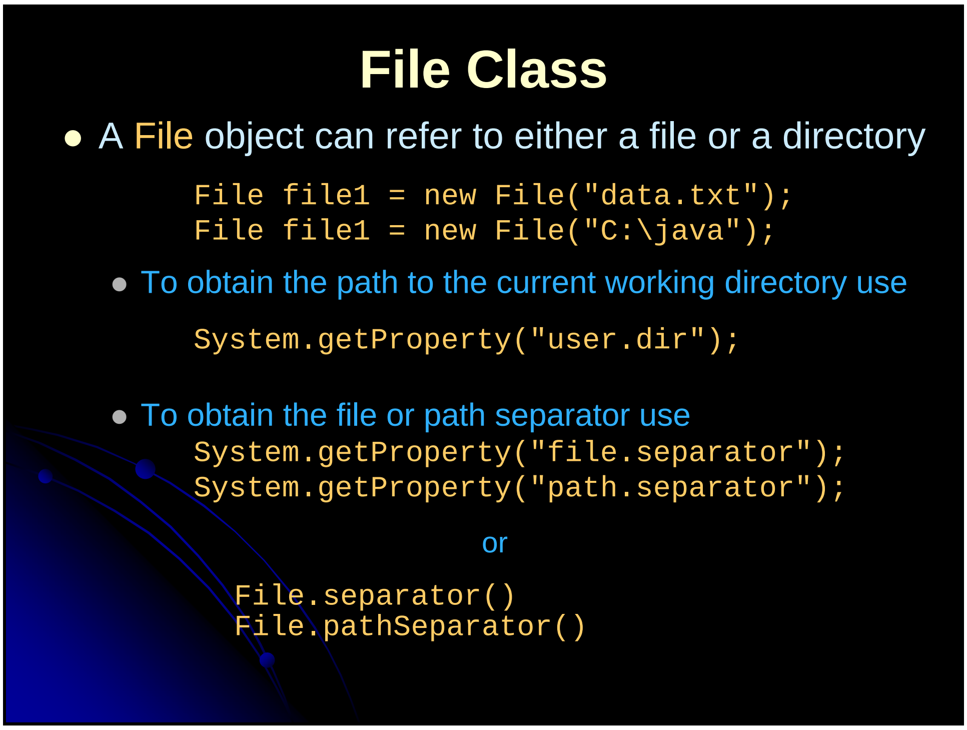


Java
Input/Output



Agenda

- Handling files and directories through the File class
- Understanding which streams to use for character-based or byte-based streams
- Character File input and output
- Formatting output
- Reading data from the console
- Binary File input and output
- Random Access Files



File Class

- A File object can refer to either a file or a directory

```
File file1 = new File("data.txt");
File file1 = new File("C:\java");
```

  - To obtain the path to the current working directory use

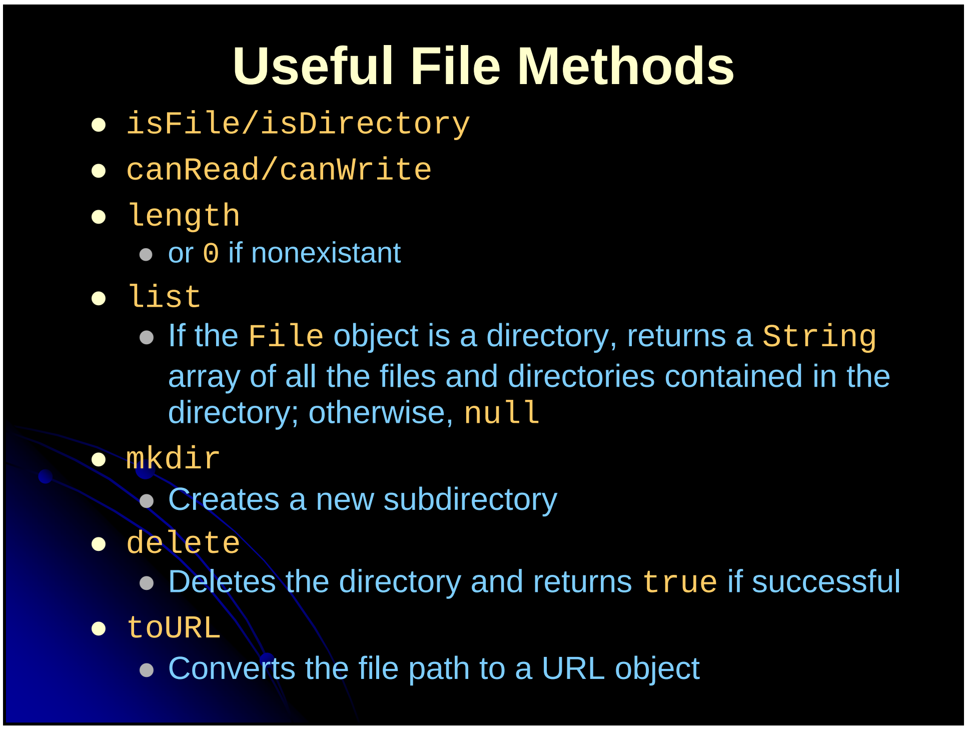
```
System.getProperty("user.dir");
```

  - To obtain the file or path separator use

```
System.getProperty("file.separator");
System.getProperty("path.separator");
```

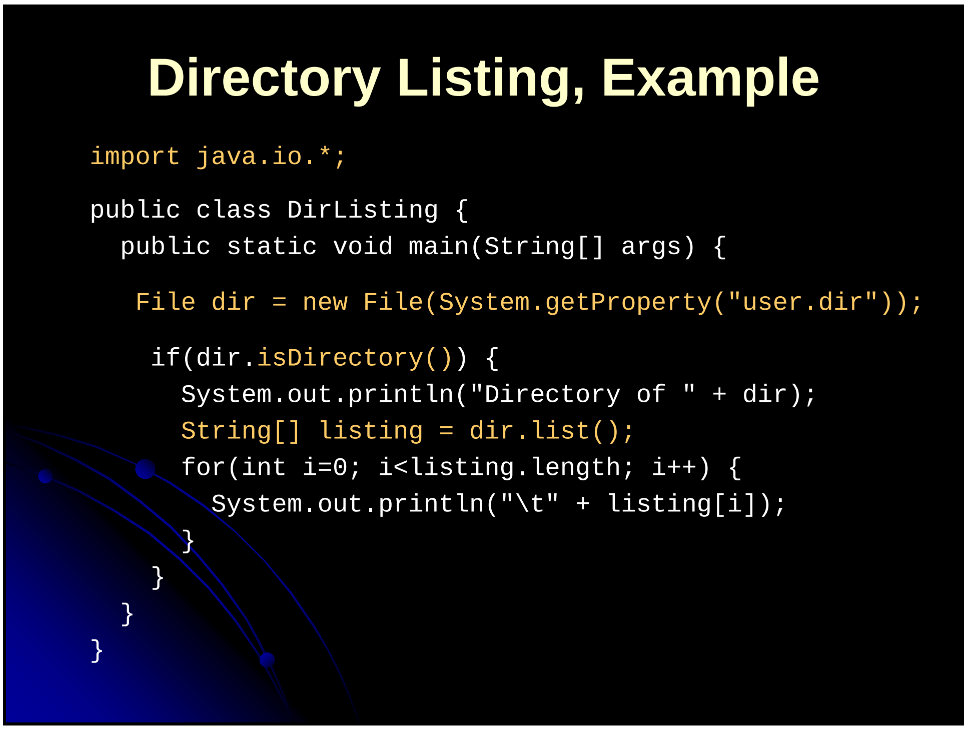
or

```
File.separator()
File.pathSeparator()
```



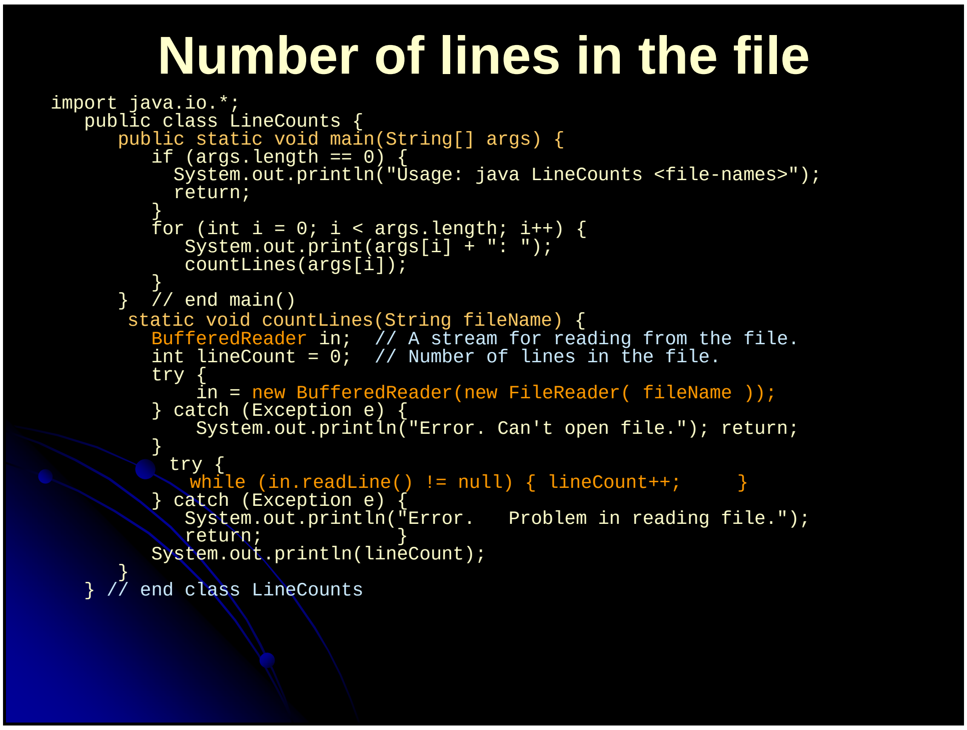
Useful File Methods

- `isFile/isDirectory`
- `canRead/canWrite`
- `length`
  - or `0` if nonexistant
- `list`
  - If the `File` object is a directory, returns a `String` array of all the files and directories contained in the directory; otherwise, `null`
- `mkdir`
  - Creates a new subdirectory
- `delete`
  - Deletes the directory and returns `true` if successful
- `toURL`
  - Converts the file path to a URL object



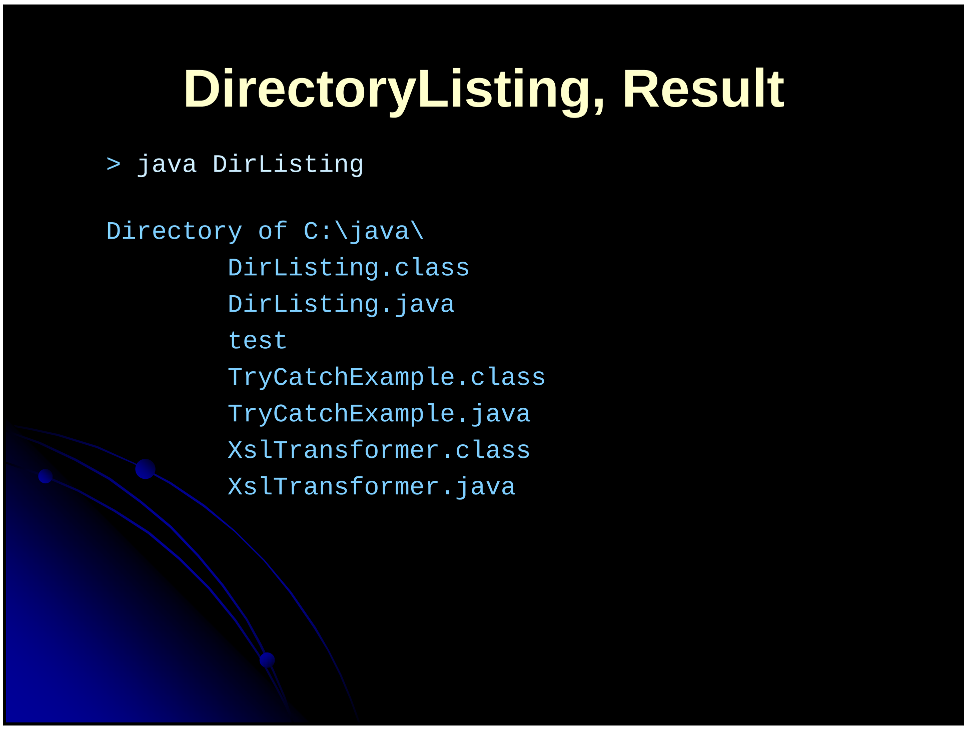
Directory Listing, Example

```
import java.io.*;

public class DirListing {
  public static void main(String[] args) {

   File dir = new File(System.getProperty("user.dir"));

    if(dir.isDirectory()) {
      System.out.println("Directory of " + dir);
      String[] listing = dir.list();
      for(int i=0; i<listing.length; i++) {
        System.out.println("\t" + listing[i]);
      }
    }
  }
}
```



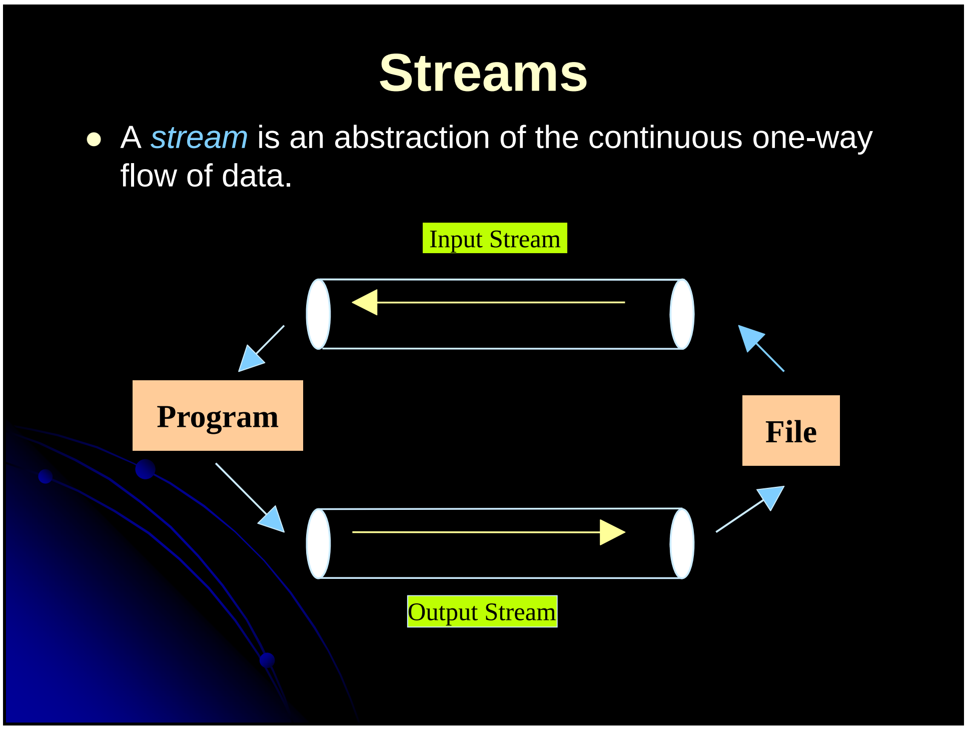
Number of lines in the file

```
import java.io.*;
   public class LineCounts {
      public static void main(String[] args) {
         if (args.length == 0) {
           System.out.println("Usage: java LineCounts <file-names>");
           return;
         }
         for (int i = 0; i < args.length; i++) {
            System.out.print(args[i] + ": ");
            countLines(args[i]);
         }
      }  // end main()
       static void countLines(String fileName) {
         BufferedReader in;  // A stream for reading from the file.
         int lineCount = 0;  // Number of lines in the file.
         try {
             in = new BufferedReader(new FileReader( fileName ));
         } catch (Exception e) {
             System.out.println("Error. Can't open file."); return;
         }
           try {
            while (in.readLine() != null) { lineCount++;     }
         } catch (Exception e) {
            System.out.println("Error.   Problem in reading file.");
            return;           }
         System.out.println(lineCount);
      }
   } // end class LineCounts
```



DirectoryListing, Result

```
> java DirListing

Directory of C:\java\
        DirListing.class
        DirListing.java
        test
        TryCatchExample.class
        TryCatchExample.java
        XslTransformer.class
        XslTransformer.java
```



Streams

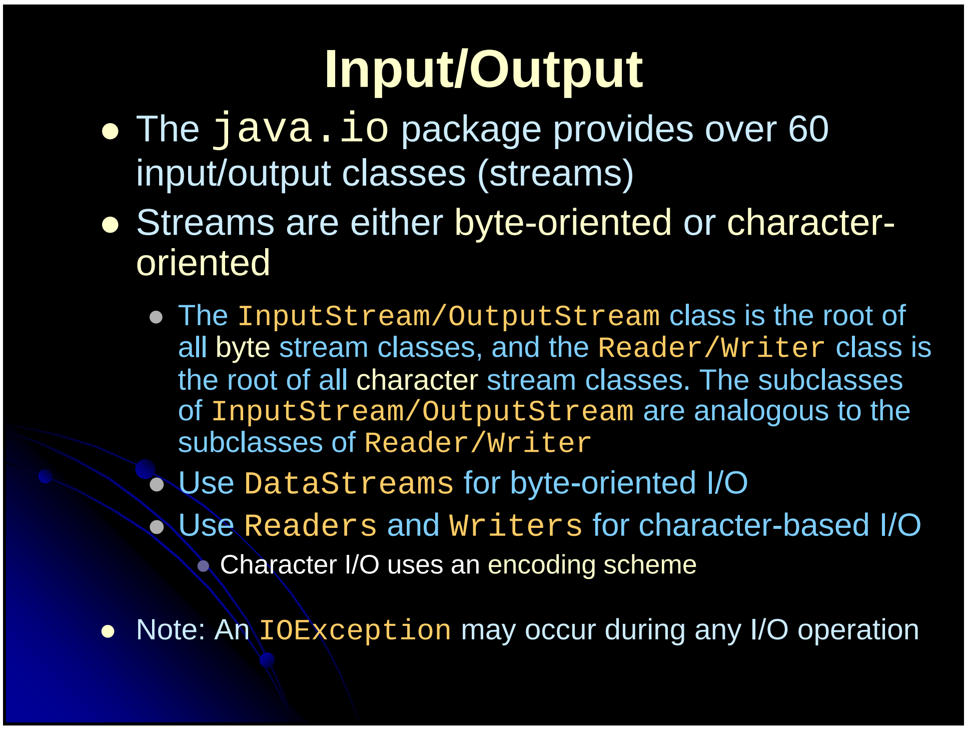
- A *stream* is an abstraction of the continuous one-way flow of data.

Input Stream

Program

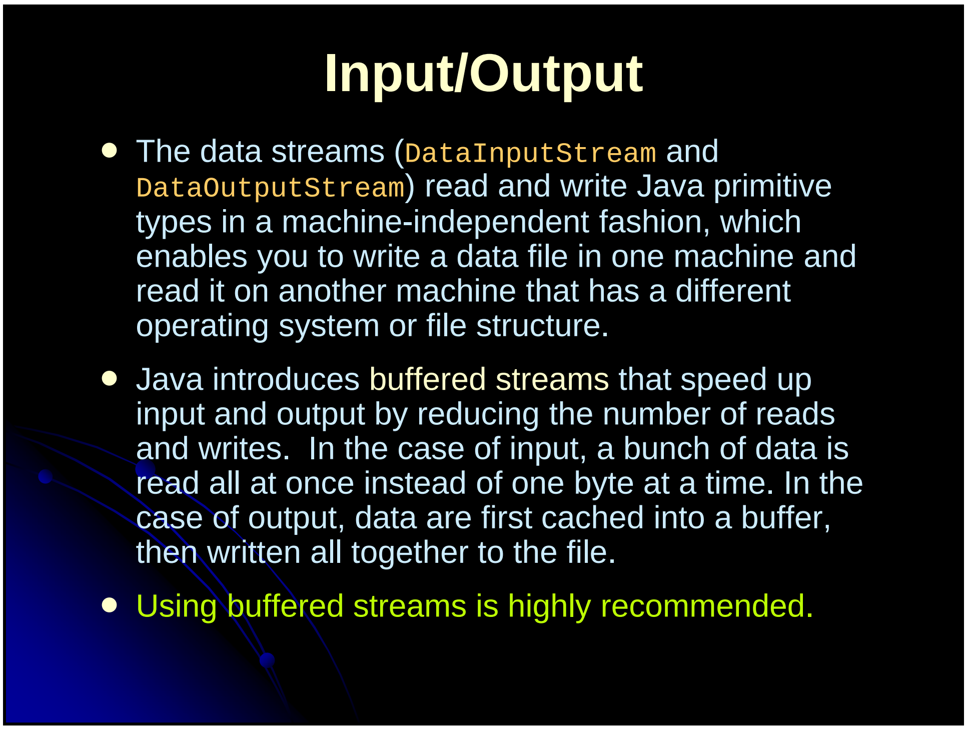
File

Output Stream



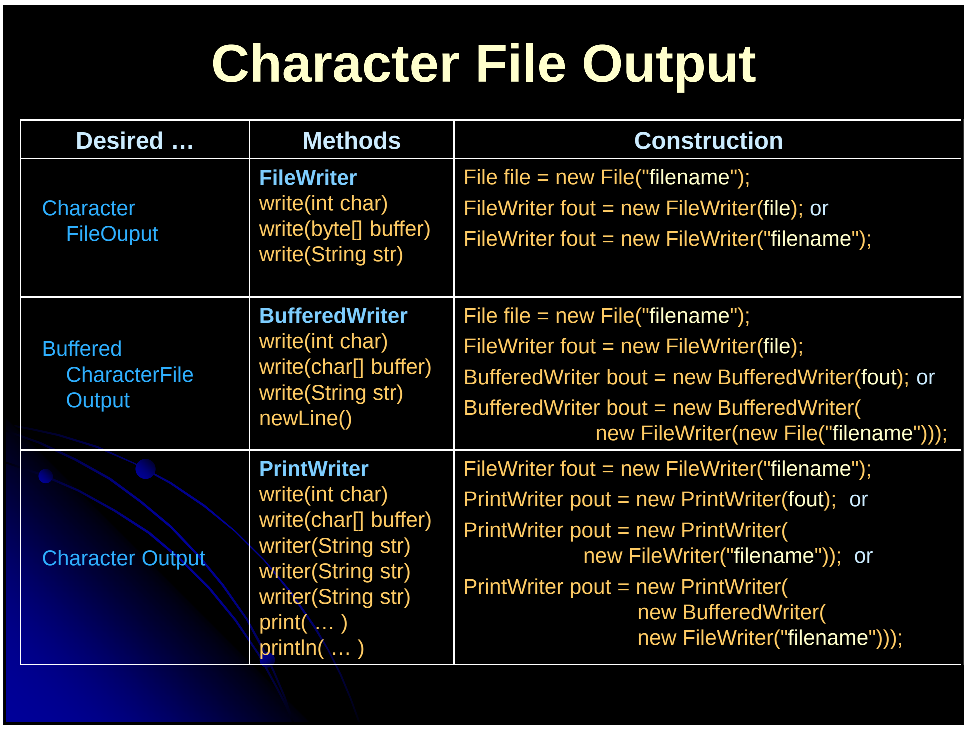
Input/Output

- The `java.io` package provides over 60 input/output classes (streams)
- Streams are either byte-oriented or character-oriented
  - The `InputStream/OutputStream` class is the root of all byte stream classes, and the `Reader/Writer` class is the root of all character stream classes. The subclasses of `InputStream/OutputStream` are analogous to the subclasses of `Reader/Writer`
  - Use `DataStreams` for byte-oriented I/O
  - Use `Readers` and `Writers` for character-based I/O
    - Character I/O uses an encoding scheme
- Note: An `IOException` may occur during any I/O operation



Input/Output

- The data streams (`DataInputStream` and `DataOutputStream`) read and write Java primitive types in a machine-independent fashion, which enables you to write a data file in one machine and read it on another machine that has a different operating system or file structure.
- Java introduces buffered streams that speed up input and output by reducing the number of reads and writes. In the case of input, a bunch of data is read all at once instead of one byte at a time. In the case of output, data are first cached into a buffer, then written all together to the file.
- Using buffered streams is highly recommended.



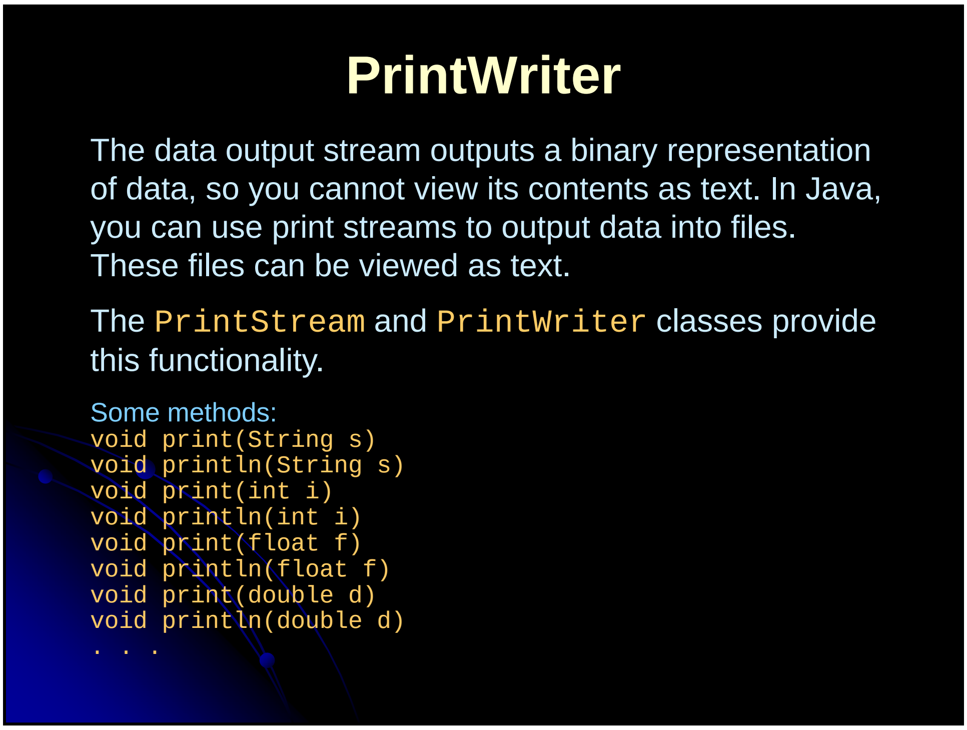
Character File Output

Desired …	Methods	Construction

Character
FileOuput
	FileWriter
	write(int char)
	write(byte[] buffer)
	write(String str)
		File file = new File("filename");
		FileWriter fout = new FileWriter(file); or
		FileWriter fout = new FileWriter("filename");

Buffered
CharacterFile
Output
	BufferedWriter
	write(int char)
	write(char[] buffer)
	write(String str)
	newLine()
		File file = new File("filename");
		FileWriter fout = new FileWriter(file);
		BufferedWriter bout = new BufferedWriter(fout); or
		BufferedWriter bout = new BufferedWriter(
		new FileWriter(new File("filename")));

Character Output	**PrintWriter**
	write(int char)
	write(char[] buffer)
	writer(String str)
	writer(String str)
	writer(String str)
	print(…)
	println(…)
		FileWriter fout = new FileWriter("filename");
		PrintWriter pout = new PrintWriter(fout); or
		PrintWriter pout = new PrintWriter(
		new FileWriter("filename")); or
		PrintWriter pout = new PrintWriter(
		new BufferedWriter(
		new FileWriter("filename")));



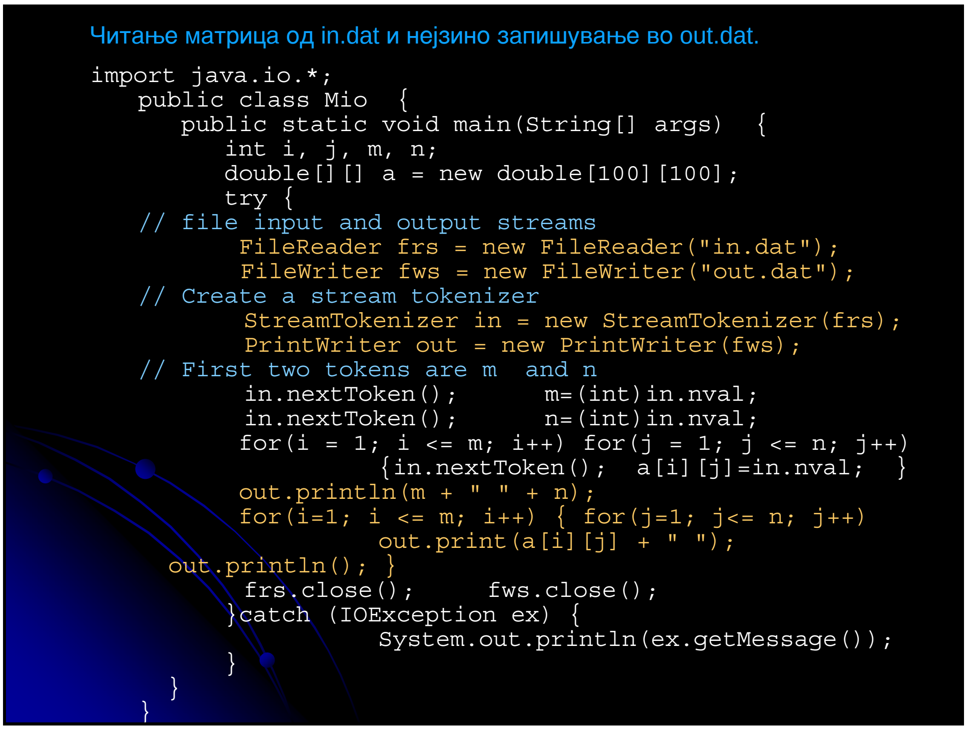
PrintWriter

The data output stream outputs a binary representation of data, so you cannot view its contents as text. In Java, you can use print streams to output data into files. These files can be viewed as text.

The `PrintStream` and `PrintWriter` classes provide this functionality.

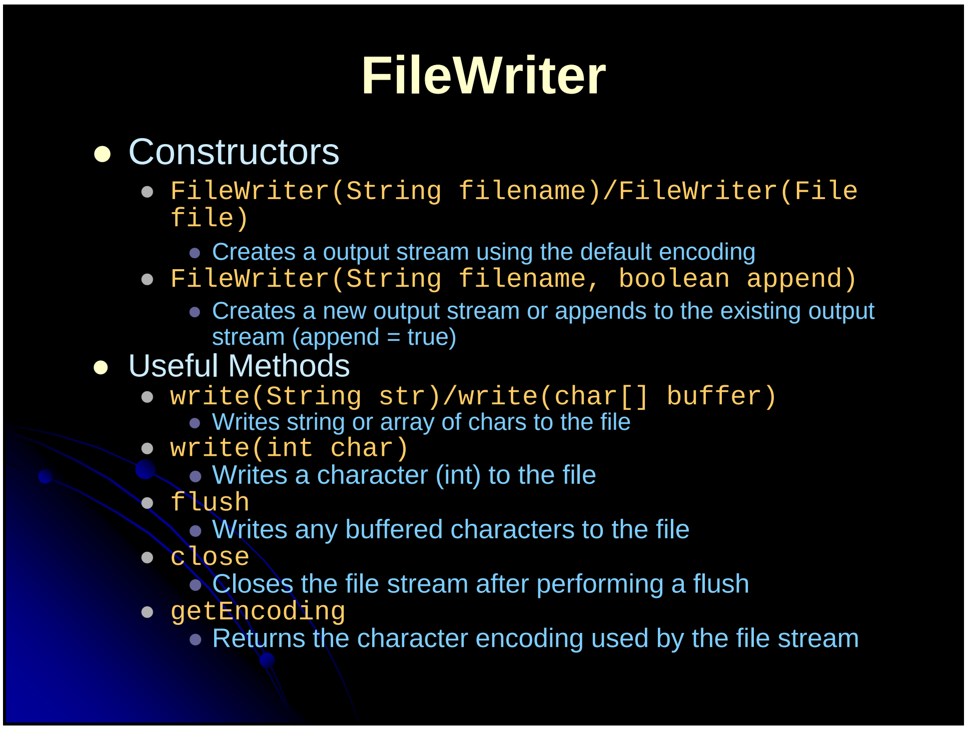
Some methods:

```
void print(String s)
void println(String s)
void print(int i)
void println(int i)
void print(float f)
void println(float f)
void print(double d)
void println(double d)
. . .
```



Читање матрица од in.dat и нејзино запишување во out.dat.

```
import java.io.*;
   public class Mio  {
      public static void main(String[] args)  {
         int i, j, m, n;
         double[][] a = new double[100][100];
         try {
   // file input and output streams
            FileReader frs = new FileReader("in.dat");
            FileWriter fws = new FileWriter("out.dat");
   // Create a stream tokenizer
            StreamTokenizer in = new StreamTokenizer(frs);
            PrintWriter out = new PrintWriter(fws);
   // First two tokens are m  and n
            in.nextToken();      m=(int)in.nval;
            in.nextToken();      n=(int)in.nval;
            for(i = 1; i <= m; i++) for(j = 1; j <= n; j++)
                     {in.nextToken();  a[i][j]=in.nval;  }
            out.println(m + " " + n);
            for(i=1; i <= m; i++) { for(j=1; j<= n; j++)
                     out.print(a[i][j] + " ");
     out.println(); }
            frs.close();     fws.close();
         }catch (IOException ex) {
                     System.out.println(ex.getMessage());
         }
      }
   }
```



FileWriter

- Constructors
  - `FileWriter(String filename)/FileWriter(File file)`
    - Creates a output stream using the default encoding
  - `FileWriter(String filename, boolean append)`
    - Creates a new output stream or appends to the existing output stream (append = true)
- Useful Methods
  - `write(String str)/write(char[] buffer)`
    - Writes string or array of chars to the file
  - `write(int char)`
    - Writes a character (int) to the file
  - `flush`
    - Writes any buffered characters to the file
  - `close`
    - Closes the file stream after performing a flush
  - `getEncoding`
    - Returns the character encoding used by the file stream

CharacterFileOutput, Example

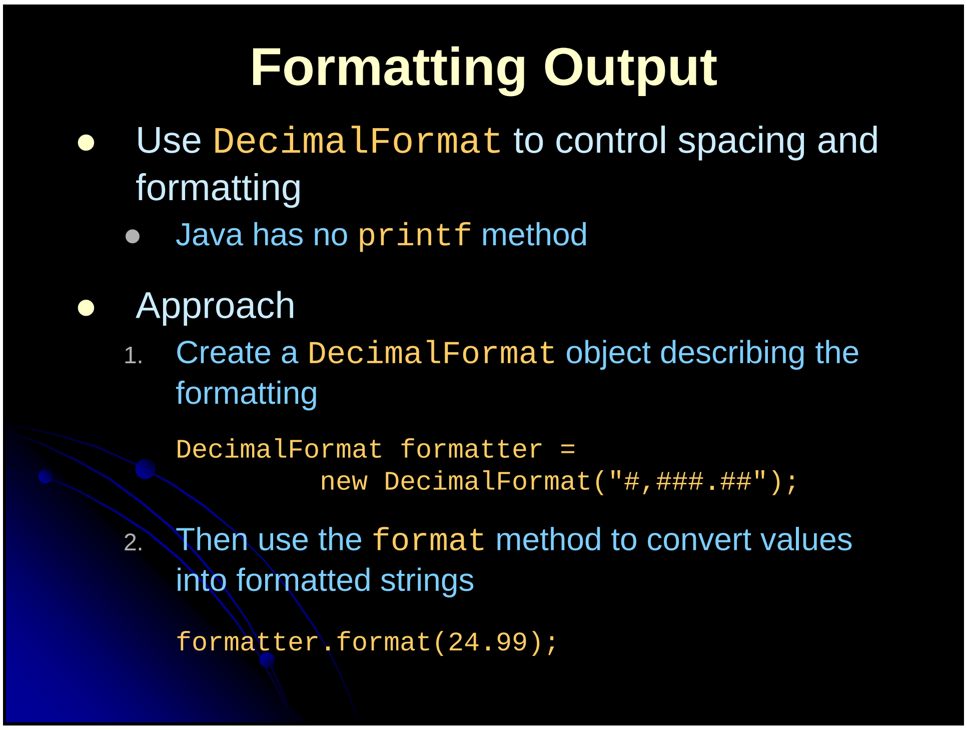
```
import java.io.*;

public class CharacterFileOutput {
  public static void main(String[] args) {
    FileWriter out = null;

    try {
      out = new FileWriter("book.txt");
      System.out.println("Encoding: " + out.getEncoding());
      out.write("Core Web Programming");
      out.close();
      out = null;
    } catch(IOException ioe) {
      System.out.println("IO problem: " + ioe);
      ioe.printStackTrace();
      try {
        if (out != null) {
          out.close();
        }
      } catch(IOException ioe2) { }
    }
  }
}
```

CharacterFileOutput, Result

```
> java CharacterFileOutput
Encoding: Cp1252

> type book.txt
Core Web Programming
```

- Note: Cp1252 is Windows Western Europe / Latin-1
  - To change the system default encoding use `System.setProperty("file.encoding", "`*`encoding`*`");`
  - To specify the encoding when creating the output steam, use an `OutputStreamWriter`

```
OutputStreamWriter out =
   new OutputStreamWriter(
     new FileOutputStream("book.txt", "8859_1"));
```



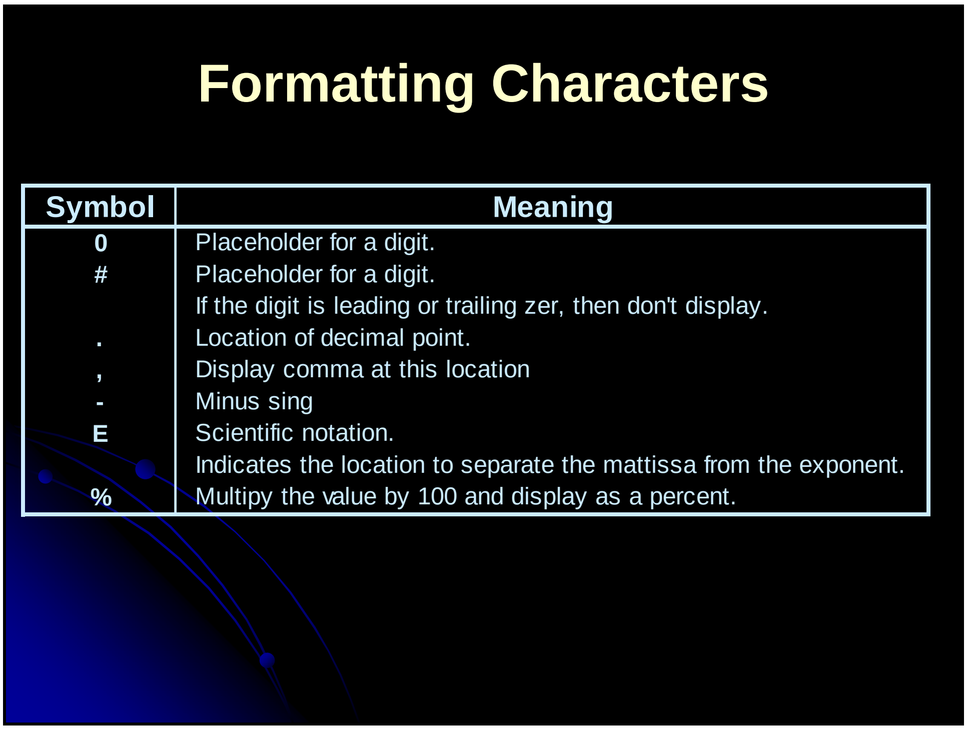
Formatting Output

- Use `DecimalFormat` to control spacing and formatting
  - Java has no `printf` method
- Approach
  1. Create a `DecimalFormat` object describing the formatting

     ```
     DecimalFormat formatter =
               new DecimalFormat("#,###.##");
     ```

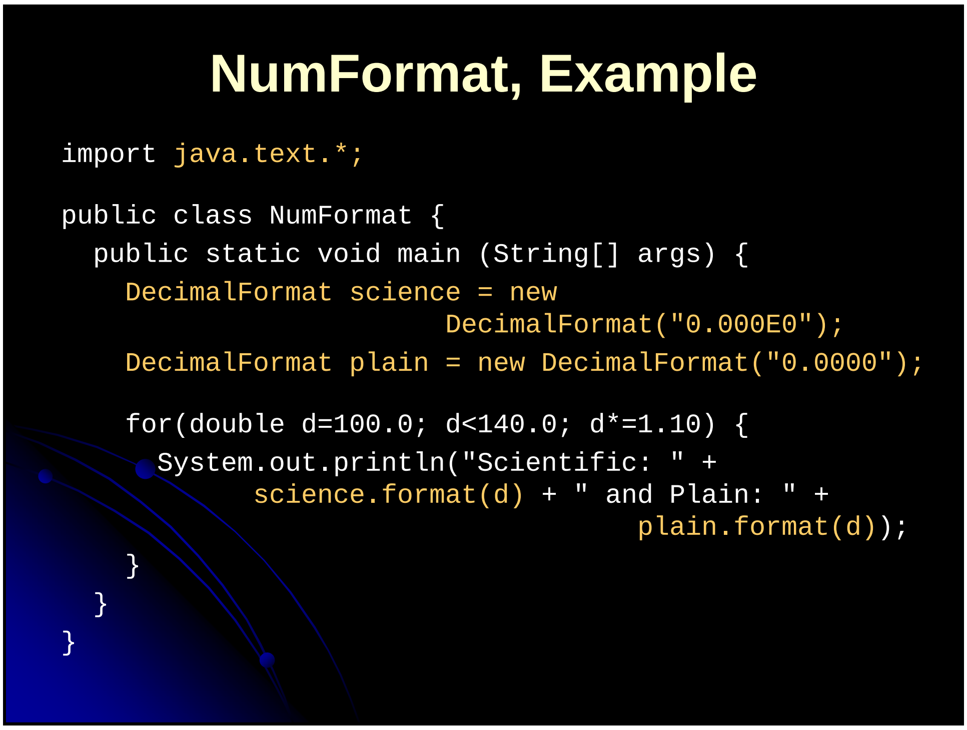
  2. Then use the `format` method to convert values into formatted strings

     ```
     formatter.format(24.99);
     ```



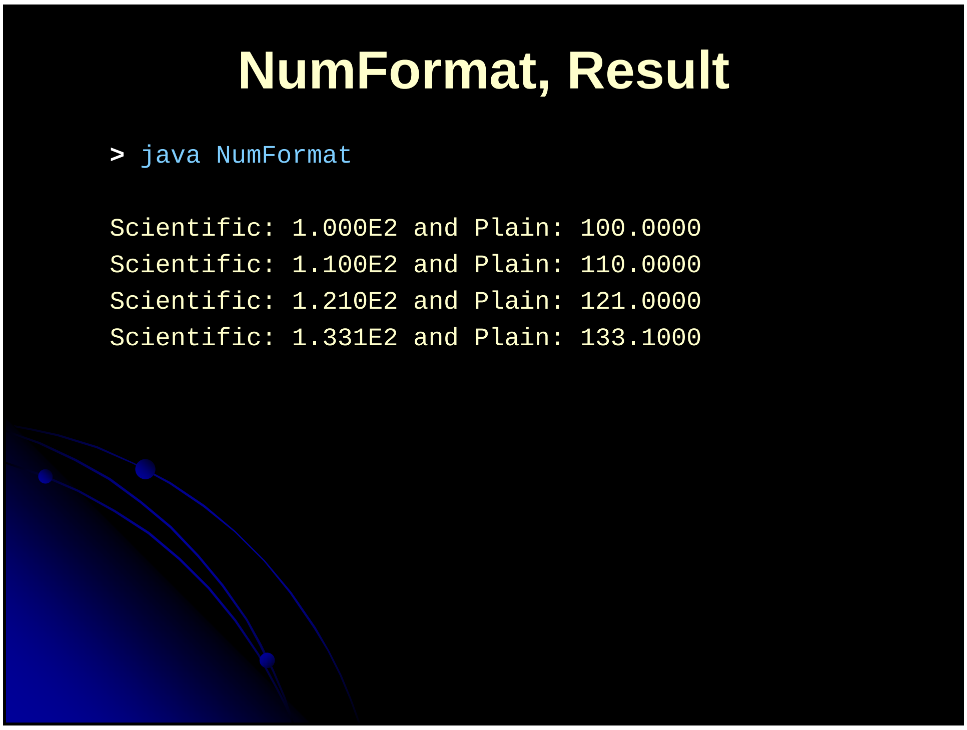
Formatting Characters

Symbol	Meaning
0	Placeholder for a digit.
#	Placeholder for a digit.
	If the digit is leading or trailing zer, then don't display.
.	Location of decimal point.
,	Display comma at this location
-	Minus sing
E	Scientific notation.
	Indicates the location to separate the mattissa from the exponent.
%	Multipy the value by 100 and display as a percent.



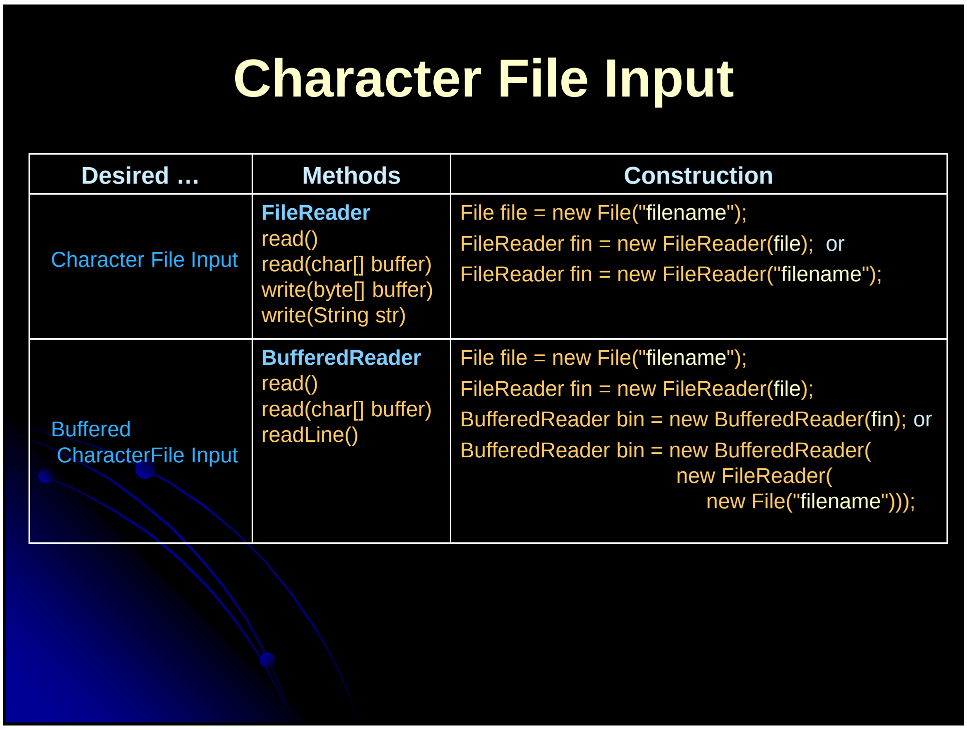
NumFormat, Example

```
import java.text.*;

public class NumFormat {
  public static void main (String[] args) {
    DecimalFormat science = new
                        DecimalFormat("0.000E0");
    DecimalFormat plain = new DecimalFormat("0.0000");

    for(double d=100.0; d<140.0; d*=1.10) {
      System.out.println("Scientific: " +
            science.format(d) + " and Plain: " +
                                    plain.format(d));
    }
  }
}
```



NumFormat, Result

```
> java NumFormat

Scientific: 1.000E2 and Plain: 100.0000
Scientific: 1.100E2 and Plain: 110.0000
Scientific: 1.210E2 and Plain: 121.0000
Scientific: 1.331E2 and Plain: 133.1000
```

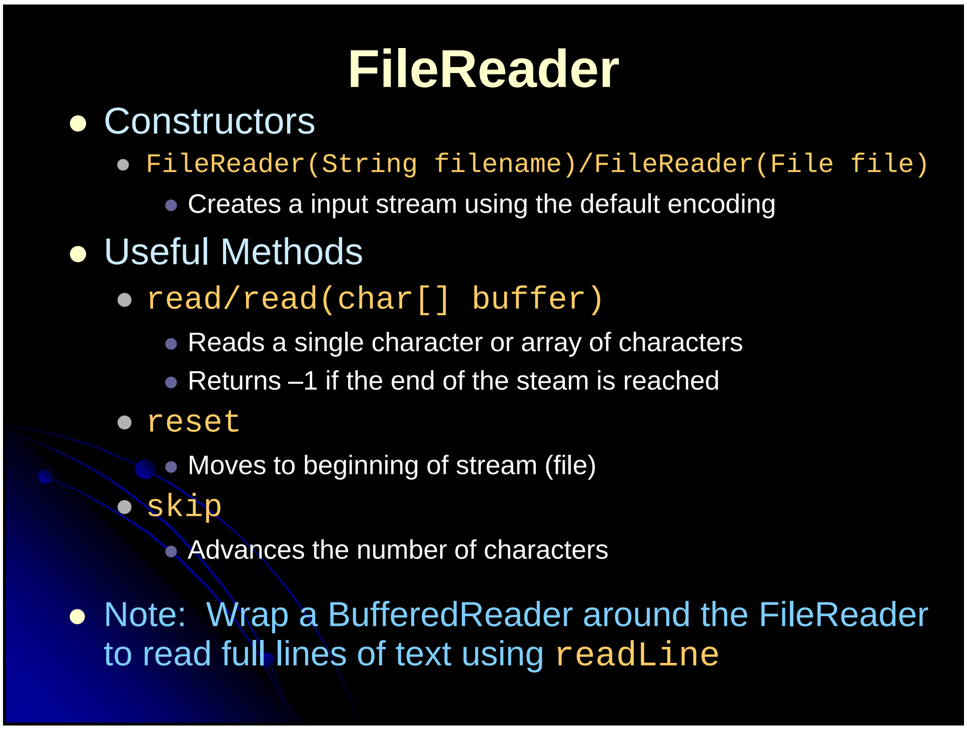


Character File Input

Desired …	Methods	Construction

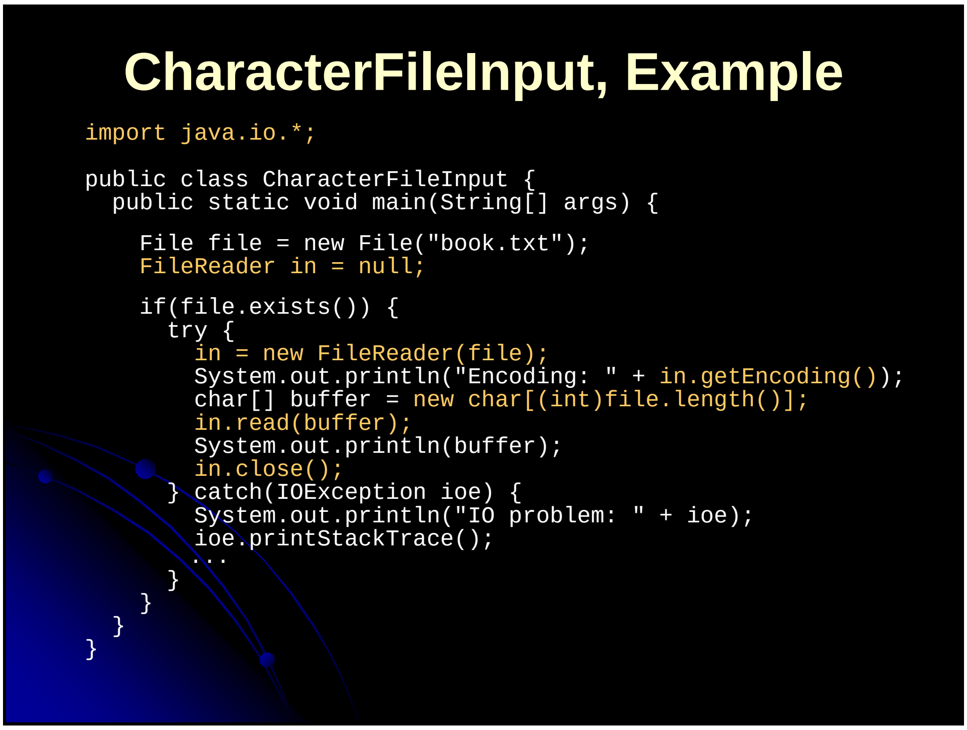
Character File Input	**FileReader**
	read()
	read(char[] buffer)
	write(byte[] buffer)
	write(String str)
		File file = new File("filename");
		FileReader fin = new FileReader(file); or
		FileReader fin = new FileReader("filename");

Buffered CharacterFile Input	**BufferedReader**
	read()
	read(char[] buffer)
	readLine()
		File file = new File("filename");
		FileReader fin = new FileReader(file);
		BufferedReader bin = new BufferedReader(fin); or
		BufferedReader bin = new BufferedReader(new FileReader(new File("filename")));



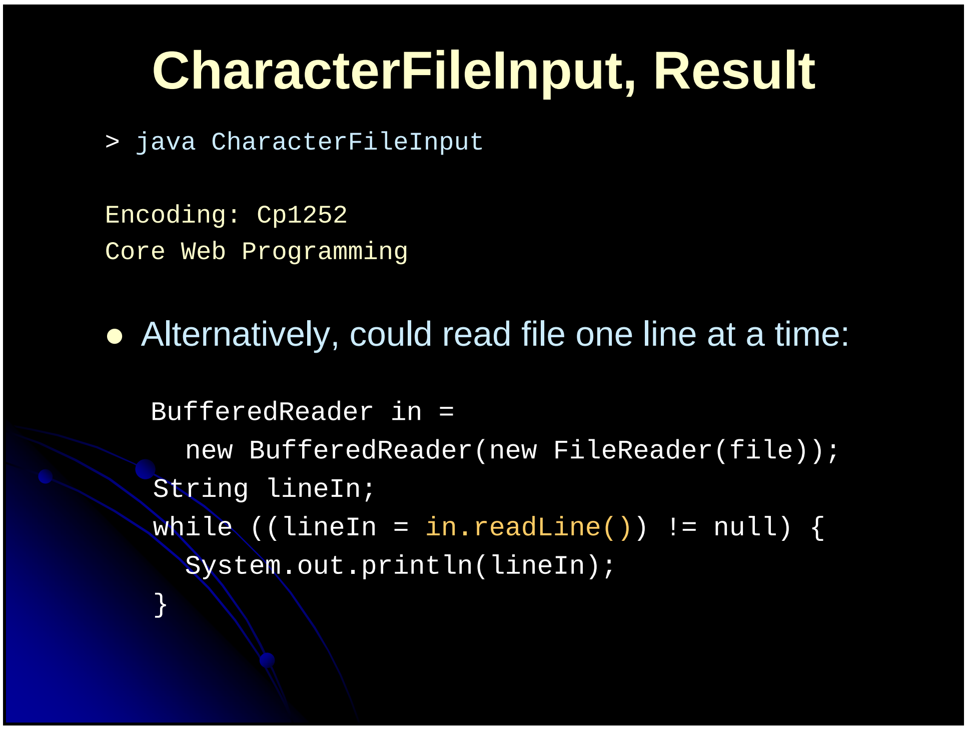
FileReader

- Constructors
  - `FileReader(String filename)/FileReader(File file)`
    - Creates a input stream using the default encoding
- Useful Methods
  - `read/read(char[] buffer)`
    - Reads a single character or array of characters
    - Returns –1 if the end of the steam is reached
  - `reset`
    - Moves to beginning of stream (file)
  - `skip`
    - Advances the number of characters
- Note: Wrap a BufferedReader around the FileReader to read full lines of text using `readLine`



CharacterFileInput, Example

```
import java.io.*;

public class CharacterFileInput {
  public static void main(String[] args) {

    File file = new File("book.txt");
    FileReader in = null;

    if(file.exists()) {
      try {
        in = new FileReader(file);
        System.out.println("Encoding: " + in.getEncoding());
        char[] buffer = new char[(int)file.length()];
        in.read(buffer);
        System.out.println(buffer);
        in.close();
      } catch(IOException ioe) {
        System.out.println("IO problem: " + ioe);
        ioe.printStackTrace();
        ...
      }
    }
  }
}
```

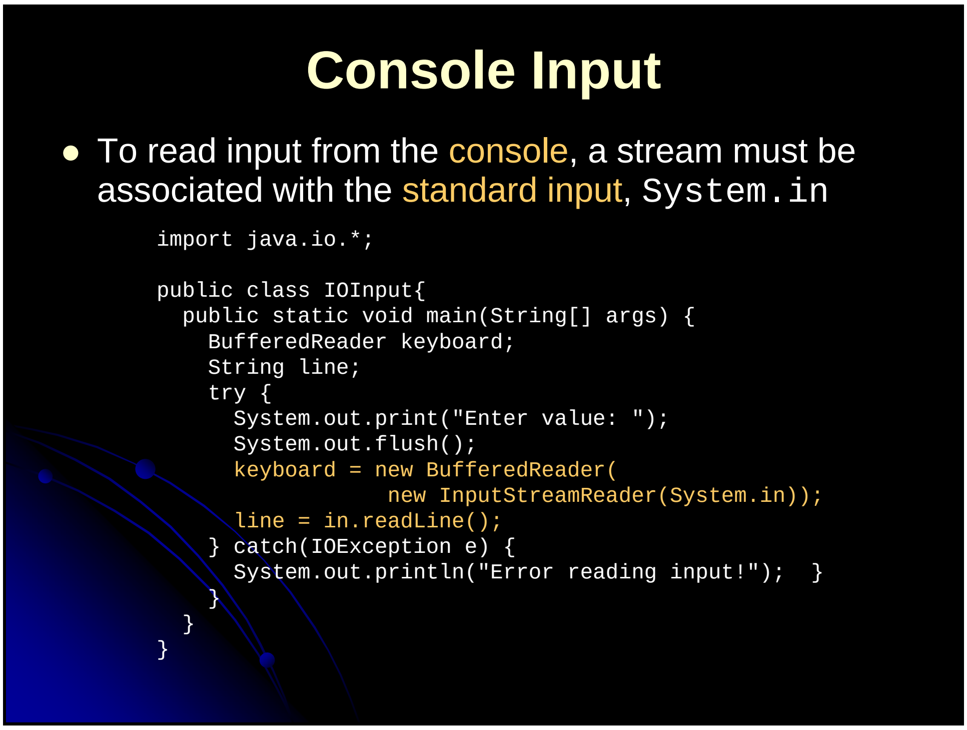


CharacterFileInput, Result

```
> java CharacterFileInput

Encoding: Cp1252
Core Web Programming
```

- Alternatively, could read file one line at a time:

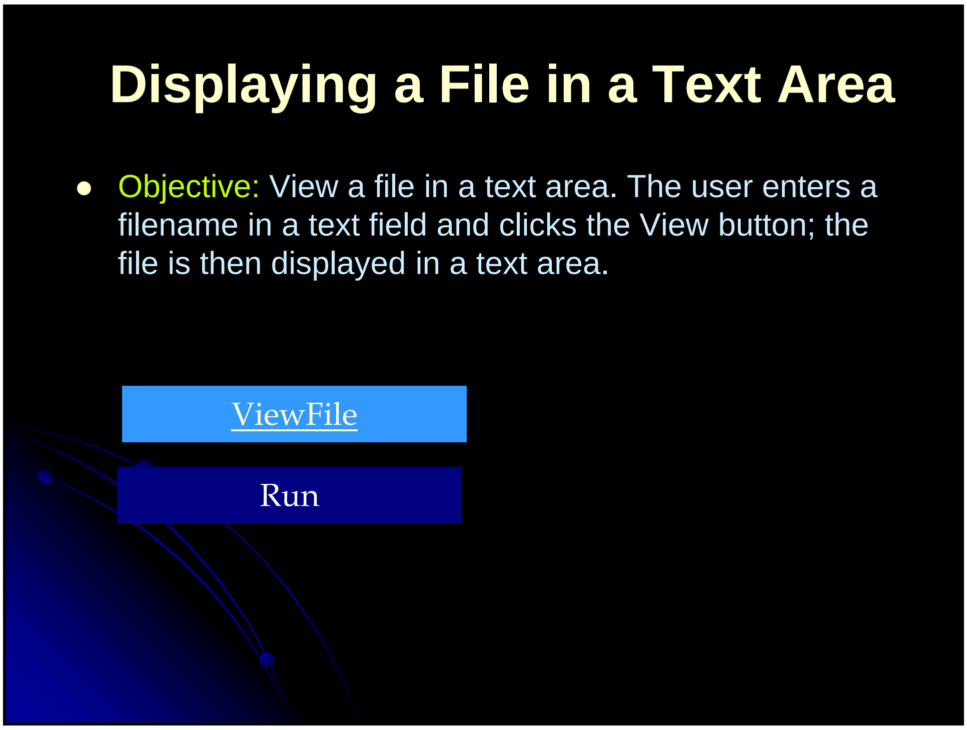
```
BufferedReader in =
  new BufferedReader(new FileReader(file));
String lineIn;
while ((lineIn = in.readLine()) != null) {
  System.out.println(lineIn);
}
```



Console Input

- To read input from the console, a stream must be associated with the standard input, `System.in`

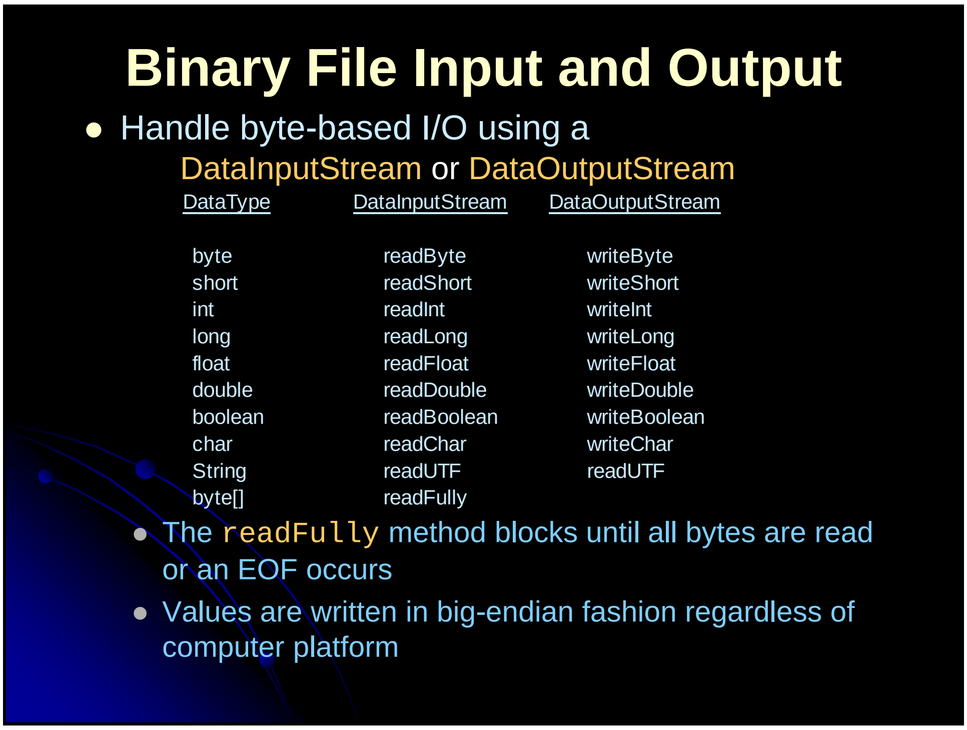
```
import java.io.*;

public class IOInput{
  public static void main(String[] args) {
    BufferedReader keyboard;
    String line;
    try {
      System.out.print("Enter value: ");
      System.out.flush();
      keyboard = new BufferedReader(
                   new InputStreamReader(System.in));
      line = in.readLine();
    } catch(IOException e) {
      System.out.println("Error reading input!");  }
    }
  }
}
```



Displaying a File in a Text Area

- Objective: View a file in a text area. The user enters a filename in a text field and clicks the View button; the file is then displayed in a text area.

ViewFile
Run

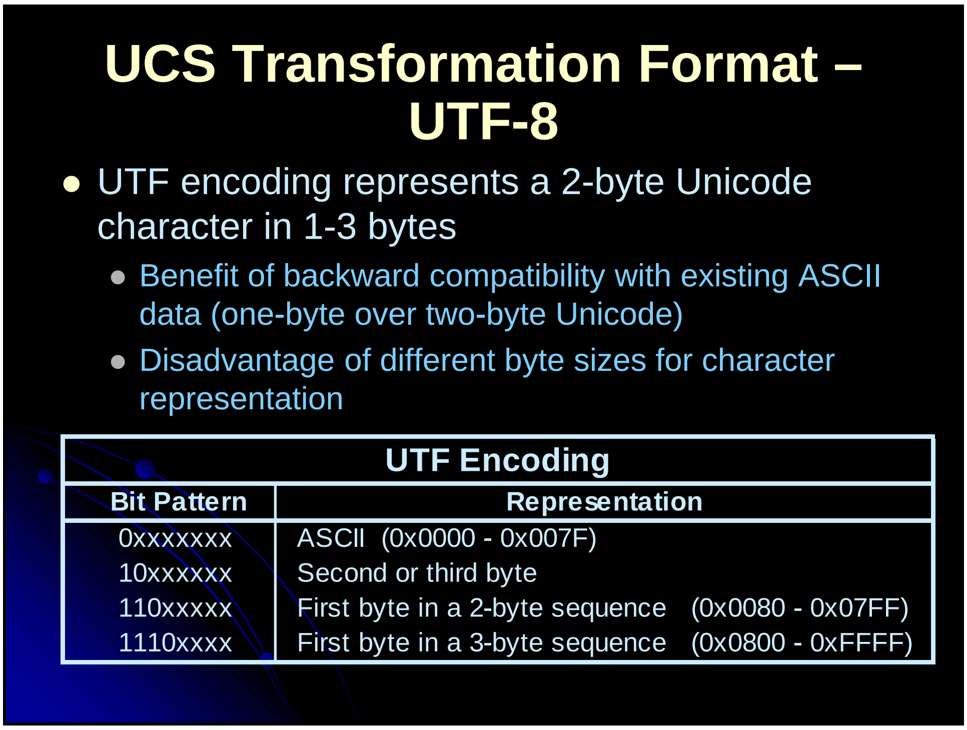


Binary File Input and Output

- Handle byte-based I/O using a DataInputStream or DataOutputStream

DataType	DataInputStream	DataOutputStream
byte	readByte	writeByte
short	readShort	writeShort
int	readInt	writeInt
long	readLong	writeLong
float	readFloat	writeFloat
double	readDouble	writeDouble
boolean	readBoolean	writeBoolean
char	readChar	writeChar
String	readUTF	readUTF
byte[]	readFully	

  - The `readFully` method blocks until all bytes are read or an EOF occurs
  - Values are written in big-endian fashion regardless of computer platform

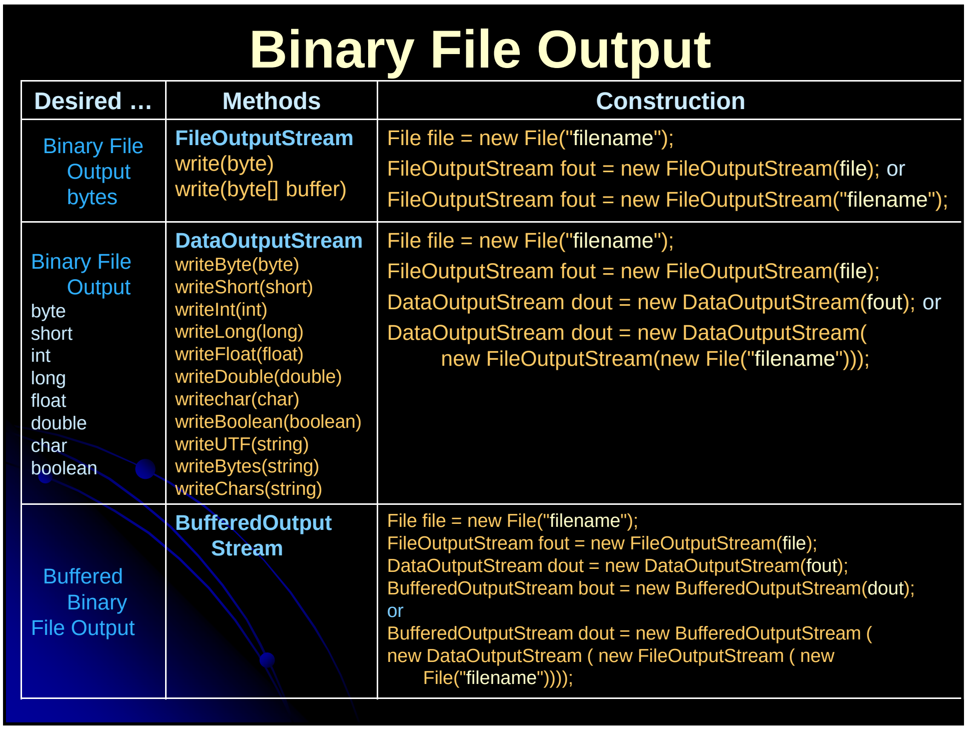


UCS Transformation Format – UTF-8

- UTF encoding represents a 2-byte Unicode character in 1-3 bytes
  - Benefit of backward compatibility with existing ASCII data (one-byte over two-byte Unicode)
  - Disadvantage of different byte sizes for character representation

UTF Encoding

Bit Pattern	Representation
0xxxxxxx	ASCII (0x0000 - 0x007F)
10xxxxxx	Second or third byte
110xxxxx	First byte in a 2-byte sequence (0x0080 - 0x07FF)
1110xxxx	First byte in a 3-byte sequence (0x0800 - 0xFFFF)



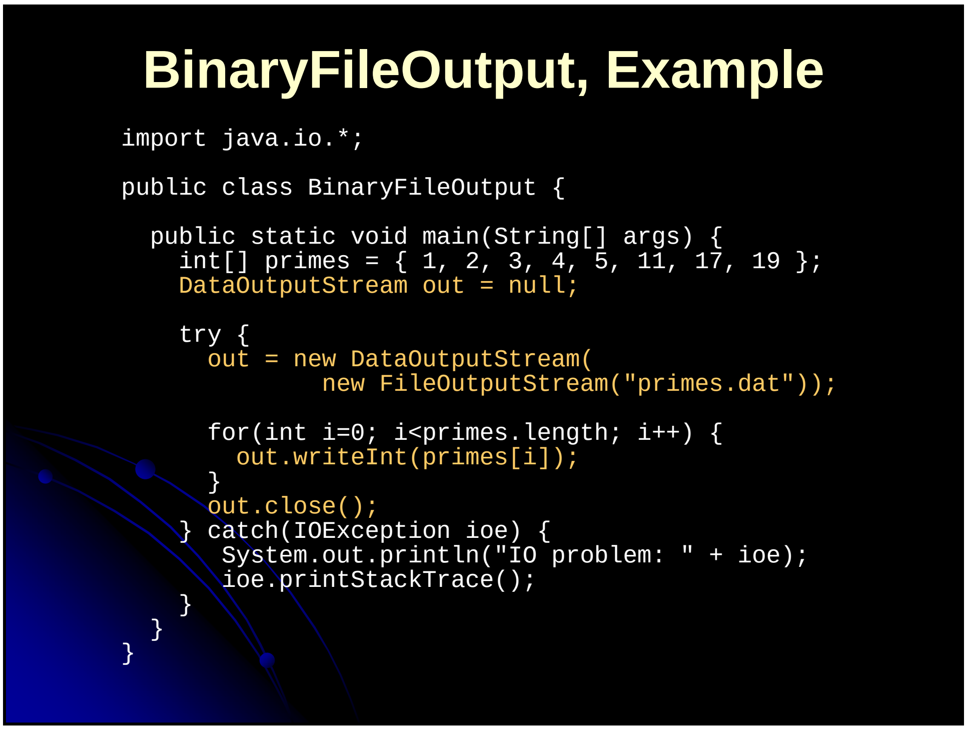
Binary File Output

Desired …	Methods	Construction

Binary File Output bytes	**FileOutputStream**
	write(byte)
	write(byte[] buffer)
		File file = new File("filename");
		FileOutputStream fout = new FileOutputStream(file); or
		FileOutputStream fout = new FileOutputStream("filename");

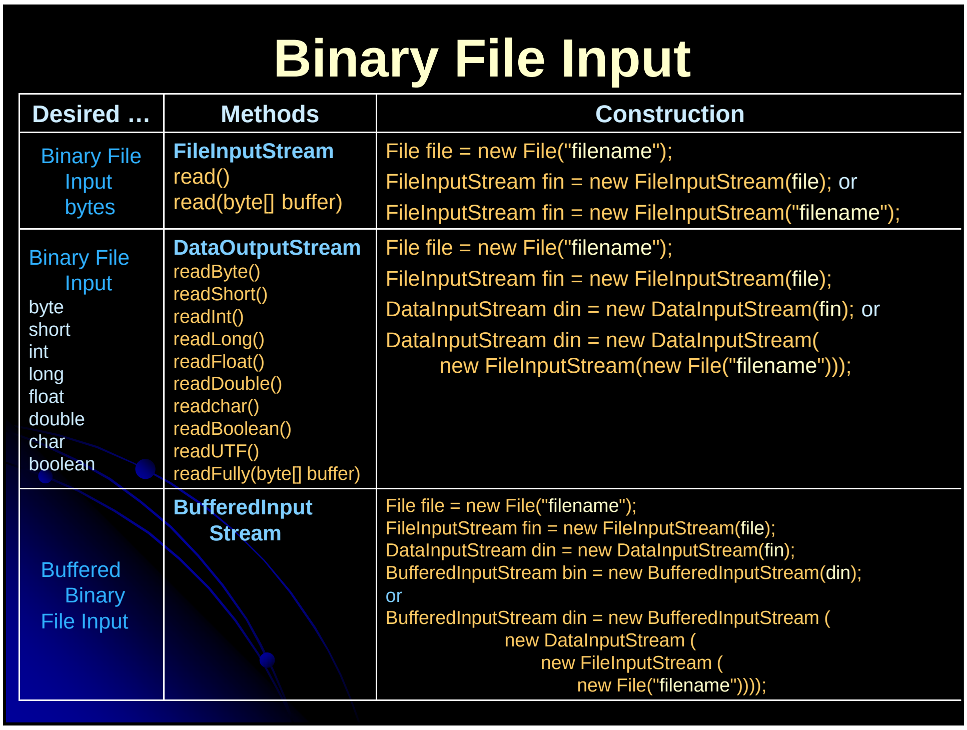
Binary File Output
byte
short
int
long
float
double
char
boolean
	DataOutputStream
	writeByte(byte)
	writeShort(short)
	writeInt(int)
	writeLong(long)
	writeFloat(float)
	writeDouble(double)
	writechar(char)
	writeBoolean(boolean)
	writeUTF(string)
	writeBytes(string)
	writeChars(string)
		File file = new File("filename");
		FileOutputStream fout = new FileOutputStream(file);
		DataOutputStream dout = new DataOutputStream(fout); or
		DataOutputStream dout = new DataOutputStream(
		new FileOutputStream(new File("filename")));

Buffered Binary File Output	**BufferedOutput Stream**	File file = new File("filename");
		FileOutputStream fout = new FileOutputStream(file);
		DataOutputStream dout = new DataOutputStream(fout);
		BufferedOutputStream bout = new BufferedOutputStream(dout);
		or
		BufferedOutputStream dout = new BufferedOutputStream (
		new DataOutputStream (new FileOutputStream (new
		File("filename"))));



BinaryFileOutput, Example

```
import java.io.*;

public class BinaryFileOutput {

  public static void main(String[] args) {
    int[] primes = { 1, 2, 3, 4, 5, 11, 17, 19 };
    DataOutputStream out = null;

    try {
      out = new DataOutputStream(
              new FileOutputStream("primes.dat"));

      for(int i=0; i<primes.length; i++) {
        out.writeInt(primes[i]);
      }
      out.close();
    } catch(IOException ioe) {
       System.out.println("IO problem: " + ioe);
       ioe.printStackTrace();
    }
  }
}
```



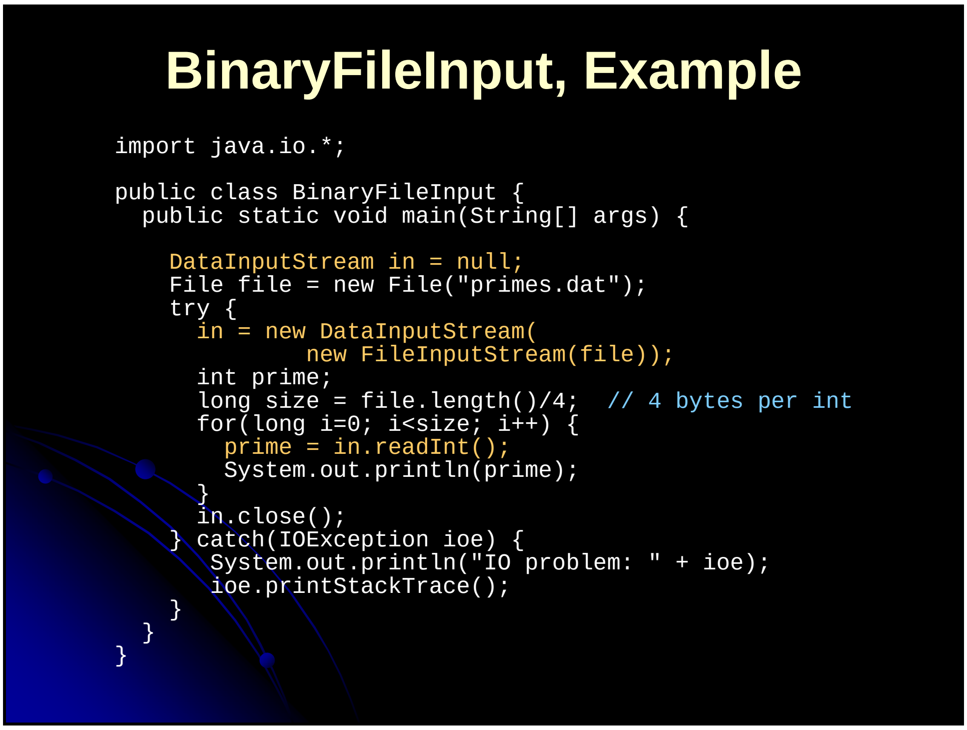
Binary File Input

Desired …	Methods	Construction

Binary File Input bytes	**FileInputStream**
	read()
	read(byte[] buffer)
		File file = new File("filename");
		FileInputStream fin = new FileInputStream(file); or
		FileInputStream fin = new FileInputStream("filename");

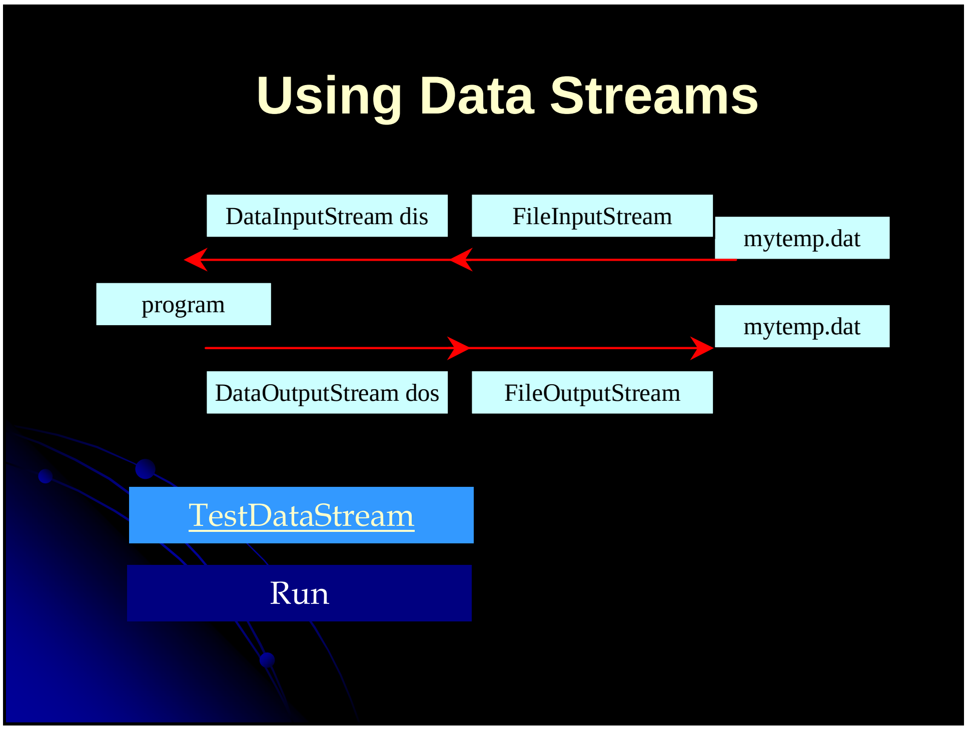
Binary File Input
byte
short
int
long
float
double
char
boolean
	DataOutputStream
	readByte()
	readShort()
	readInt()
	readLong()
	readFloat()
	readDouble()
	readchar()
	readBoolean()
	readUTF()
	readFully(byte[] buffer)
		File file = new File("filename");
		FileInputStream fin = new FileInputStream(file);
		DataInputStream din = new DataInputStream(fin); or
		DataInputStream din = new DataInputStream(
		new FileInputStream(new File("filename")));

Buffered Binary File Input	**BufferedInput Stream**	File file = new File("filename");
		FileInputStream fin = new FileInputStream(file);
		DataInputStream din = new DataInputStream(fin);
		BufferedInputStream bin = new BufferedInputStream(din);
		or
		BufferedInputStream din = new BufferedInputStream (
		new DataInputStream (
		new FileInputStream (
		new File("filename"))));

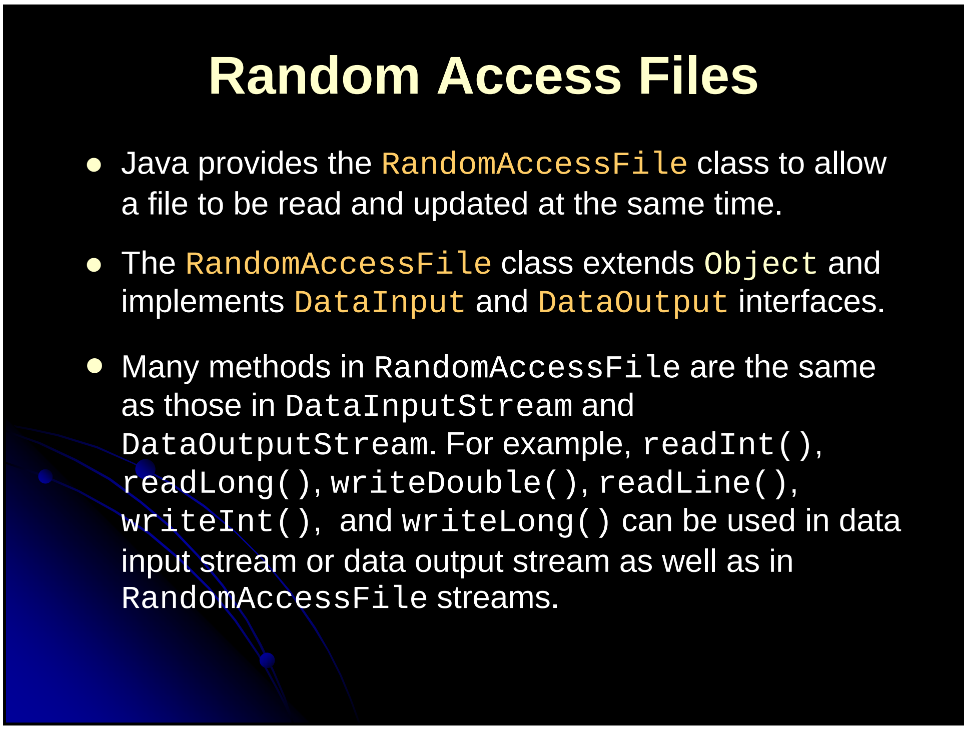


BinaryFileInput, Example

```
import java.io.*;

public class BinaryFileInput {
  public static void main(String[] args) {

    DataInputStream in = null;
    File file = new File("primes.dat");
    try {
      in = new DataInputStream(
              new FileInputStream(file));
      int prime;
      long size = file.length()/4;  // 4 bytes per int
      for(long i=0; i<size; i++) {
        prime = in.readInt();
        System.out.println(prime);
      }
      in.close();
    } catch(IOException ioe) {
       System.out.println("IO problem: " + ioe);
       ioe.printStackTrace();
    }
  }
}
```

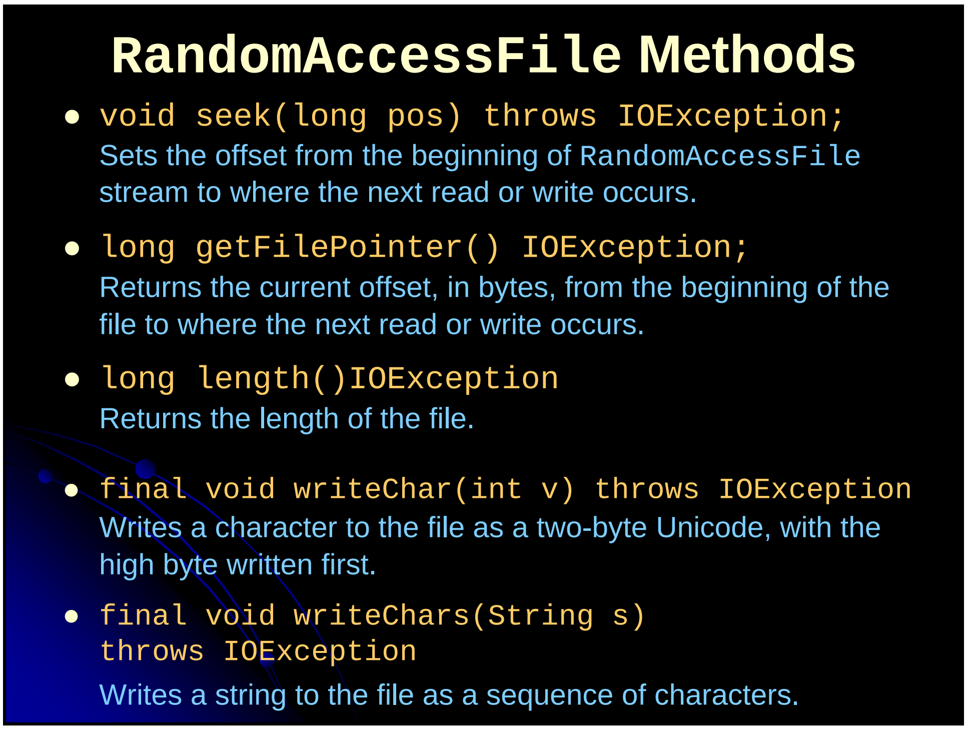


Using Data Streams



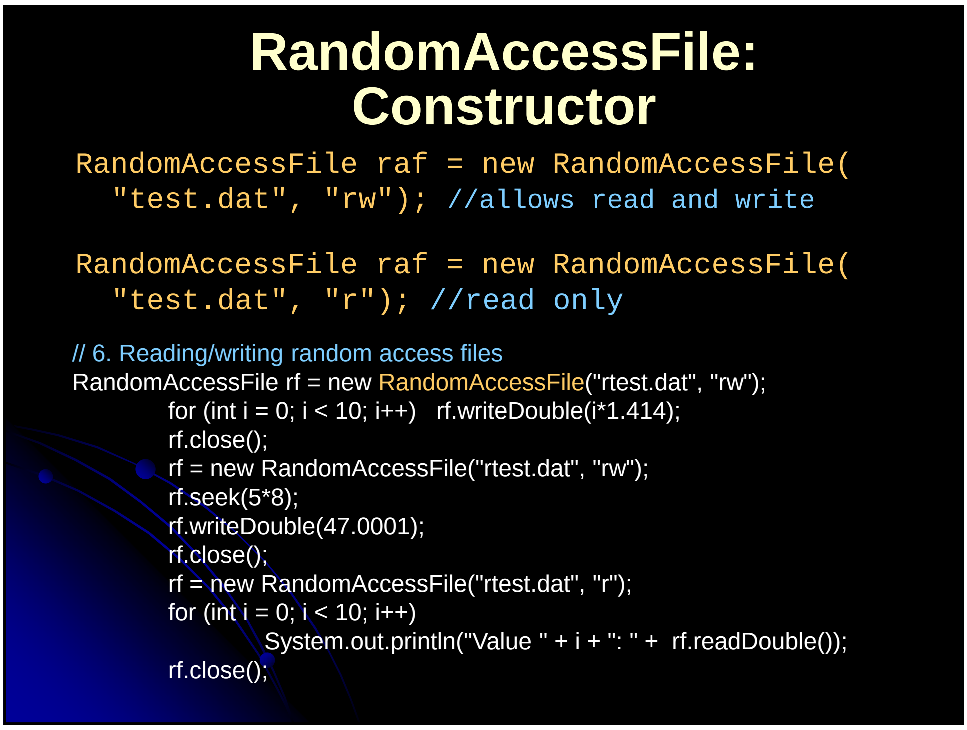
Random Access Files

- Java provides the RandomAccessFile class to allow a file to be read and updated at the same time.
- The RandomAccessFile class extends Object and implements DataInput and DataOutput interfaces.
- Many methods in RandomAccessFile are the same as those in DataInputStream and DataOutputStream. For example, readInt(), readLong(), writeDouble(), readLine(), writeInt(), and writeLong() can be used in data input stream or data output stream as well as in RandomAccessFile streams.



RandomAccessFile Methods

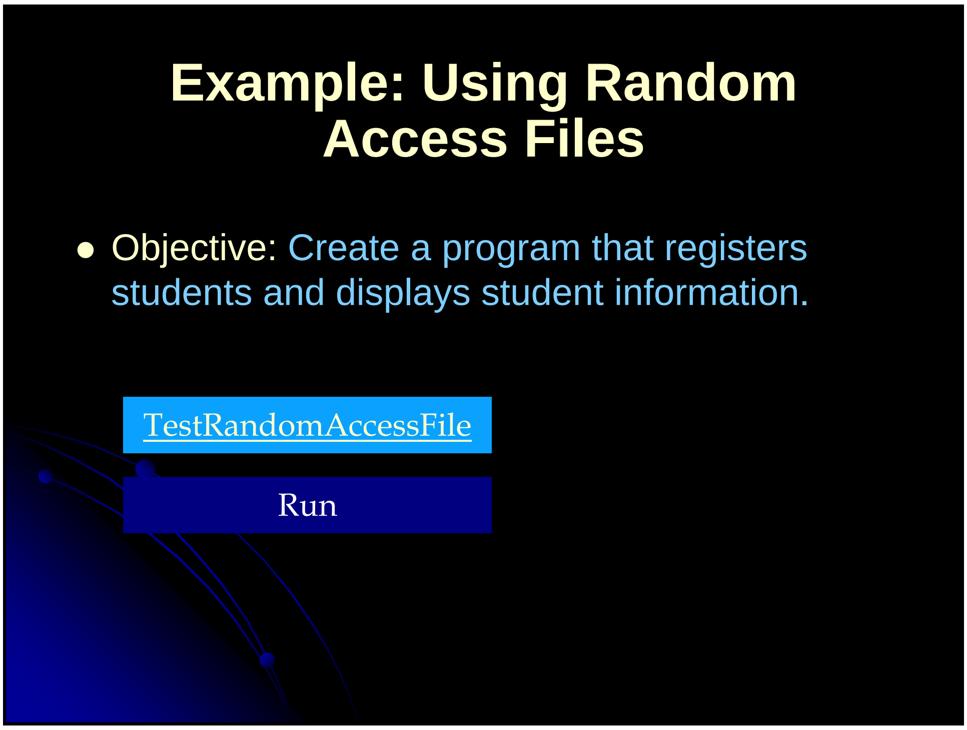
- `void seek(long pos) throws IOException;`
  Sets the offset from the beginning of `RandomAccessFile` stream to where the next read or write occurs.
- `long getFilePointer() IOException;`
  Returns the current offset, in bytes, from the beginning of the file to where the next read or write occurs.
- `long length()IOException`
  Returns the length of the file.
- `final void writeChar(int v) throws IOException`
  Writes a character to the file as a two-byte Unicode, with the high byte written first.
- `final void writeChars(String s) throws IOException`
  Writes a string to the file as a sequence of characters.



RandomAccessFile: Constructor

```
RandomAccessFile raf = new RandomAccessFile(
  "test.dat", "rw"); //allows read and write

RandomAccessFile raf = new RandomAccessFile(
  "test.dat", "r"); //read only
```

```
// 6. Reading/writing random access files
RandomAccessFile rf = new RandomAccessFile("rtest.dat", "rw");
        for (int i = 0; i < 10; i++)   rf.writeDouble(i*1.414);
        rf.close();
        rf = new RandomAccessFile("rtest.dat", "rw");
        rf.seek(5*8);
        rf.writeDouble(47.0001);
        rf.close();
        rf = new RandomAccessFile("rtest.dat", "r");
        for (int i = 0; i < 10; i++)
                System.out.println("Value " + i + ": " +  rf.readDouble());
        rf.close();
```

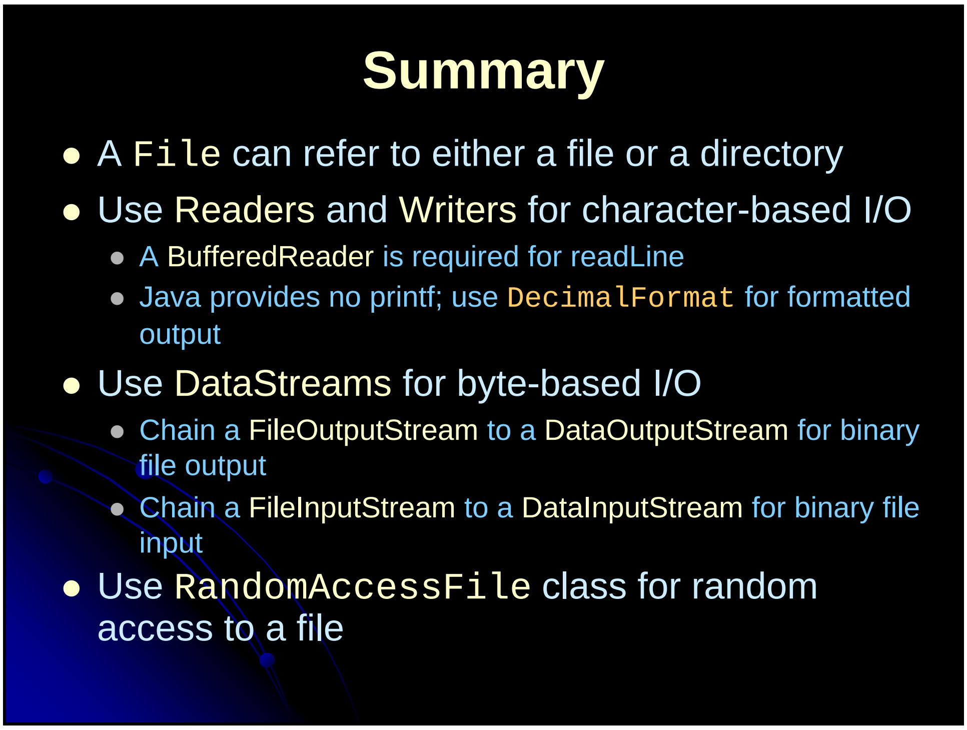


Example: Using Random Access Files

- Objective: Create a program that registers students and displays student information.

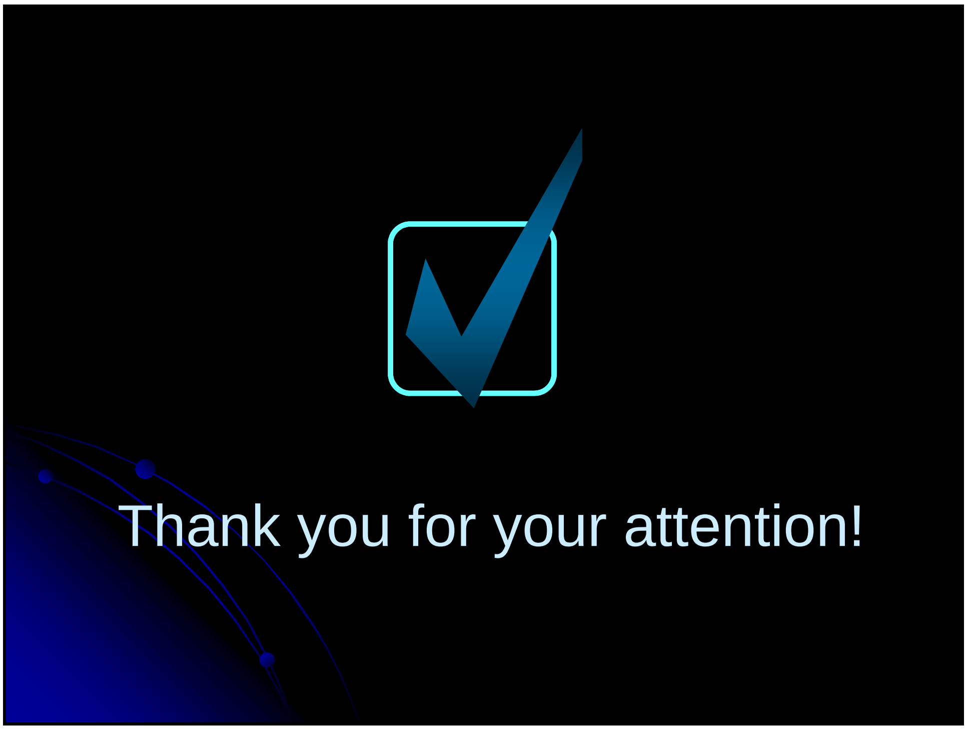
TestRandomAccessFile

Run

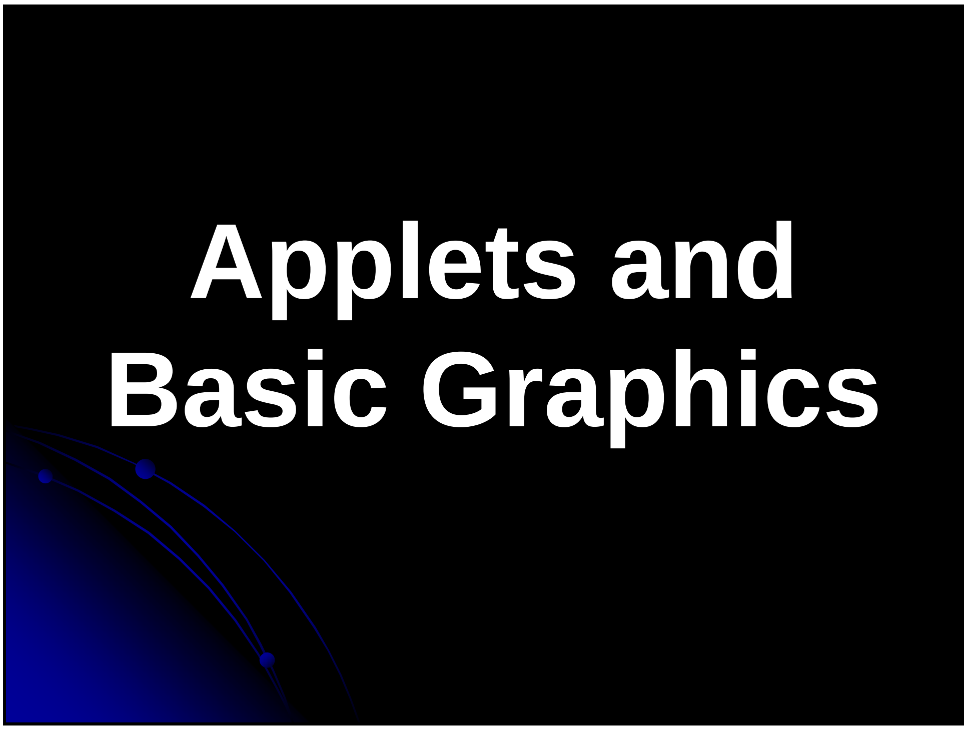


Summary

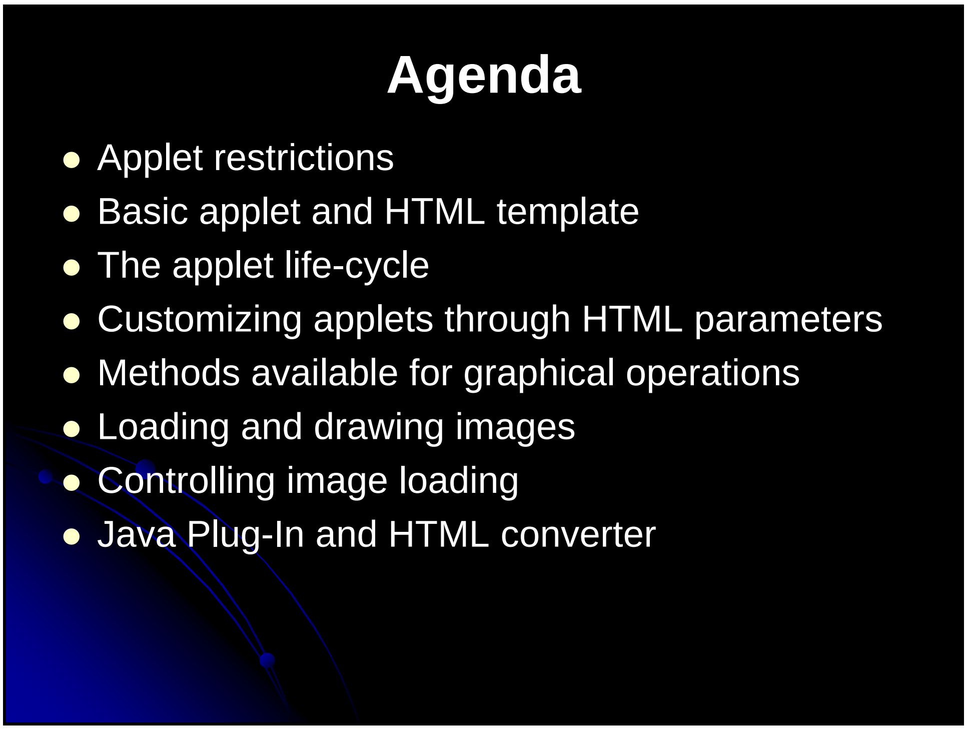
- A `File` can refer to either a file or a directory
- Use Readers and Writers for character-based I/O
  - A BufferedReader is required for readLine
  - Java provides no printf; use `DecimalFormat` for formatted output
- Use DataStreams for byte-based I/O
  - Chain a FileOutputStream to a DataOutputStream for binary file output
  - Chain a FileInputStream to a DataInputStream for binary file input
- Use `RandomAccessFile` class for random access to a file



Thank you for your attention!

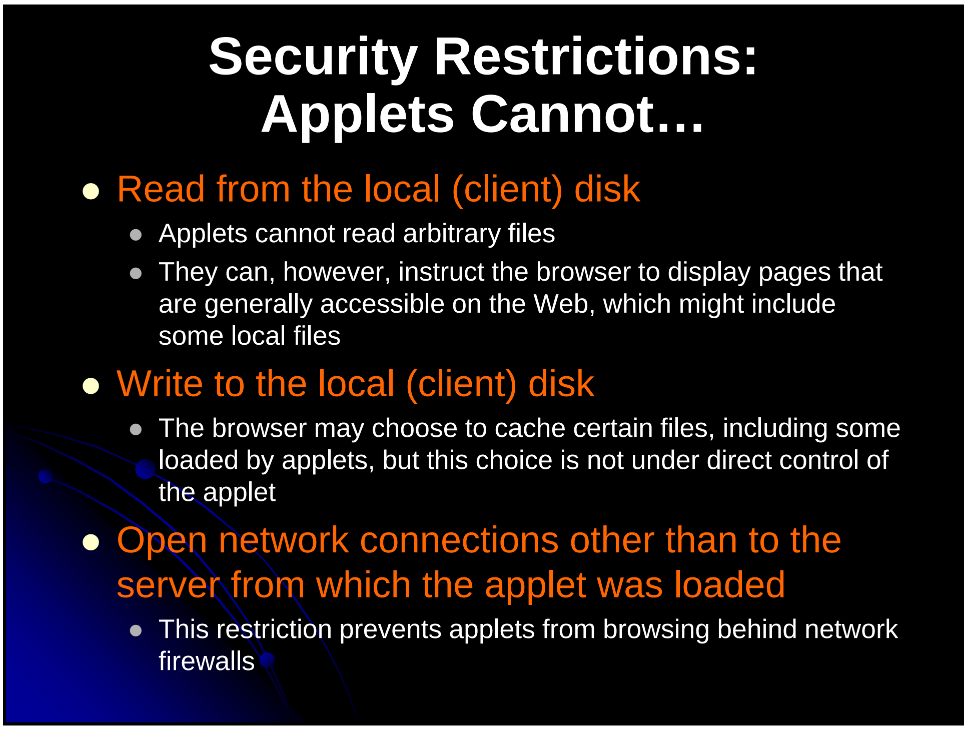


Applets and Basic Graphics



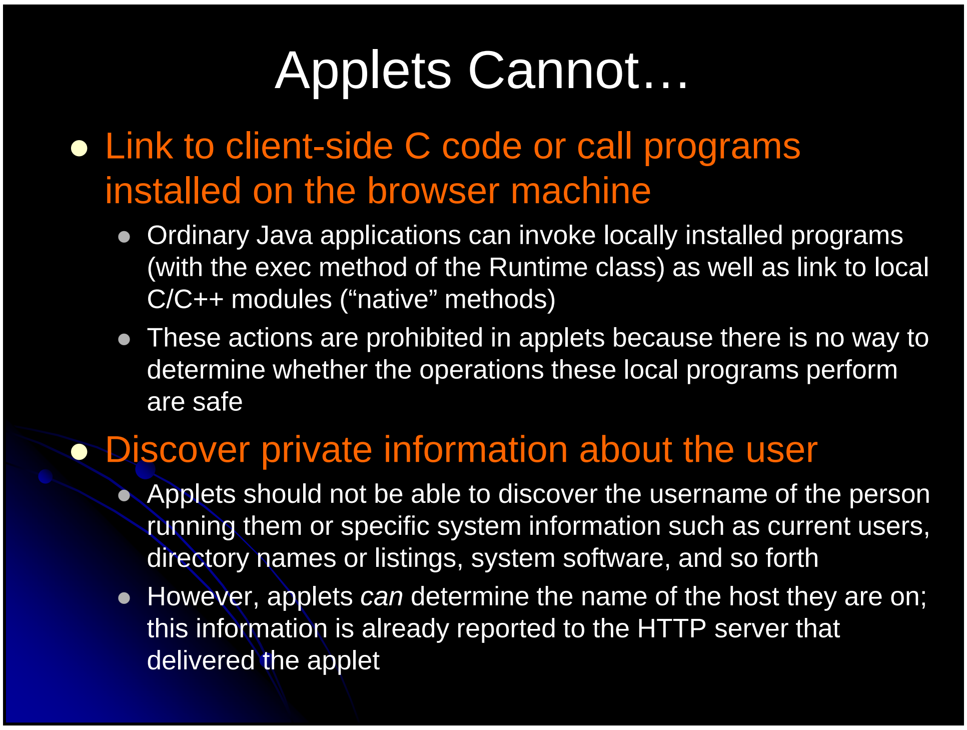
Agenda

- Applet restrictions
- Basic applet and HTML template
- The applet life-cycle
- Customizing applets through HTML parameters
- Methods available for graphical operations
- Loading and drawing images
- Controlling image loading
- Java Plug-In and HTML converter



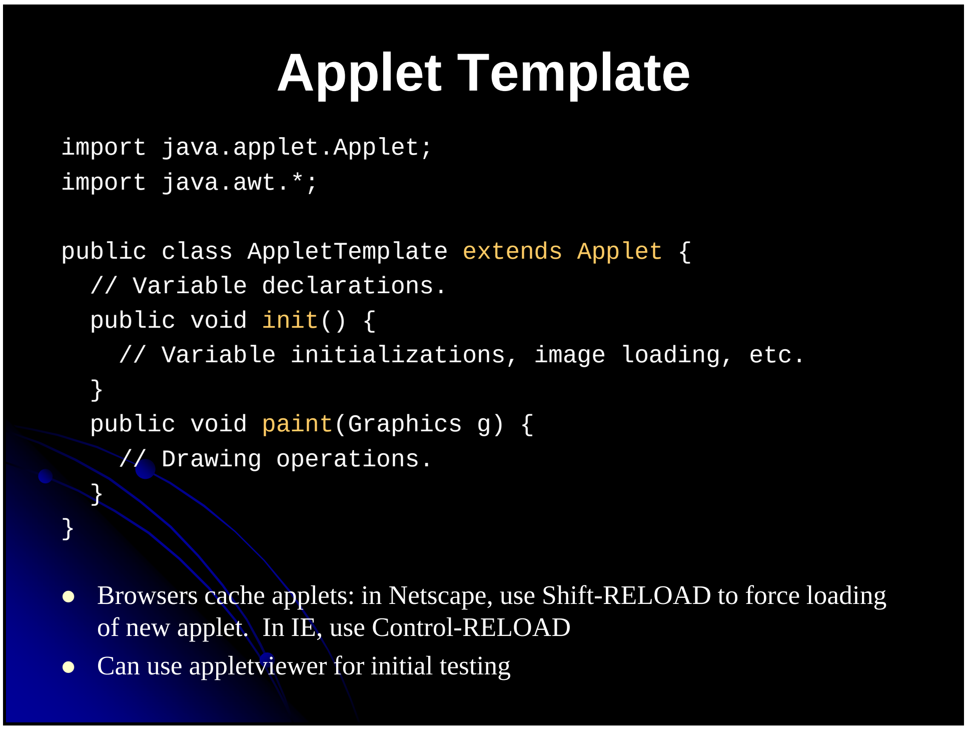
Security Restrictions: Applets Cannot…

- Read from the local (client) disk
  - Applets cannot read arbitrary files
  - They can, however, instruct the browser to display pages that are generally accessible on the Web, which might include some local files
- Write to the local (client) disk
  - The browser may choose to cache certain files, including some loaded by applets, but this choice is not under direct control of the applet
- Open network connections other than to the server from which the applet was loaded
  - This restriction prevents applets from browsing behind network firewalls



Applets Cannot…

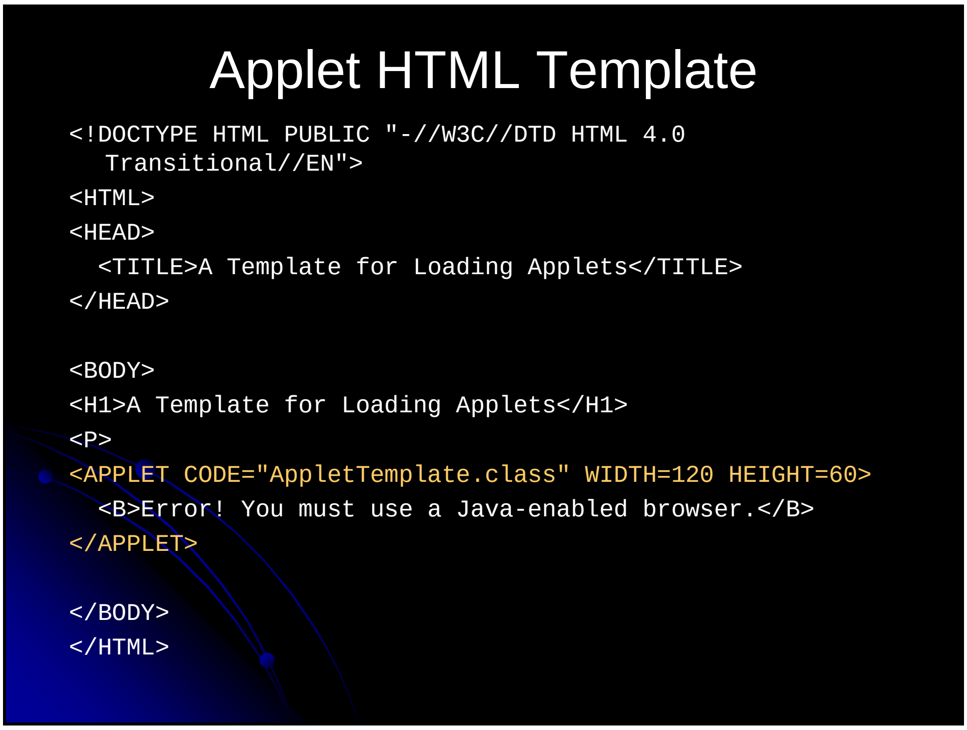
- Link to client-side C code or call programs installed on the browser machine
  - Ordinary Java applications can invoke locally installed programs (with the exec method of the Runtime class) as well as link to local C/C++ modules (“native” methods)
  - These actions are prohibited in applets because there is no way to determine whether the operations these local programs perform are safe
- Discover private information about the user
  - Applets should not be able to discover the username of the person running them or specific system information such as current users, directory names or listings, system software, and so forth
  - However, applets *can* determine the name of the host they are on; this information is already reported to the HTTP server that delivered the applet



Applet Template

```
import java.applet.Applet;
import java.awt.*;

public class AppletTemplate extends Applet {
  // Variable declarations.
  public void init() {
    // Variable initializations, image loading, etc.
  }
  public void paint(Graphics g) {
    // Drawing operations.
  }
}
```

- Browsers cache applets: in Netscape, use Shift-RELOAD to force loading of new applet. In IE, use Control-RELOAD
- Can use appletviewer for initial testing



Applet HTML Template

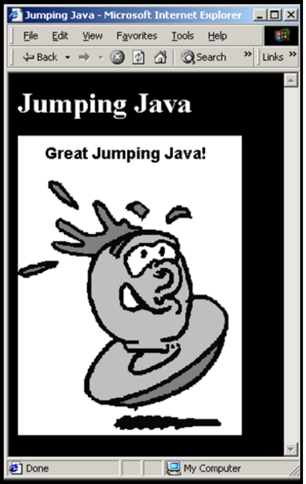
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
   Transitional//EN">
<HTML>
<HEAD>
  <TITLE>A Template for Loading Applets</TITLE>
</HEAD>

<BODY>
<H1>A Template for Loading Applets</H1>
<P>
<APPLET CODE="AppletTemplate.class" WIDTH=120 HEIGHT=60>
  <B>Error! You must use a Java-enabled browser.</B>
</APPLET>

</BODY>
</HTML>
```

Applet Example

```
import java.applet.Applet;
import java.awt.*;

/** An applet that draws an image. */

public class JavaJump extends Applet {
  private Image jumpingJava; // Instance var declarations here

  public void init() {       // Initializations here
    setBackground(Color.white);
    setFont(new Font("SansSerif", Font.BOLD, 18));
    jumpingJava = getImage(getDocumentBase(),
                           "images/Jumping-Java.gif");
    add(new Label("Great Jumping Java!"));
    System.out.println("Yow! I'm jiving with Java.");
  }

  public void paint(Graphics g) {  // Drawing here
    g.drawImage(jumpingJava, 0, 50, this);
  }
}
```

Applet Example: Result

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Jumping Java</TITLE>
</HEAD>
<BODY BGCOLOR="BLACK" TEXT="WHITE">
<H1>Jumping Java</H1>
<P>
<APPLET CODE="JavaJump.class"
        WIDTH=250
        HEIGHT=335>
<B>Sorry, this example requires Java.</B>
</APPLET>
</BODY>
</HTML>
```



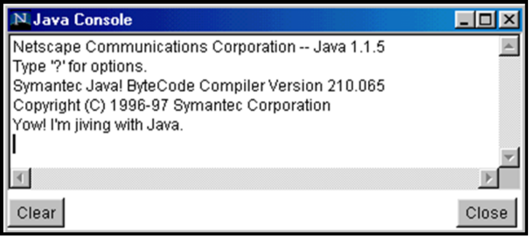
Jumping Java - Microsoft Internet Explorer

Jumping Java

Great Jumping Java!

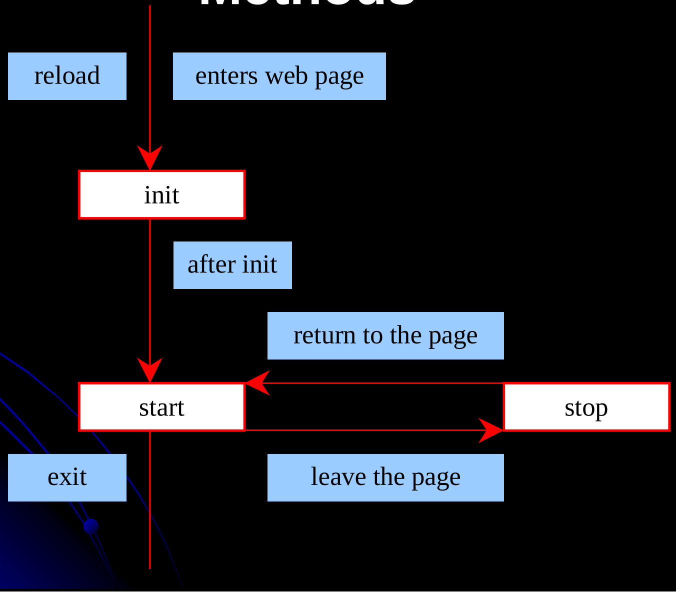
Debugging Applets: The Java Console

- Standard output (from System.out.println) is sent to the Java Console
  - Navigator: open from `Window` menu
  - Communicator: open from `Communicator` … `Tools`
  - IE 4: open from `View` menu (enable from `Tools` … `Internet Options` … `Advanced` screen)
  - IE 5/6 with Java plugin: go to Control Panel, click on Java Plugin, and select "Show Console" option.



```
Netscape Communications Corporation -- Java 1.1.5
Type '?' for options.
Symantec Java! ByteCode Compiler Version 210.065
Copyright (C) 1996-97 Symantec Corporation
Yow! I'm jiving with Java.
```

Browser Calling Applet Methods



reload

enters web page

init

after init

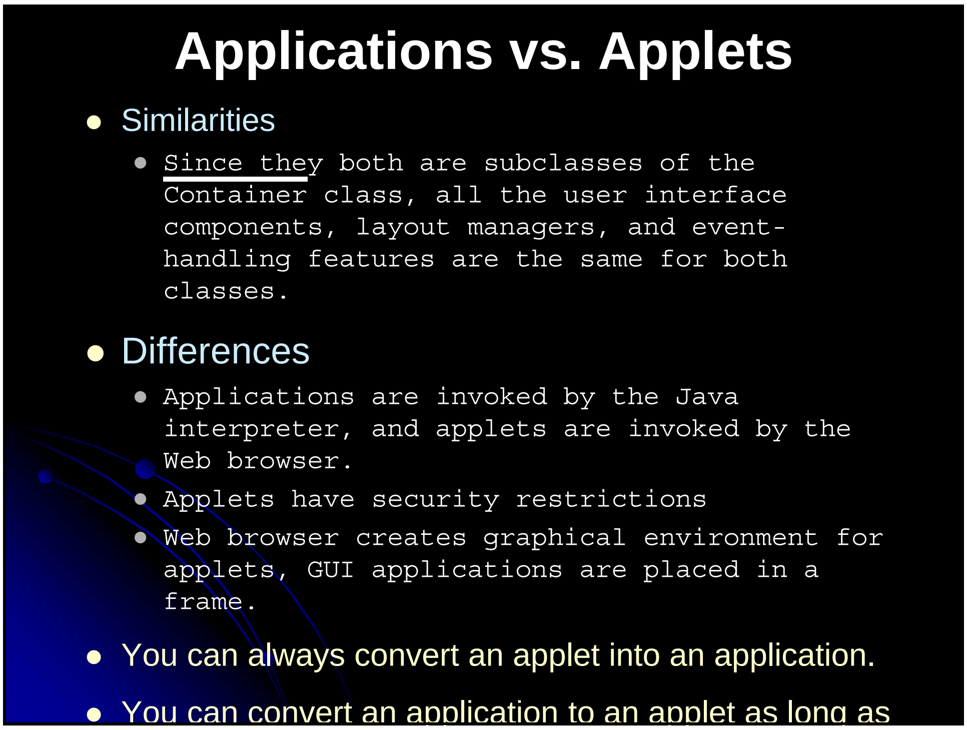
return to the page

start

stop

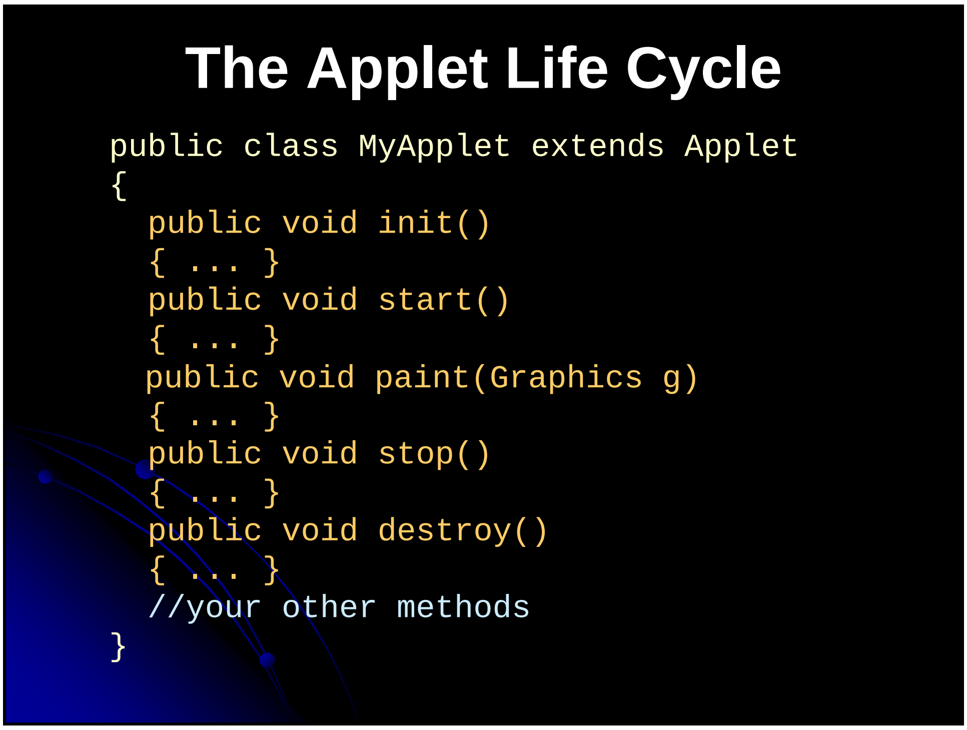
exit

leave the page



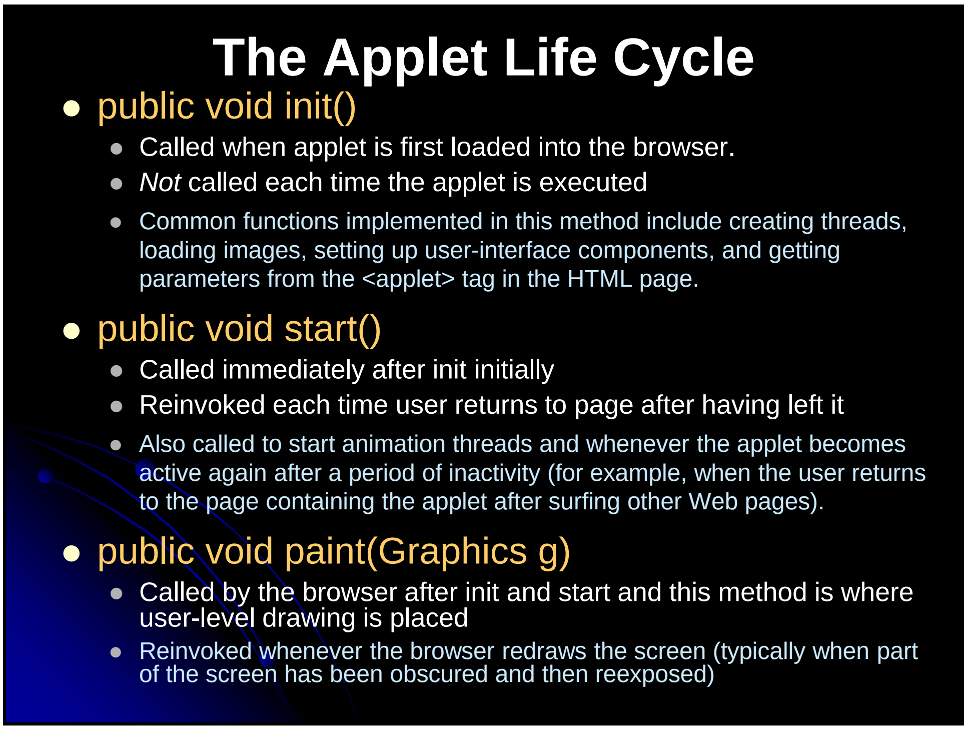
Applications vs. Applets

- Similarities
  - Since they both are subclasses of the Container class, all the user interface components, layout managers, and event-handling features are the same for both classes.
- Differences
  - Applications are invoked by the Java interpreter, and applets are invoked by the Web browser.
  - Applets have security restrictions
  - Web browser creates graphical environment for applets, GUI applications are placed in a frame.
- You can always convert an applet into an application.
- You can convert an application to an applet as long as



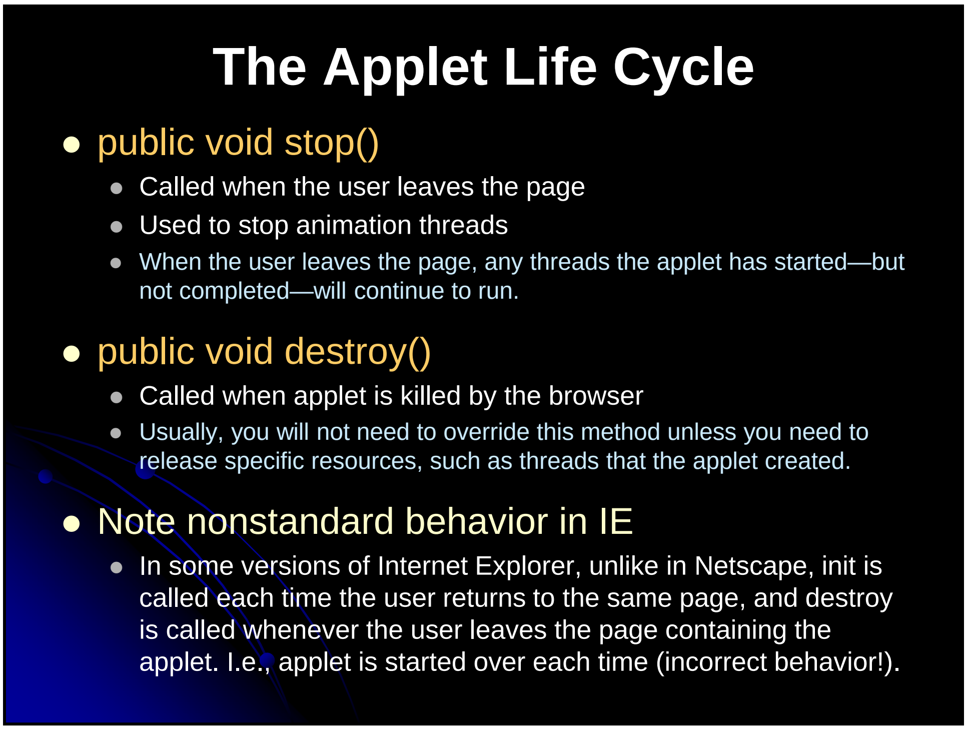
The Applet Life Cycle

```
public class MyApplet extends Applet
{
  public void init()
  { ... }
  public void start()
  { ... }
  public void paint(Graphics g)
  { ... }
  public void stop()
  { ... }
  public void destroy()
  { ... }
  //your other methods
}
```



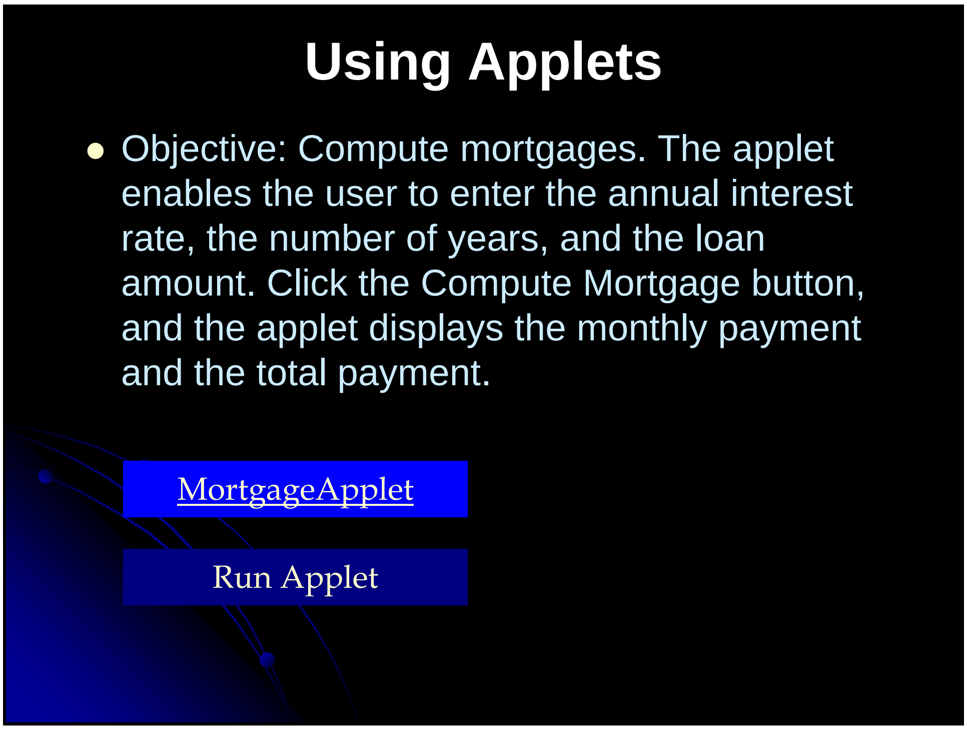
The Applet Life Cycle

- public void init()
  - Called when applet is first loaded into the browser.
  - *Not* called each time the applet is executed
  - Common functions implemented in this method include creating threads, loading images, setting up user-interface components, and getting parameters from the <applet> tag in the HTML page.
- public void start()
  - Called immediately after init initially
  - Reinvoked each time user returns to page after having left it
  - Also called to start animation threads and whenever the applet becomes active again after a period of inactivity (for example, when the user returns to the page containing the applet after surfing other Web pages).
- public void paint(Graphics g)
  - Called by the browser after init and start and this method is where user-level drawing is placed
  - Reinvoked whenever the browser redraws the screen (typically when part of the screen has been obscured and then reexposed)



The Applet Life Cycle

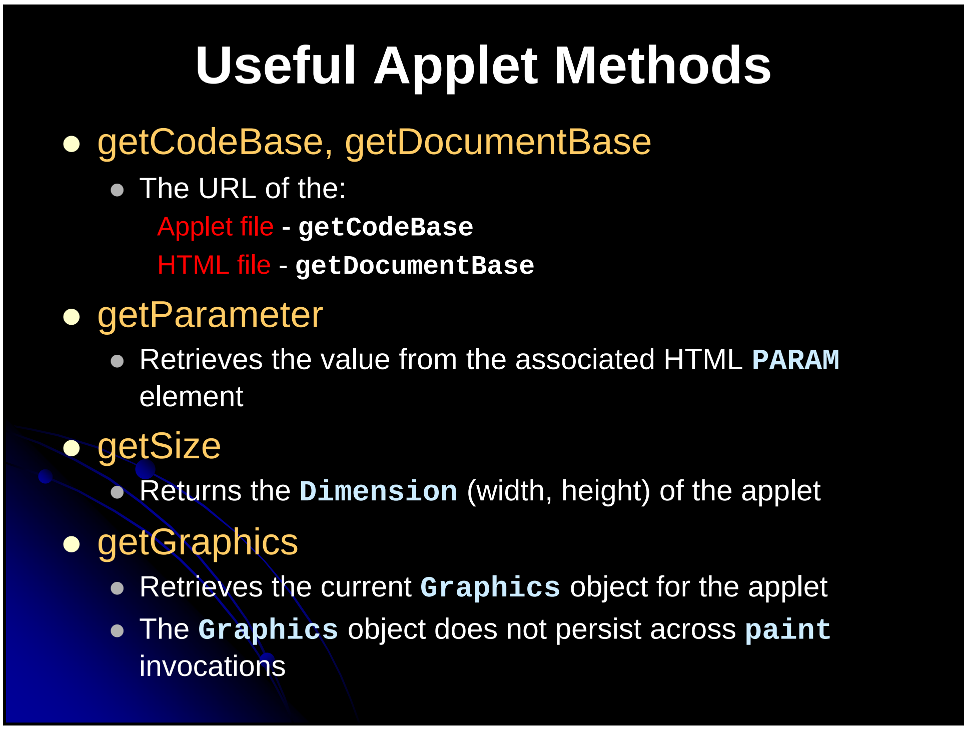
- public void stop()
  - Called when the user leaves the page
  - Used to stop animation threads
  - When the user leaves the page, any threads the applet has started—but not completed—will continue to run.
- public void destroy()
  - Called when applet is killed by the browser
  - Usually, you will not need to override this method unless you need to release specific resources, such as threads that the applet created.
- Note nonstandard behavior in IE
  - In some versions of Internet Explorer, unlike in Netscape, init is called each time the user returns to the same page, and destroy is called whenever the user leaves the page containing the applet. I.e., applet is started over each time (incorrect behavior!).



Using Applets

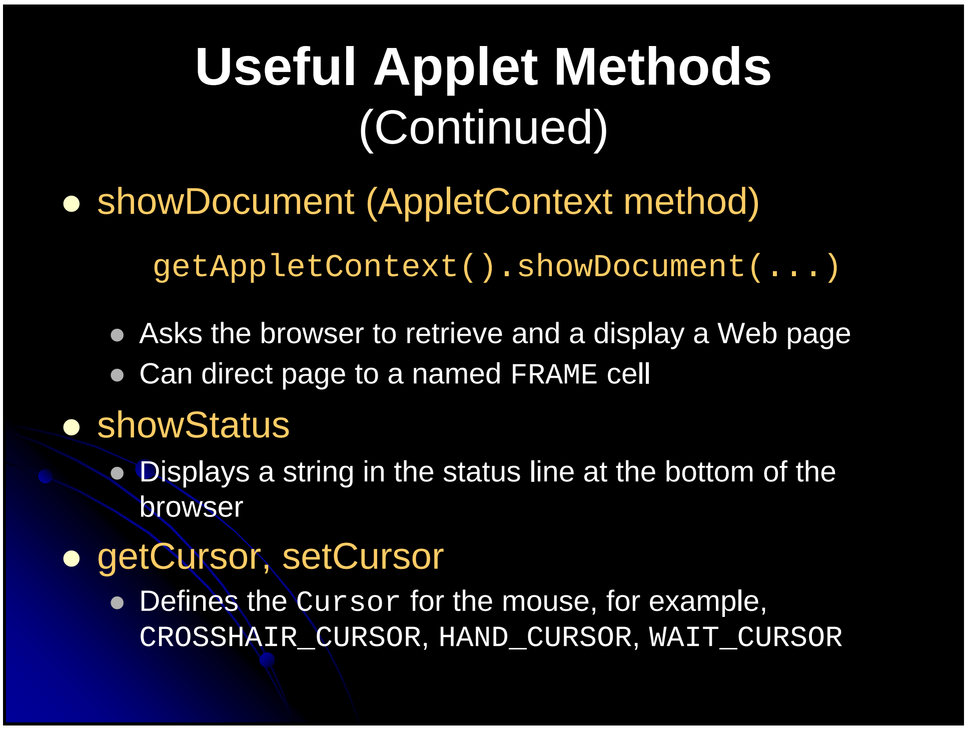
- Objective: Compute mortgages. The applet enables the user to enter the annual interest rate, the number of years, and the loan amount. Click the Compute Mortgage button, and the applet displays the monthly payment and the total payment.

MortgageApplet
Run Applet



Useful Applet Methods

- getCodeBase, getDocumentBase
  - The URL of the:
    Applet file - **`getCodeBase`**
    HTML file - **`getDocumentBase`**
- getParameter
  - Retrieves the value from the associated HTML **`PARAM`** element
- getSize
  - Returns the **`Dimension`** (width, height) of the applet
- getGraphics
  - Retrieves the current **`Graphics`** object for the applet
  - The **`Graphics`** object does not persist across **`paint`** invocations

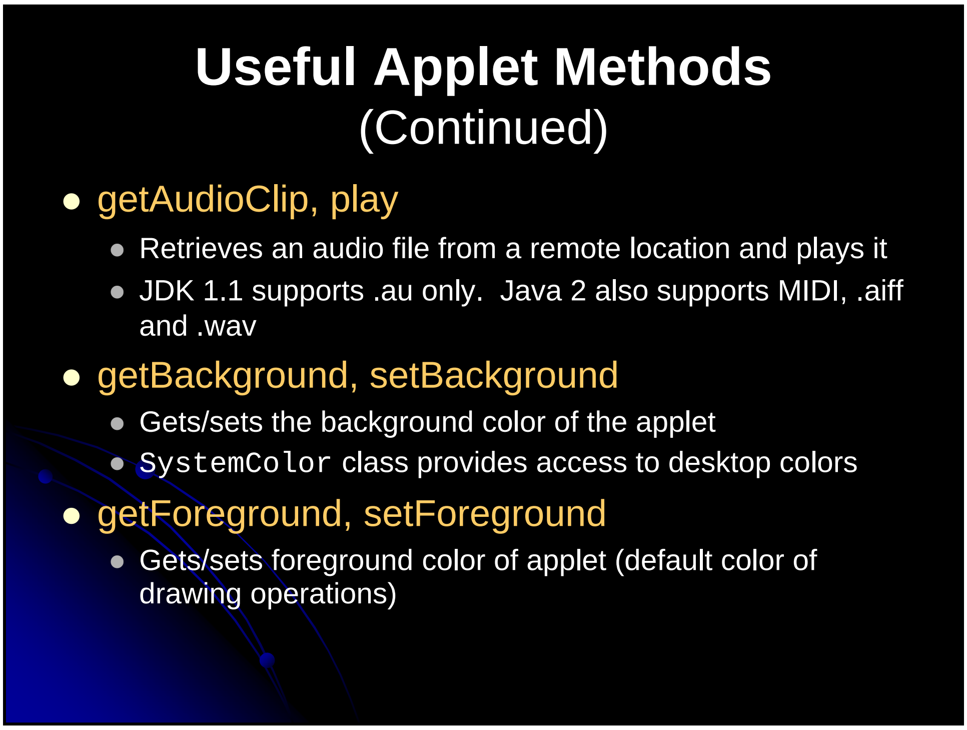


Useful Applet Methods
(Continued)

- showDocument (AppletContext method)

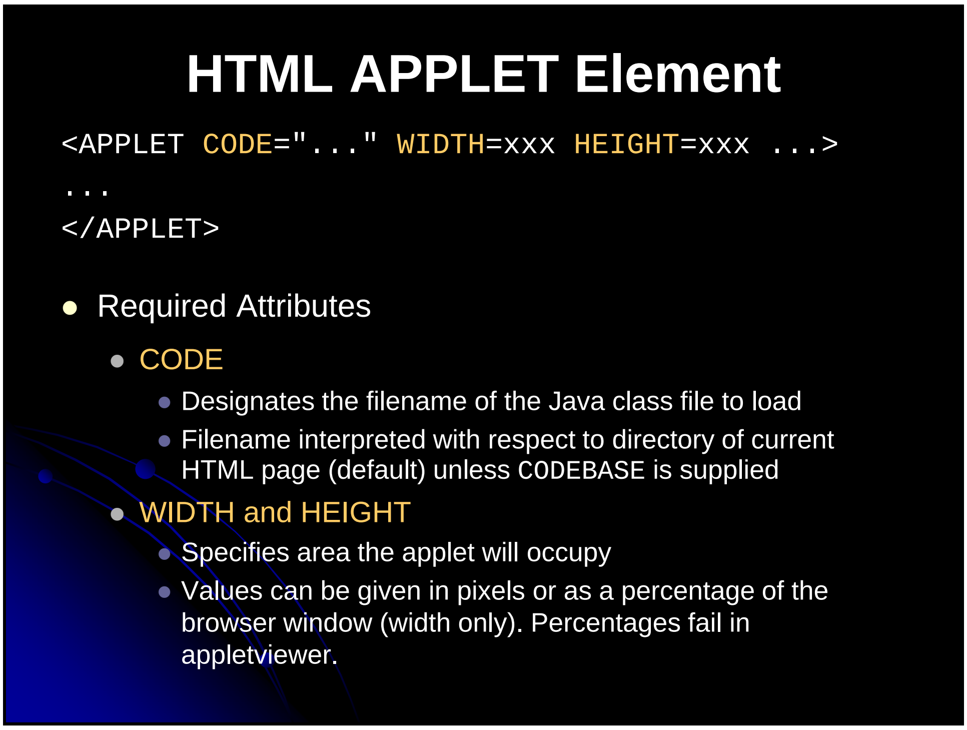
  ```
  getAppletContext().showDocument(...)
  ```

  - Asks the browser to retrieve and a display a Web page
  - Can direct page to a named `FRAME` cell
- showStatus
  - Displays a string in the status line at the bottom of the browser
- getCursor, setCursor
  - Defines the `Cursor` for the mouse, for example, `CROSSHAIR_CURSOR`, `HAND_CURSOR`, `WAIT_CURSOR`



Useful Applet Methods
(Continued)

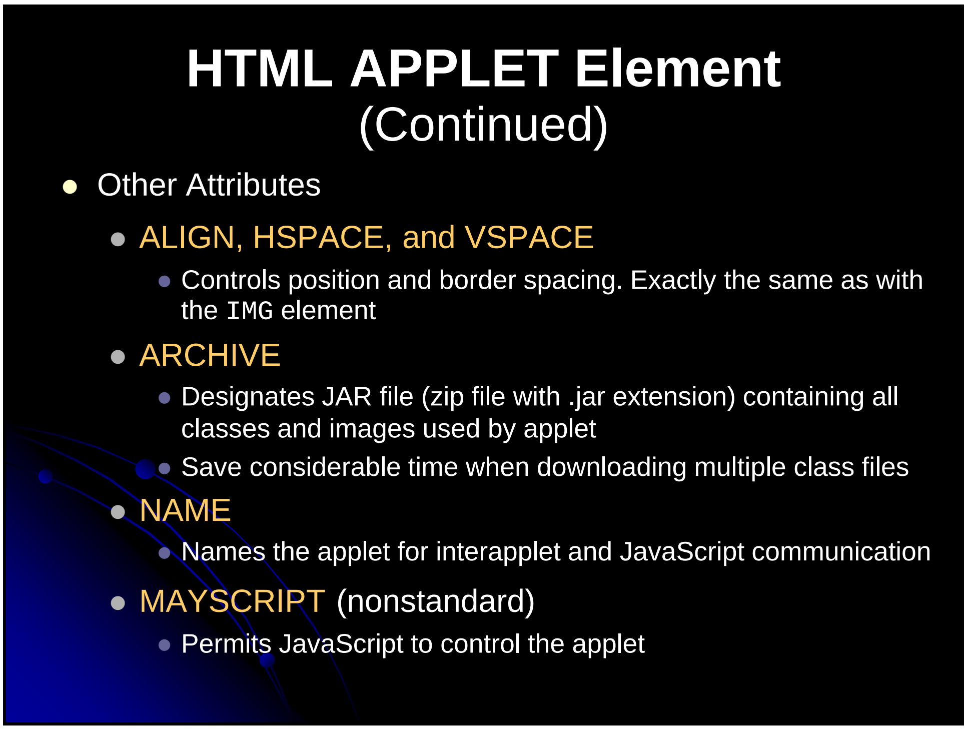
- getAudioClip, play
  - Retrieves an audio file from a remote location and plays it
  - JDK 1.1 supports .au only. Java 2 also supports MIDI, .aiff and .wav
- getBackground, setBackground
  - Gets/sets the background color of the applet
  - `SystemColor` class provides access to desktop colors
- getForeground, setForeground
  - Gets/sets foreground color of applet (default color of drawing operations)



HTML APPLET Element

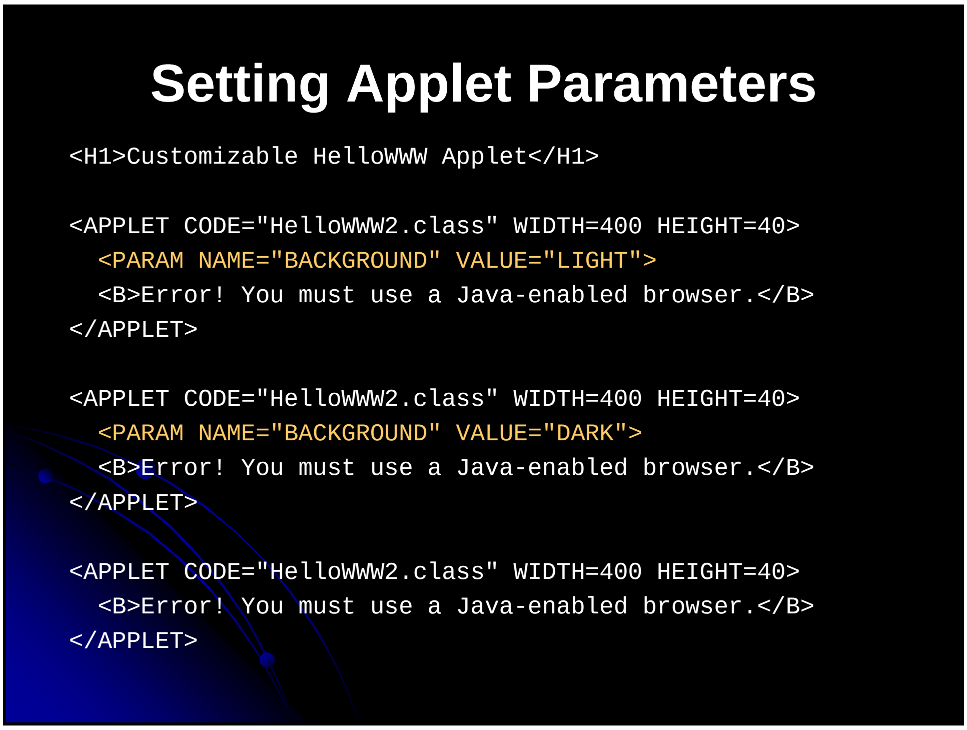
```
<APPLET CODE="..." WIDTH=xxx HEIGHT=xxx ...>
...
</APPLET>
```

- Required Attributes
  - CODE
    - Designates the filename of the Java class file to load
    - Filename interpreted with respect to directory of current HTML page (default) unless `CODEBASE` is supplied
  - WIDTH and HEIGHT
    - Specifies area the applet will occupy
    - Values can be given in pixels or as a percentage of the browser window (width only). Percentages fail in appletviewer.



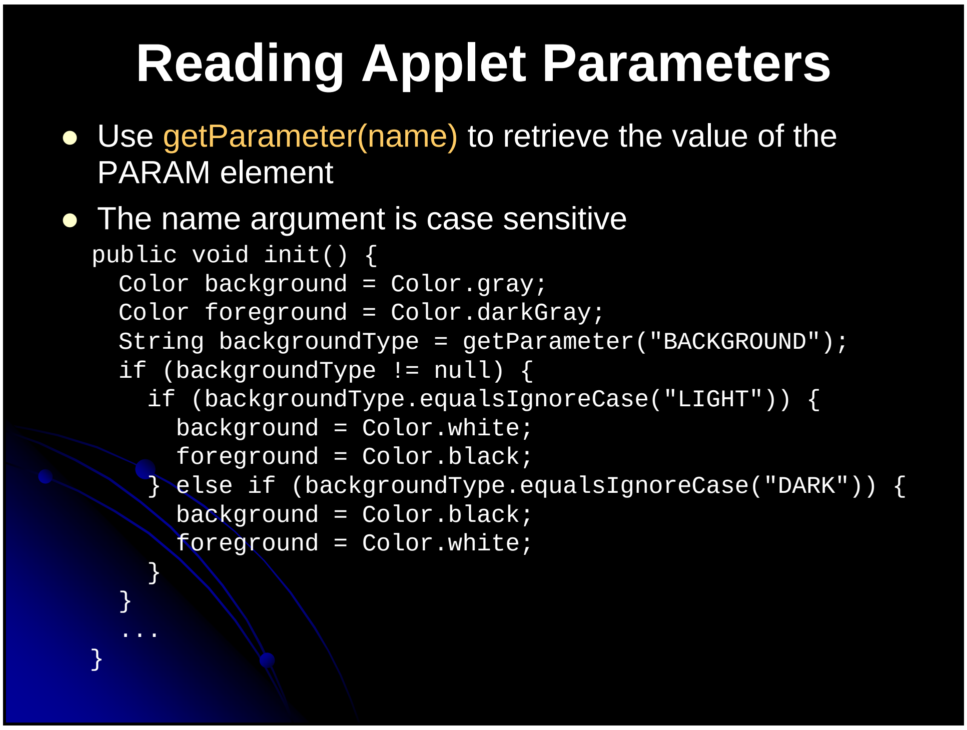
HTML APPLET Element (Continued)

- Other Attributes
  - ALIGN, HSPACE, and VSPACE
    - Controls position and border spacing. Exactly the same as with the `IMG` element
  - ARCHIVE
    - Designates JAR file (zip file with .jar extension) containing all classes and images used by applet
    - Save considerable time when downloading multiple class files
  - NAME
    - Names the applet for interapplet and JavaScript communication
  - MAYSCRIPT (nonstandard)
    - Permits JavaScript to control the applet



Setting Applet Parameters

```
<H1>Customizable HelloWWW Applet</H1>

<APPLET CODE="HelloWWW2.class" WIDTH=400 HEIGHT=40>
  <PARAM NAME="BACKGROUND" VALUE="LIGHT">
  <B>Error! You must use a Java-enabled browser.</B>
</APPLET>

<APPLET CODE="HelloWWW2.class" WIDTH=400 HEIGHT=40>
  <PARAM NAME="BACKGROUND" VALUE="DARK">
  <B>Error! You must use a Java-enabled browser.</B>
</APPLET>

<APPLET CODE="HelloWWW2.class" WIDTH=400 HEIGHT=40>
  <B>Error! You must use a Java-enabled browser.</B>
</APPLET>
```

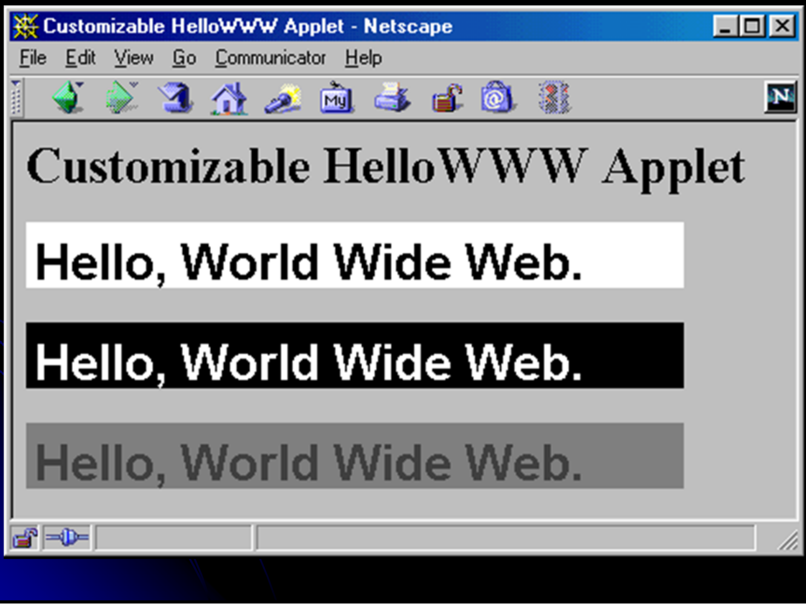


Reading Applet Parameters

- Use getParameter(name) to retrieve the value of the PARAM element
- The name argument is case sensitive

```
public void init() {
  Color background = Color.gray;
  Color foreground = Color.darkGray;
  String backgroundType = getParameter("BACKGROUND");
  if (backgroundType != null) {
    if (backgroundType.equalsIgnoreCase("LIGHT")) {
      background = Color.white;
      foreground = Color.black;
    } else if (backgroundType.equalsIgnoreCase("DARK")) {
      background = Color.black;
      foreground = Color.white;
    }
  }
  ...
}
```

Reading Applet Parameters: Result


Customizable HelloWWW Applet - Netscape
File Edit View Go Communicator Help

Customizable HelloWWW Applet

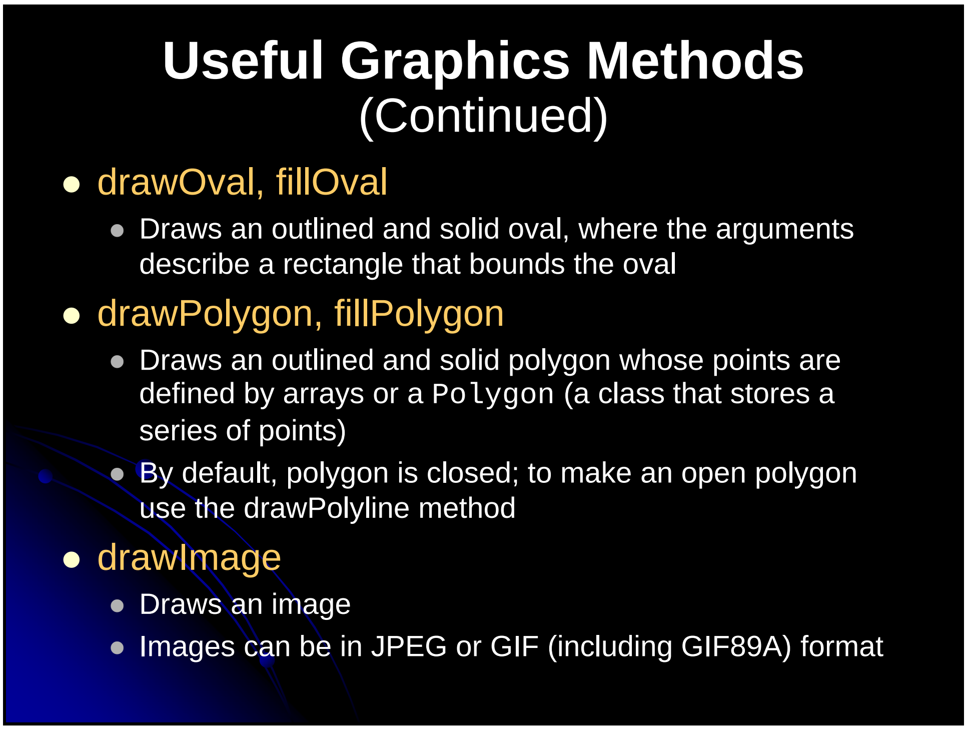
Hello, World Wide Web.

Hello, World Wide Web.

Hello, World Wide Web.

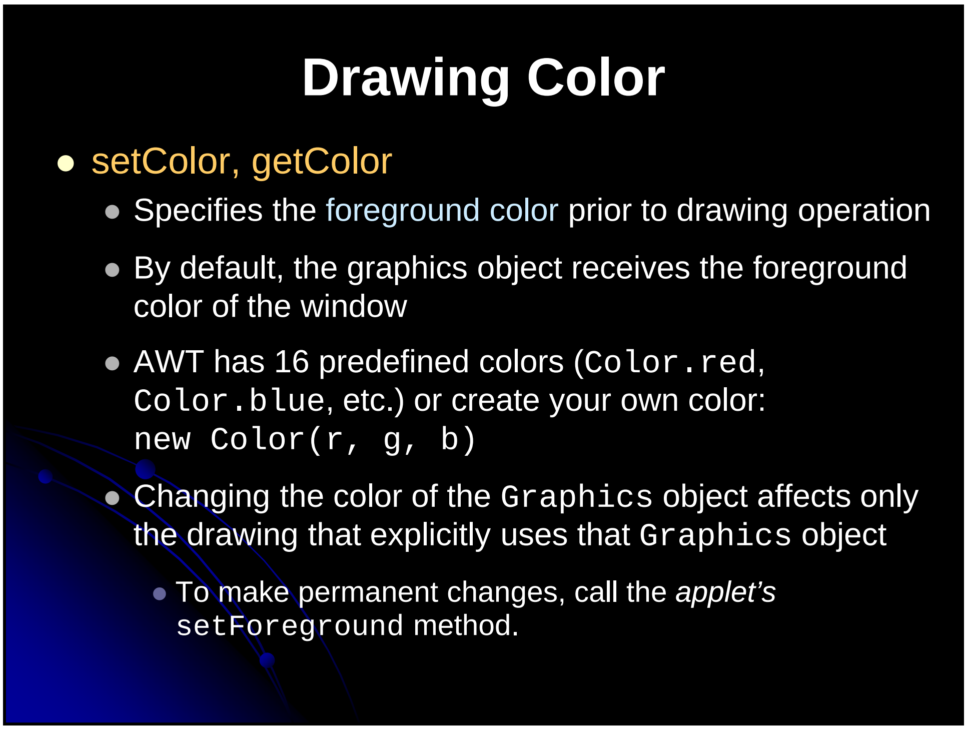
Useful Graphics Methods

- drawString(string, left, bottom)
  - Draws a string in the current font and color with the *bottom left* corner of the string at the specified location
  - One of the few methods where the y coordinate refers to the bottom of shape, not the top. But y values are still with respect to the *top left* corner of the applet window
- drawRect(left, top, width, height)
  - Draws the outline of a rectangle (1-pixel border) in the current color
- fillRect(left, top, width, height)
  - Draws a solid rectangle in the current color
- drawLine(x1, y1, x2, y2)
  - Draws a 1-pixel-thick line from (x1, y1) to (x2, y2)



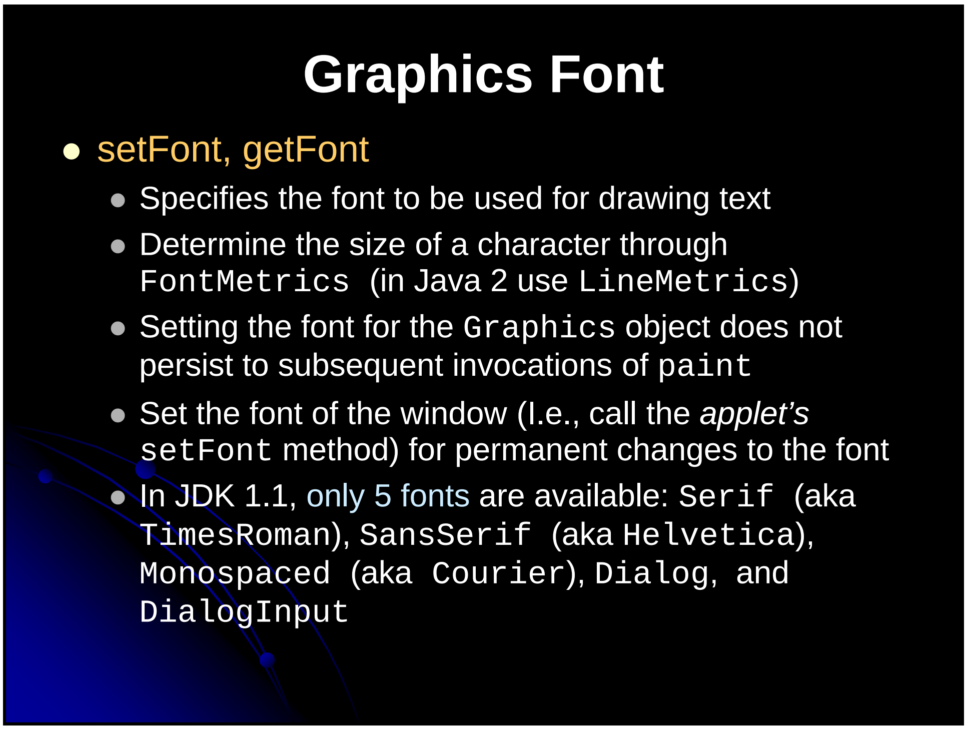
Useful Graphics Methods (Continued)

- drawOval, fillOval
  - Draws an outlined and solid oval, where the arguments describe a rectangle that bounds the oval
- drawPolygon, fillPolygon
  - Draws an outlined and solid polygon whose points are defined by arrays or a `Polygon` (a class that stores a series of points)
  - By default, polygon is closed; to make an open polygon use the drawPolyline method
- drawImage
  - Draws an image
  - Images can be in JPEG or GIF (including GIF89A) format



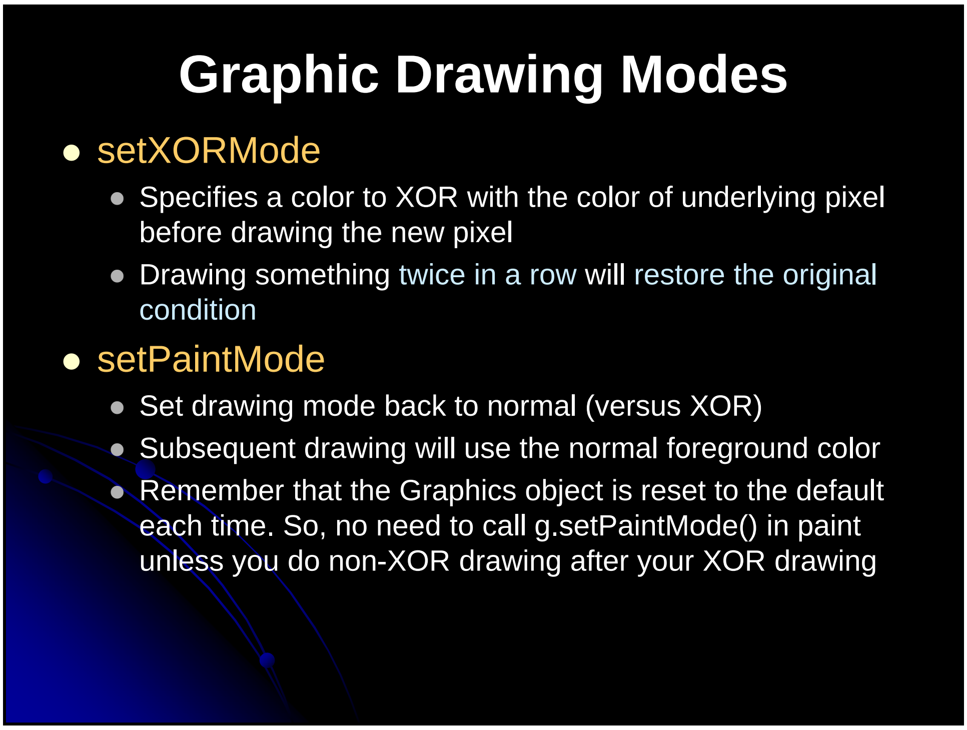
Drawing Color

- setColor, getColor
  - Specifies the foreground color prior to drawing operation
  - By default, the graphics object receives the foreground color of the window
  - AWT has 16 predefined colors (`Color.red`, `Color.blue`, etc.) or create your own color: `new Color(r, g, b)`
  - Changing the color of the `Graphics` object affects only the drawing that explicitly uses that `Graphics` object
    - To make permanent changes, call the *applet's* `setForeground` method.



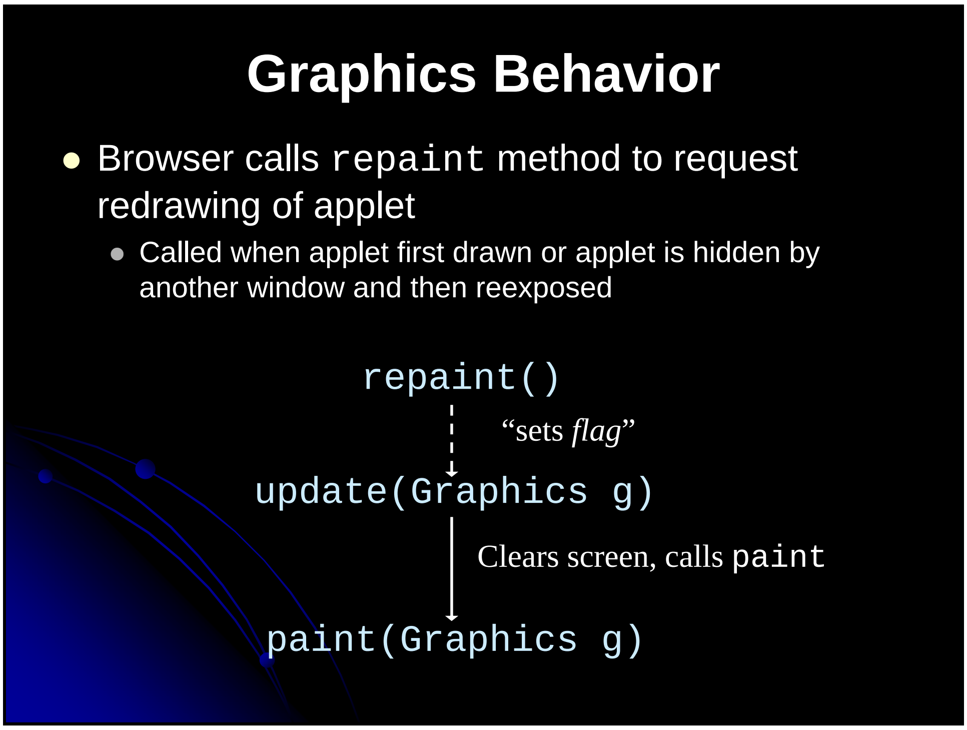
Graphics Font

- setFont, getFont
  - Specifies the font to be used for drawing text
  - Determine the size of a character through `FontMetrics` (in Java 2 use `LineMetrics`)
  - Setting the font for the `Graphics` object does not persist to subsequent invocations of `paint`
  - Set the font of the window (I.e., call the *applet's* `setFont` method) for permanent changes to the font
  - In JDK 1.1, only 5 fonts are available: `Serif` (aka `TimesRoman`), `SansSerif` (aka `Helvetica`), `Monospaced` (aka `Courier`), `Dialog`, and `DialogInput`



Graphic Drawing Modes

- setXORMode
  - Specifies a color to XOR with the color of underlying pixel before drawing the new pixel
  - Drawing something twice in a row will restore the original condition
- setPaintMode
  - Set drawing mode back to normal (versus XOR)
  - Subsequent drawing will use the normal foreground color
  - Remember that the Graphics object is reset to the default each time. So, no need to call g.setPaintMode() in paint unless you do non-XOR drawing after your XOR drawing



Graphics Behavior

- Browser calls `repaint` method to request redrawing of applet
  - Called when applet first drawn or applet is hidden by another window and then reexposed

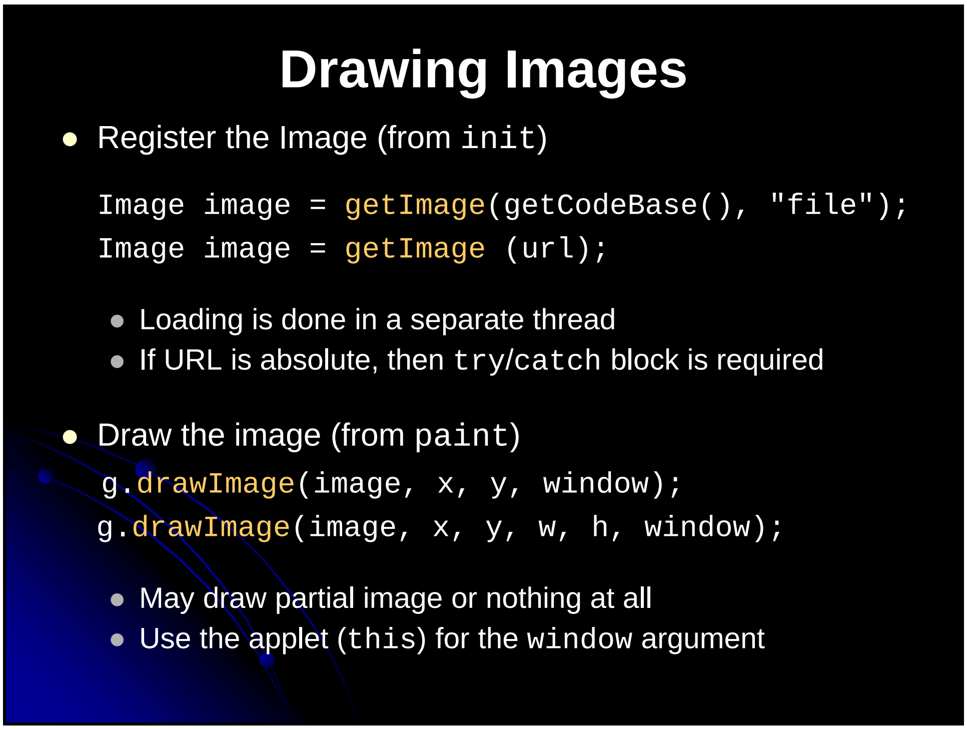
```
repaint()
```

"sets *flag*"

```
update(Graphics g)
```

Clears screen, calls `paint`

```
paint(Graphics g)
```



Drawing Images

- Register the Image (from `init`)

```
Image image = getImage(getCodeBase(), "file");
Image image = getImage (url);
```

  - Loading is done in a separate thread
  - If URL is absolute, then `try/catch` block is required
- Draw the image (from `paint`)

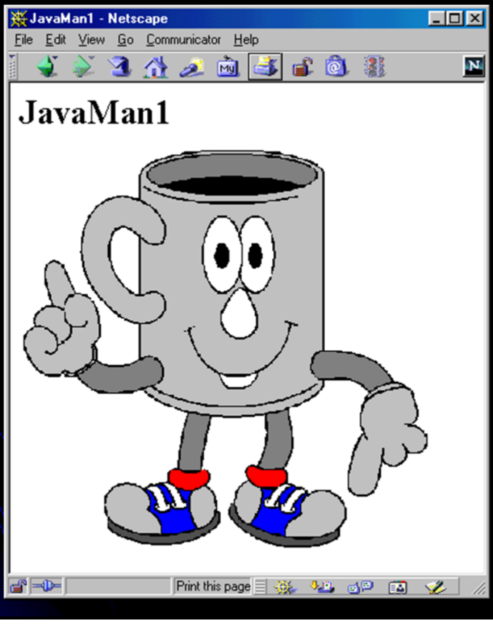
```
g.drawImage(image, x, y, window);
g.drawImage(image, x, y, w, h, window);
```

  - May draw partial image or nothing at all
  - Use the applet (`this`) for the `window` argument

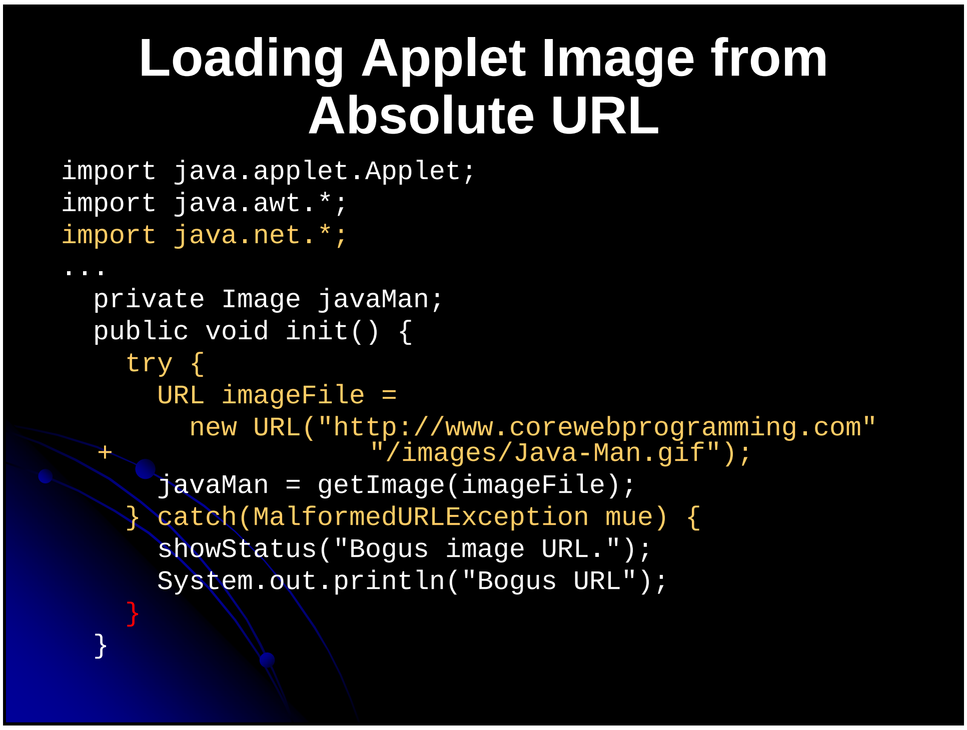
Loading Applet Image from Relative URL

```
import java.applet.Applet;
import java.awt.*;

/** An applet that loads an image from a relative URL. */
public class JavaMan1 extends Applet {
  private Image javaMan;
  public void init() {
    javaMan = getImage(getCodeBase(),
                       "images/Java-Man.gif");
  }
  public void paint(Graphics g) {
    g.drawImage(javaMan, 0, 0, this);
  }
}
```

Image Loading Result

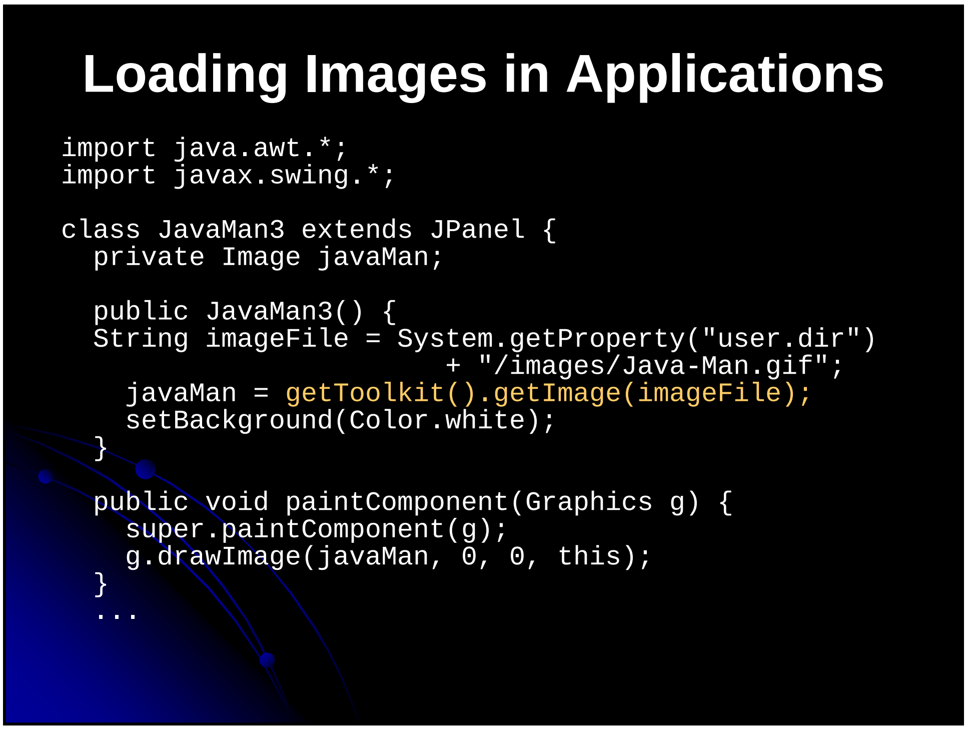


JavaMan1



Loading Applet Image from Absolute URL

```
import java.applet.Applet;
import java.awt.*;
import java.net.*;
...
  private Image javaMan;
  public void init() {
    try {
      URL imageFile =
        new URL("http://www.corewebprogramming.com"
  +                  "/images/Java-Man.gif");
      javaMan = getImage(imageFile);
    } catch(MalformedURLException mue) {
      showStatus("Bogus image URL.");
      System.out.println("Bogus URL");
    }
  }
```

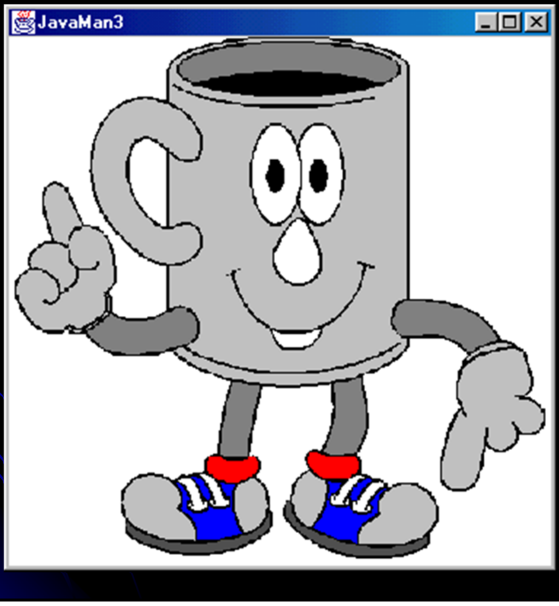


Loading Images in Applications

```
import java.awt.*;
import javax.swing.*;

class JavaMan3 extends JPanel {
  private Image javaMan;

  public JavaMan3() {
  String imageFile = System.getProperty("user.dir")
                        + "/images/Java-Man.gif";
    javaMan = getToolkit().getImage(imageFile);
    setBackground(Color.white);
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(javaMan, 0, 0, this);
  }
  ...
  ...

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(javaMan, 0, 0, this);
  }

  public static void main(String[] args) {
    JPanel panel = new JavaMan3();
    WindowUtilities.setNativeLookAndFeel();
    WindowUtilities.openInJFrame(panel, 380, 390);
  }
}
```

- See Swing chapter for `WindowUtilities`

Loading Images in Applications



Controlling Image Loading

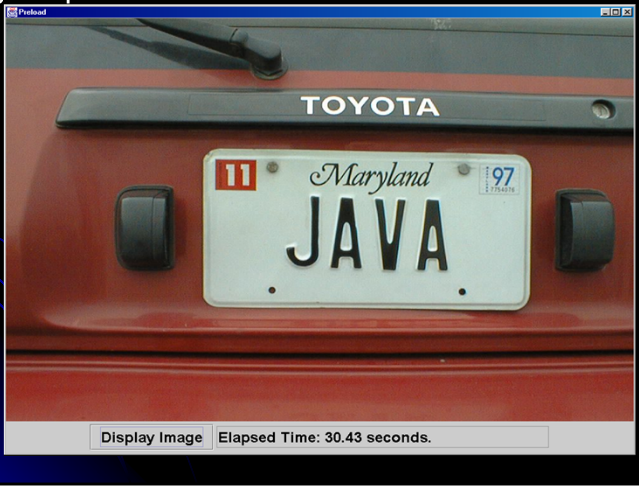
- Use `prepareImage` to start loading image

```
prepareImage(image, window)
prepareImage(image, width, height, window)
```

  - Starts loading image immediately (on separate thread), instead of when needed by `drawImage`
  - Particularly useful if the images will not be drawn until the user initiates some action such as clicking on a button or choosing a menu option
  - Since the applet thread immediately continues execution after the call to `prepareImage`, the image *may* not be completely loaded before `paint` is reached

Controlling Image Loading, Case I: No prepareImage

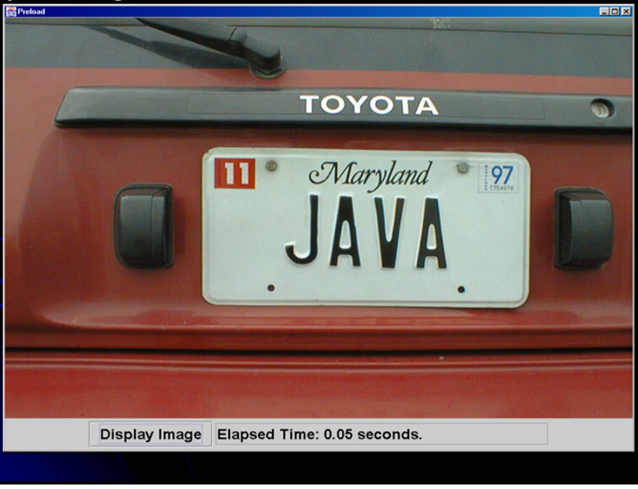
- Image is not loaded over network until after Display Image is pressed. 30.4 seconds.



Elapsed Time: 30.43 seconds.

Controlling Image Loading, Case 2: With prepareImage

- Image loaded over network immediately. 0.05 seconds after pressing button.



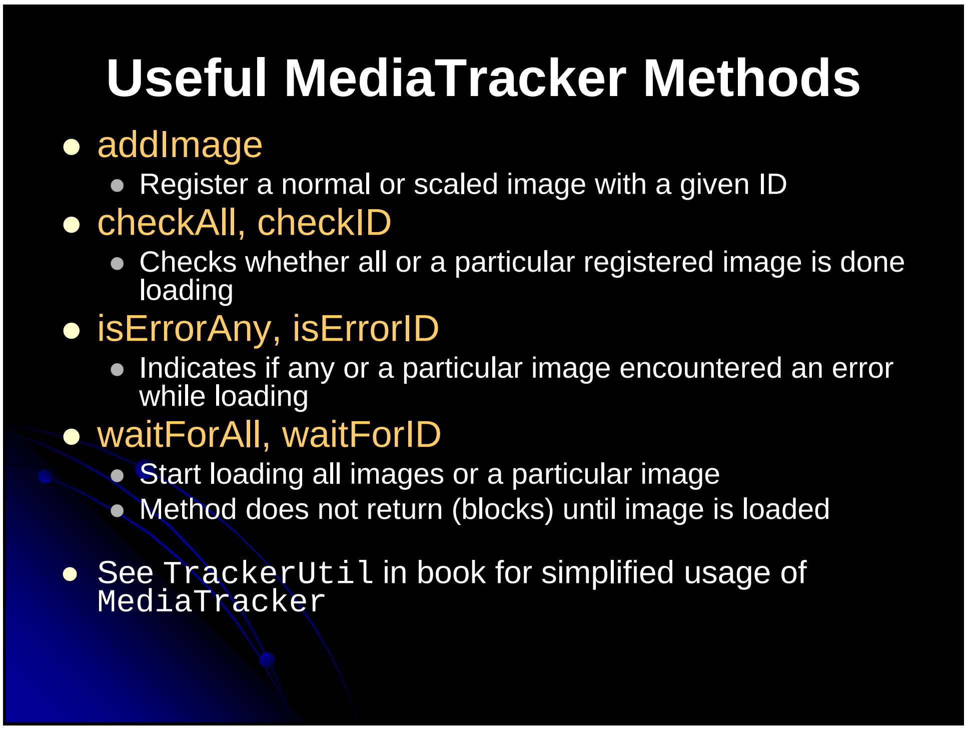
Elapsed Time: 0.05 seconds.

Controlling Image Loading: MediaTracker

- Registering images with a MediaTracker to control image loading

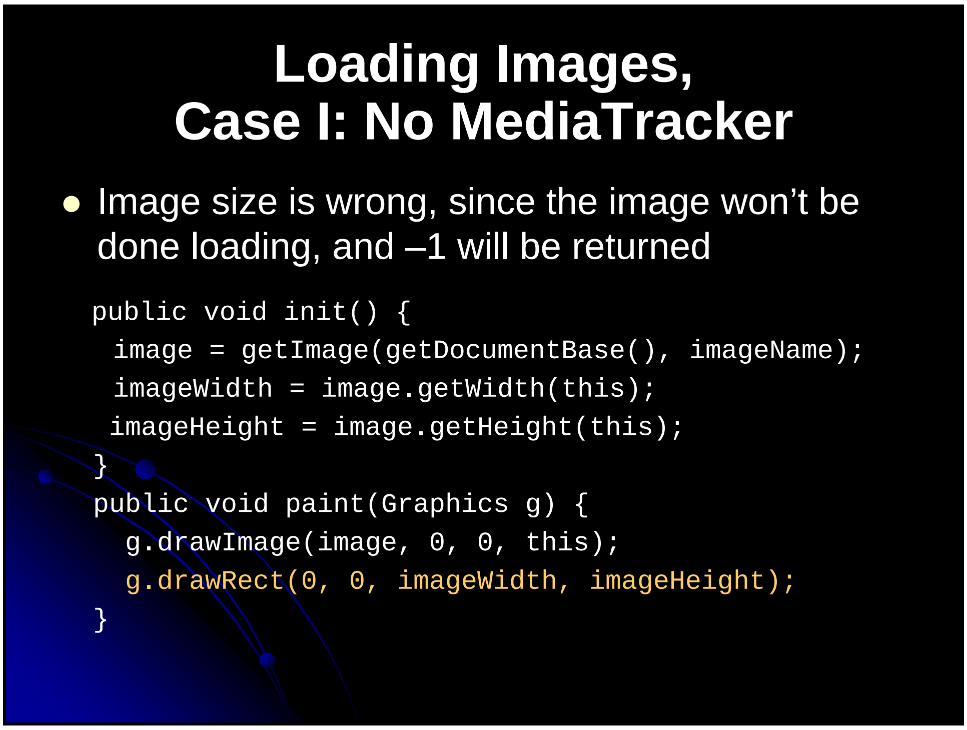
```
MediaTracker tracker = new MediaTracker(this);
  tracker.addImage(image1, 0);
  tracker.addImage(image2, 1);
  try {
    tracker.waitForAll();
  } catch(InterruptedException ie) {}
  if (tracker.isErrorAny()) {
    System.out.println("Error while loading image");
  }
```

  - Applet thread will block until all images are loaded
  - Each image is loaded in parallel on a separate thread



Useful MediaTracker Methods

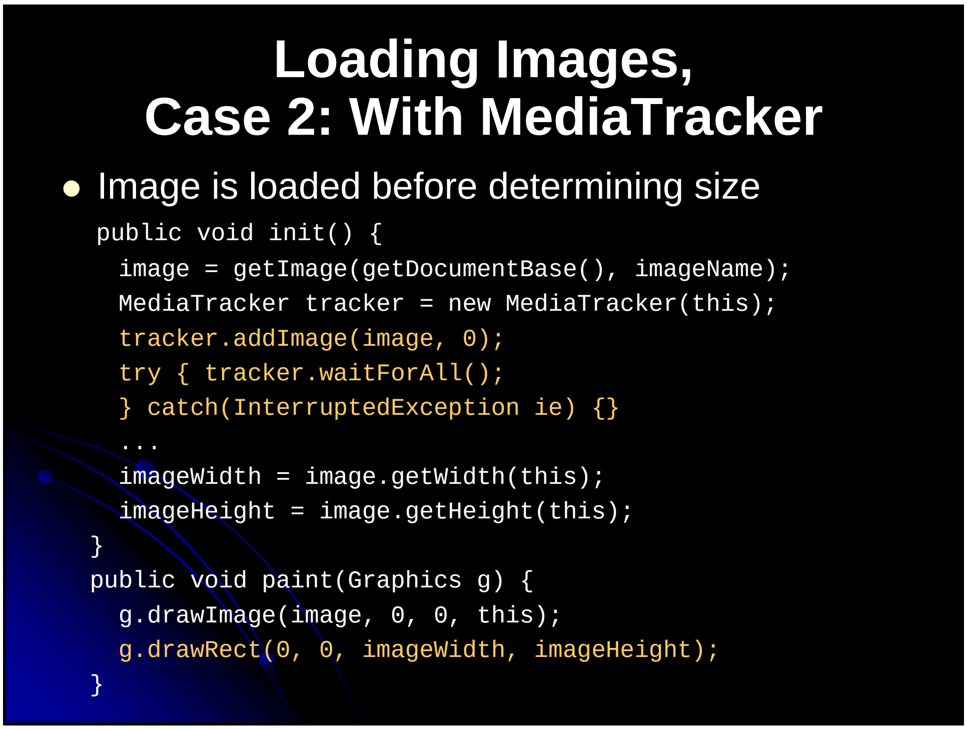
- addImage
  - Register a normal or scaled image with a given ID
- checkAll, checkID
  - Checks whether all or a particular registered image is done loading
- isErrorAny, isErrorID
  - Indicates if any or a particular image encountered an error while loading
- waitForAll, waitForID
  - Start loading all images or a particular image
  - Method does not return (blocks) until image is loaded
- See `TrackerUtil` in book for simplified usage of `MediaTracker`



Loading Images, Case I: No MediaTracker

- Image size is wrong, since the image won’t be done loading, and –1 will be returned

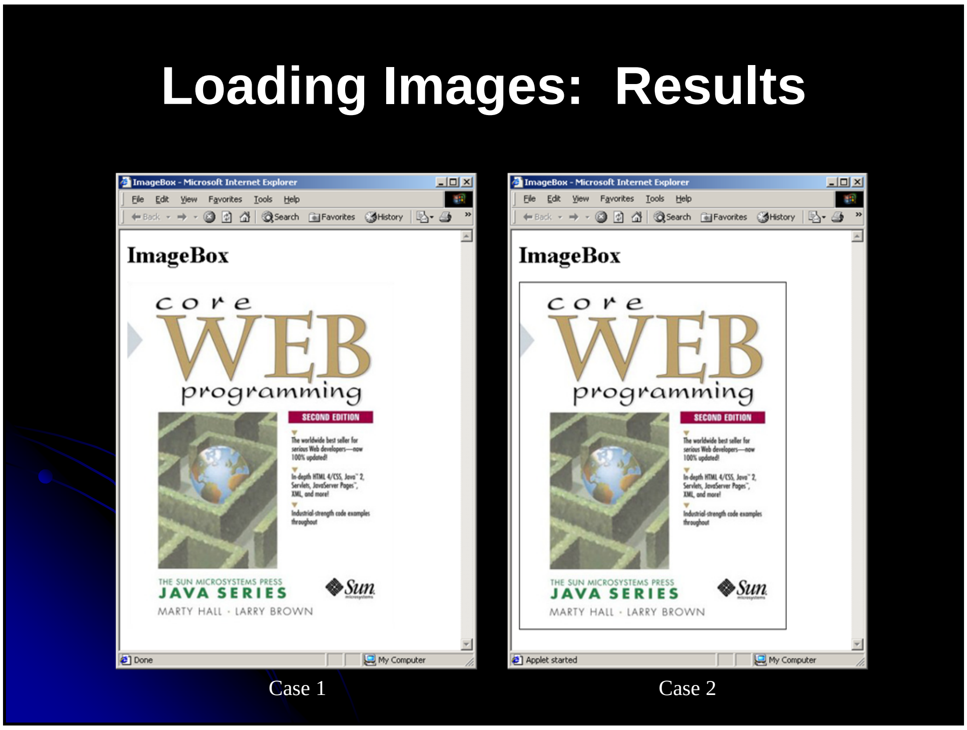
```
public void init() {
  image = getImage(getDocumentBase(), imageName);
  imageWidth = image.getWidth(this);
  imageHeight = image.getHeight(this);
}
public void paint(Graphics g) {
   g.drawImage(image, 0, 0, this);
   g.drawRect(0, 0, imageWidth, imageHeight);
}
```



Loading Images, Case 2: With MediaTracker

- Image is loaded before determining size

```
public void init() {
  image = getImage(getDocumentBase(), imageName);
  MediaTracker tracker = new MediaTracker(this);
  tracker.addImage(image, 0);
  try { tracker.waitForAll();
  } catch(InterruptedException ie) {}
  ...
  imageWidth = image.getWidth(this);
  imageHeight = image.getHeight(this);
}
public void paint(Graphics g) {
  g.drawImage(image, 0, 0, this);
  g.drawRect(0, 0, imageWidth, imageHeight);
}
```



Loading Images: Results

ImageBox - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
ImageBox
core WEB programming
SECOND EDITION
THE SUN MICROSYSTEMS PRESS JAVA SERIES
MARTY HALL · LARRY BROWN
Sun
Done
My Computer
ImageBox - Microsoft Internet Explorer
ImageBox
Applet started
My Computer

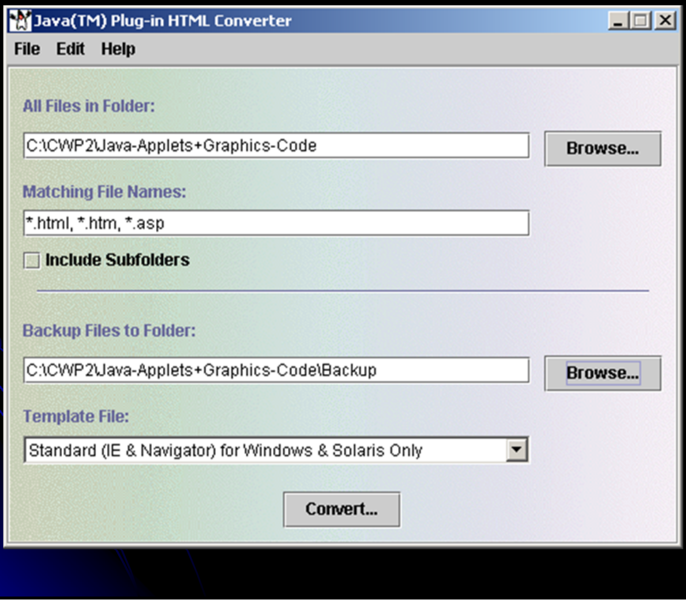
Case 1

Case 2

Java Plug-In

- Internet Explorer and Netscape 4 only support JDK 1.1
- Plugin provides support for the latest JDK
  - http://java.sun.com/products/plugin/
  - Java 2 Plug-In > 5 Mbytes
  - Installing JDK 1.4 or 1.5 installs plugin automatically
- Older browsers require modification of APPLET element to support OBJECT element (IE) or EMBED element (Netscape)
  - Use HTML Converter to perform the modification
  - Not necessary with IE 5/6 or Netscape 6/7

Java Plug-In HTML Converter



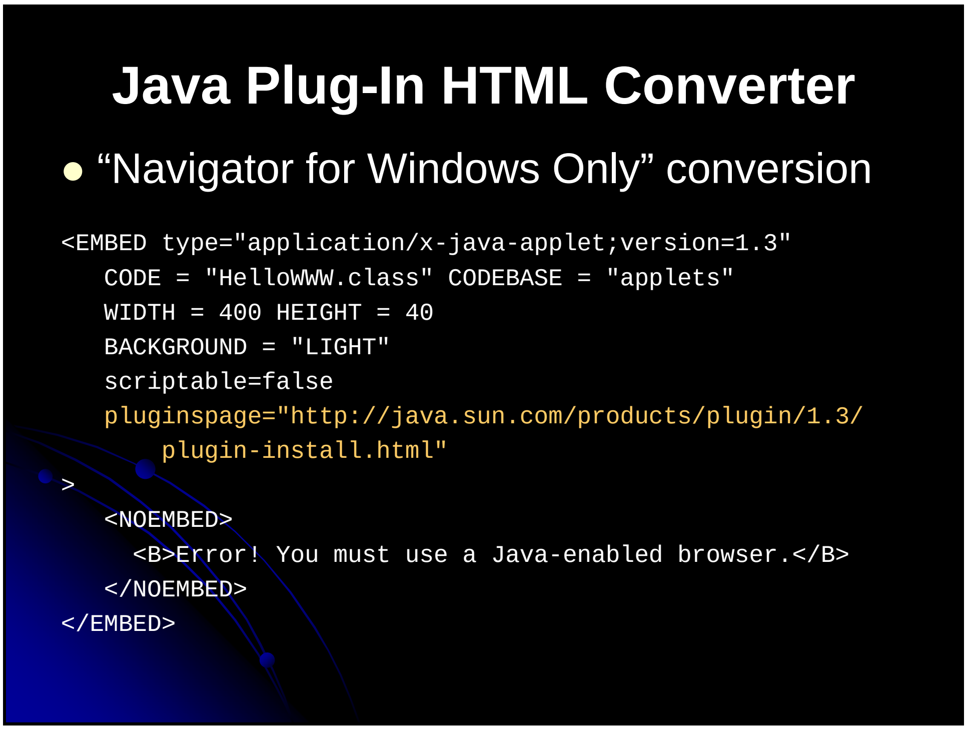
All Files in Folder: C:\CWP2\Java-Applets+Graphics-Code

Matching File Names: *.html, *.htm, *.asp

□ Include Subfolders

Backup Files to Folder: C:\CWP2\Java-Applets+Graphics-Code\Backup

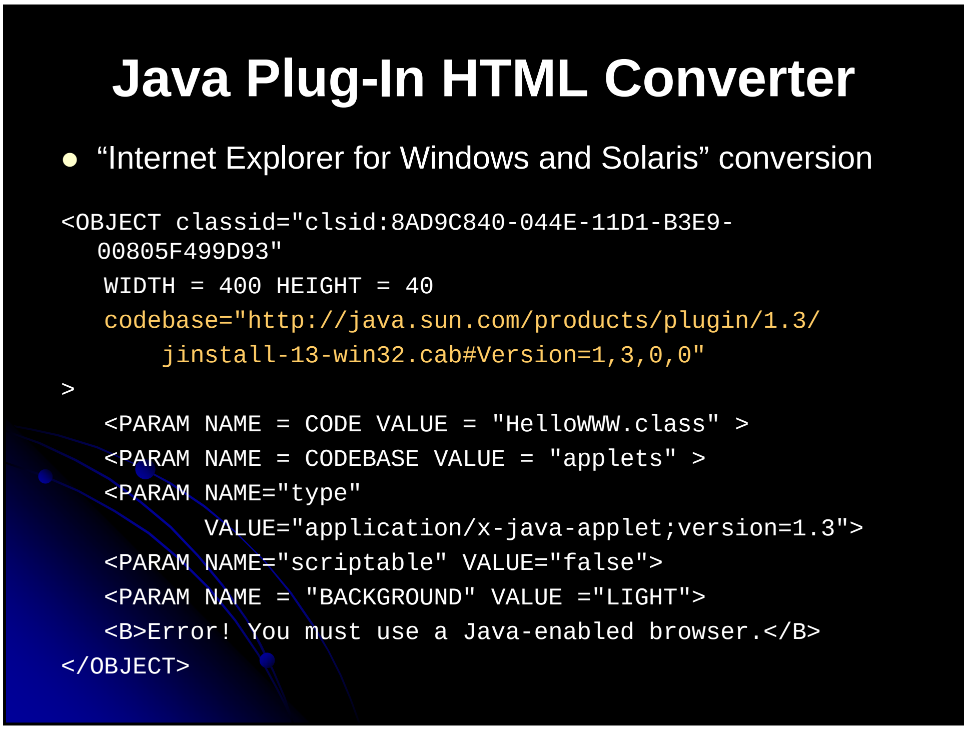
Template File: Standard (IE & Navigator) for Windows & Solaris Only



Java Plug-In HTML Converter

- “Navigator for Windows Only” conversion

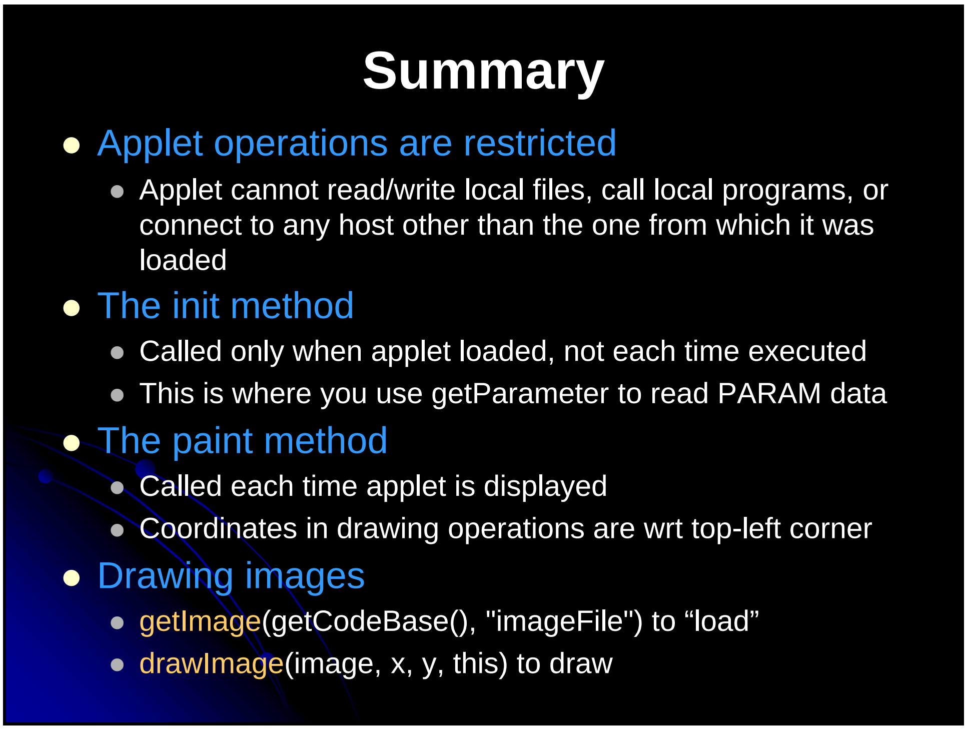
```
<EMBED type="application/x-java-applet;version=1.3"
   CODE = "HelloWWW.class" CODEBASE = "applets"
   WIDTH = 400 HEIGHT = 40
   BACKGROUND = "LIGHT"
   scriptable=false
   pluginspage="http://java.sun.com/products/plugin/1.3/
       plugin-install.html"
>
   <NOEMBED>
     <B>Error! You must use a Java-enabled browser.</B>
   </NOEMBED>
</EMBED>
```



Java Plug-In HTML Converter

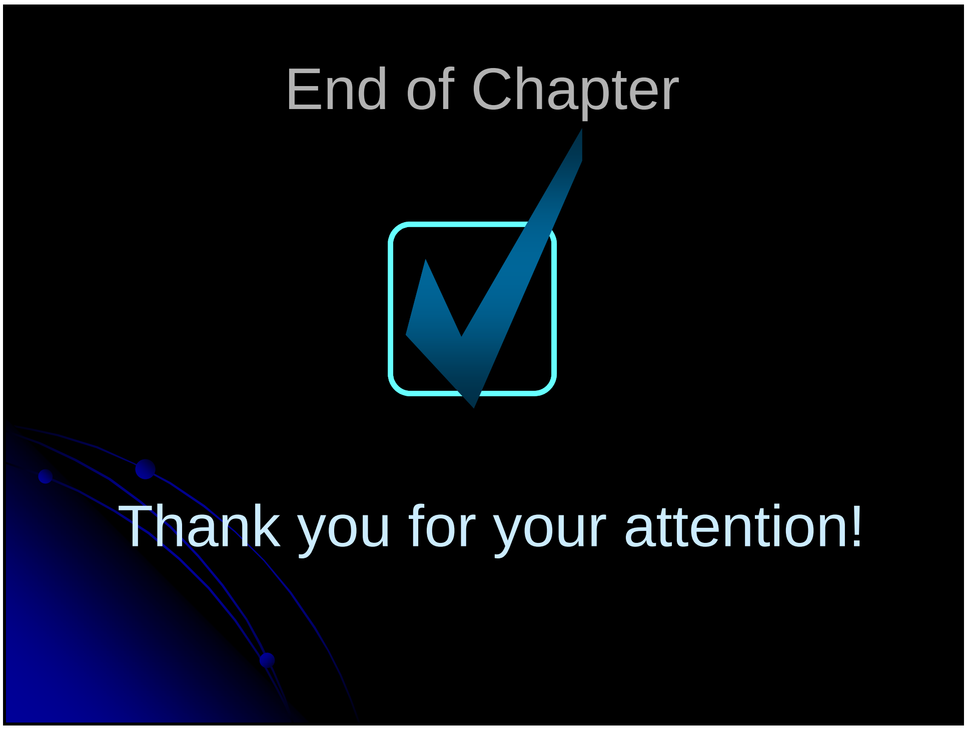
- “Internet Explorer for Windows and Solaris” conversion

```
<OBJECT classid="clsid:8AD9C840-044E-11D1-B3E9-
  00805F499D93"
   WIDTH = 400 HEIGHT = 40
   codebase="http://java.sun.com/products/plugin/1.3/
       jinstall-13-win32.cab#Version=1,3,0,0"
>
   <PARAM NAME = CODE VALUE = "HelloWWW.class" >
   <PARAM NAME = CODEBASE VALUE = "applets" >
   <PARAM NAME="type"
          VALUE="application/x-java-applet;version=1.3">
   <PARAM NAME="scriptable" VALUE="false">
   <PARAM NAME = "BACKGROUND" VALUE ="LIGHT">
   <B>Error! You must use a Java-enabled browser.</B>
</OBJECT>
```



Summary

- Applet operations are restricted
  - Applet cannot read/write local files, call local programs, or connect to any host other than the one from which it was loaded
- The init method
  - Called only when applet loaded, not each time executed
  - This is where you use getParameter to read PARAM data
- The paint method
  - Called each time applet is displayed
  - Coordinates in drawing operations are wrt top-left corner
- Drawing images
  - getImage(getCodeBase(), "imageFile") to “load”
  - drawImage(image, x, y, this) to draw



End of Chapter

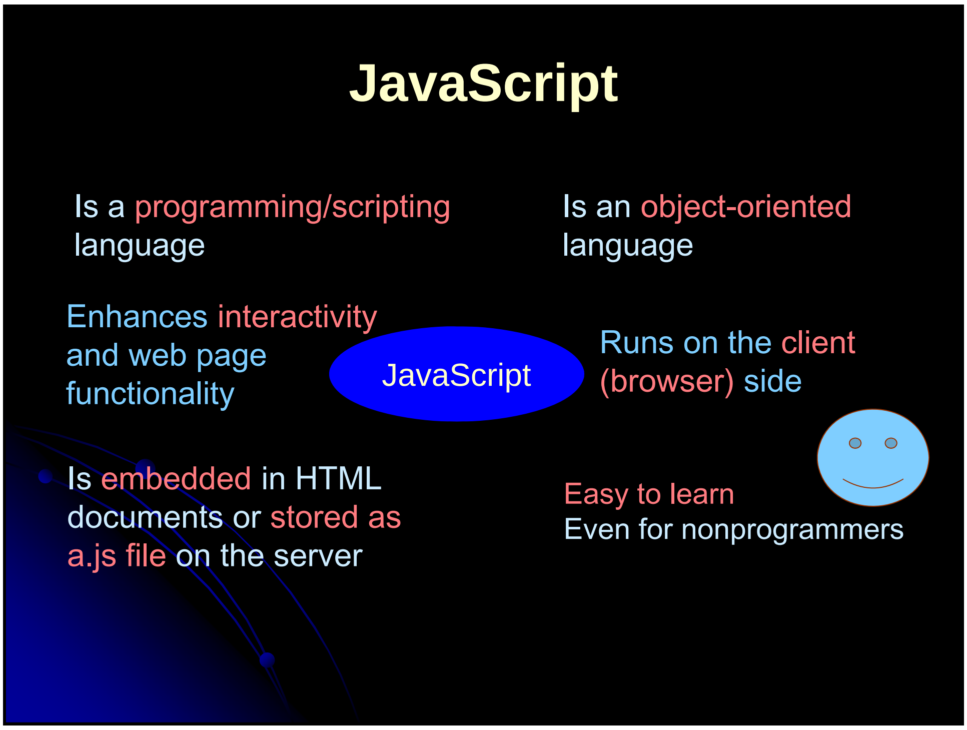
Thank you for your attention!

Java Scripts

Adding Dynamic Content
to Web Pages

Agenda

- Generating HTML Dynamically
- Monitoring User Events
- Basic JavaScript Syntax
- Applications
  - Using JavaScript to customize Web pages
  - Using JavaScript to make pages more dynamic
  - Using JavaScript to validate CGI forms
  - Using JavaScript to manipulate HTTP cookies
  - Using JavaScript to interact with and control frames
  - Controlling applets and calling Java from JavaScript
  - Accessing JavaScript from Java



JavaScript

Is a programming/scripting language

Is an object-oriented language

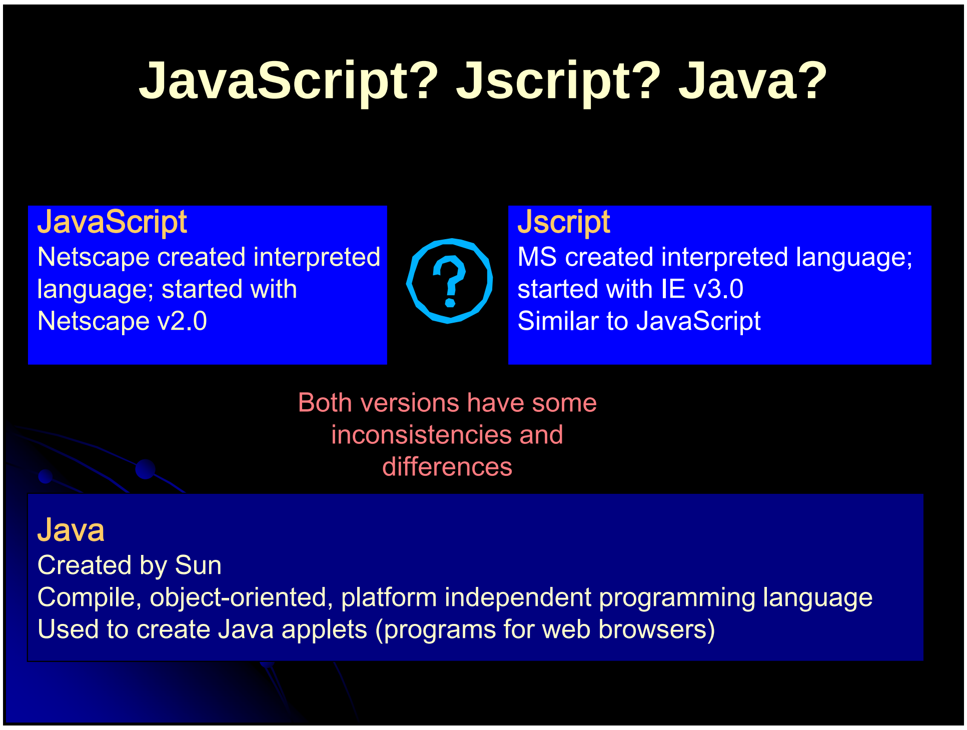
Enhances interactivity and web page functionality

JavaScript

Runs on the client (browser) side

Is embedded in HTML documents or stored as a.js file on the server

Easy to learn
Even for nonprogrammers



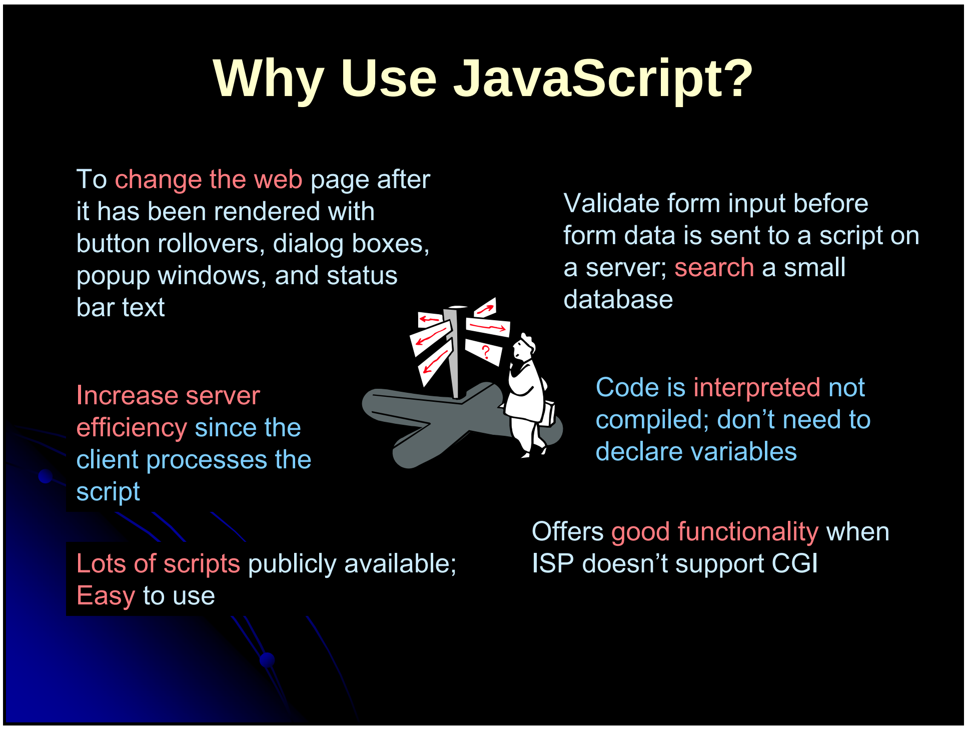
JavaScript? Jscript? Java?

JavaScript
Netscape created interpreted language; started with Netscape v2.0

Jscript
MS created interpreted language; started with IE v3.0
Similar to JavaScript

Both versions have some inconsistencies and differences

Java
Created by Sun
Compile, object-oriented, platform independent programming language
Used to create Java applets (programs for web browsers)



Why Use JavaScript?

To change the web page after it has been rendered with button rollovers, dialog boxes, popup windows, and status bar text

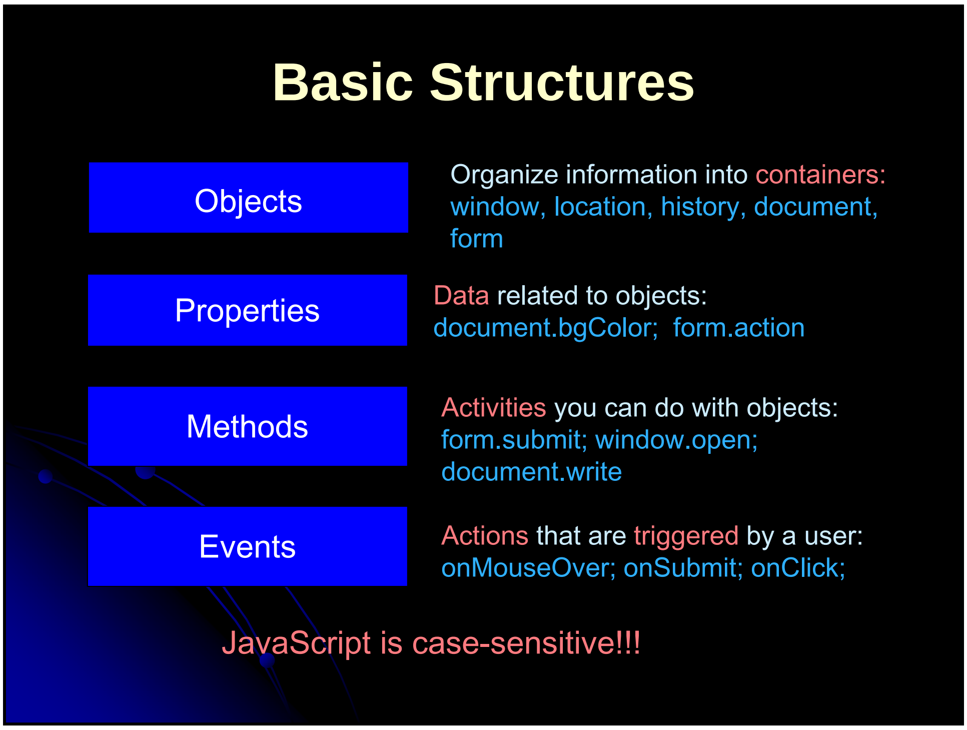
Validate form input before form data is sent to a script on a server; search a small database

Increase server efficiency since the client processes the script

Code is interpreted not compiled; don't need to declare variables

Lots of scripts publicly available; Easy to use

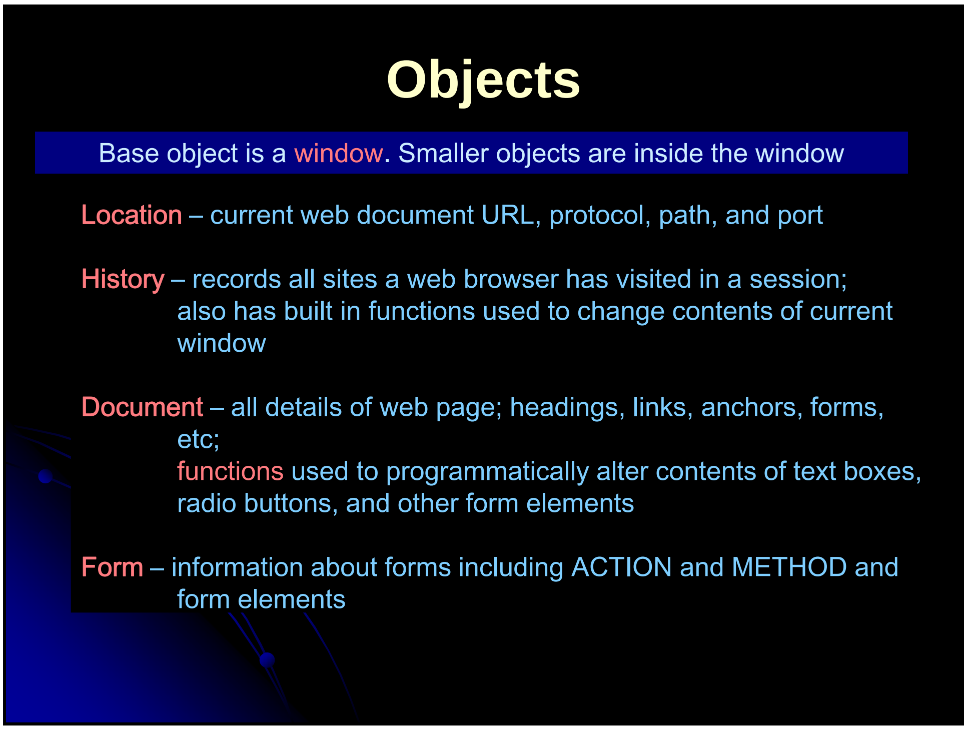
Offers good functionality when ISP doesn't support CGI



Basic Structures

Objects	Organize information into containers: window, location, history, document, form
Properties	Data related to objects: document.bgColor; form.action
Methods	Activities you can do with objects: form.submit; window.open; document.write
Events	Actions that are triggered by a user: onMouseOver; onSubmit; onClick;

JavaScript is case-sensitive!!!



Objects

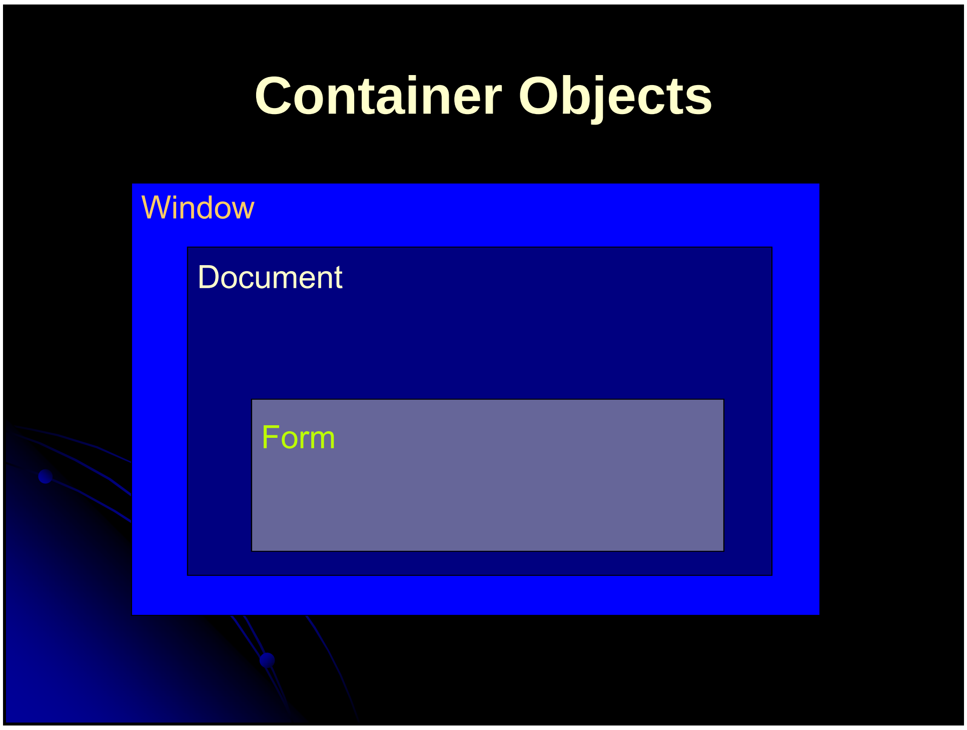
Base object is a window. Smaller objects are inside the window

Location – current web document URL, protocol, path, and port

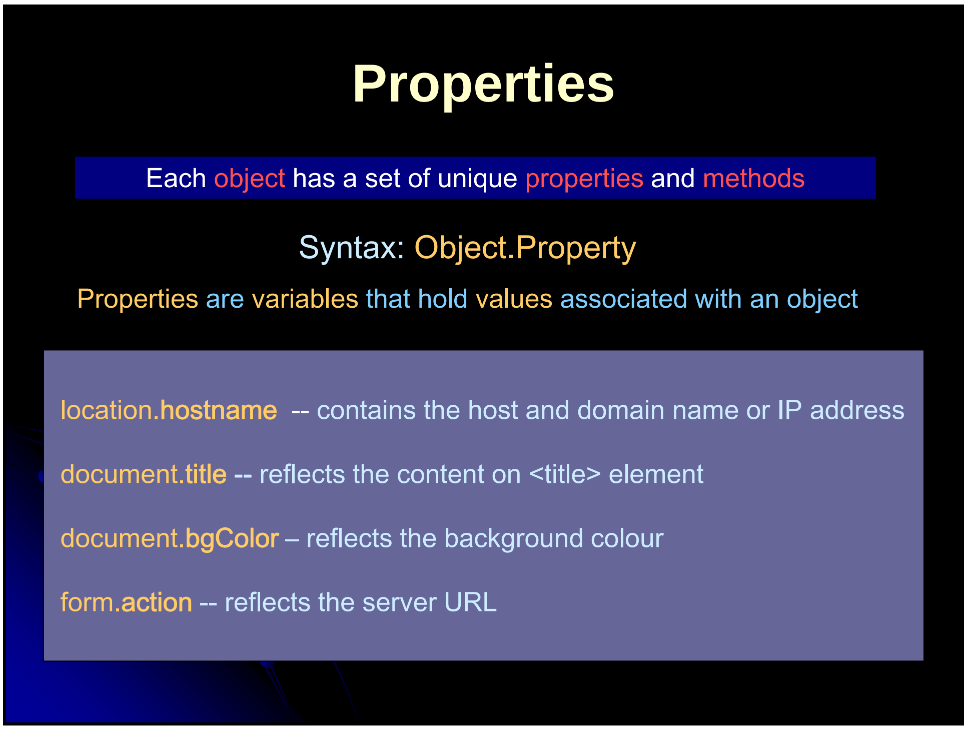
History – records all sites a web browser has visited in a session; also has built in functions used to change contents of current window

Document – all details of web page; headings, links, anchors, forms, etc;
functions used to programmatically alter contents of text boxes, radio buttons, and other form elements

Form – information about forms including ACTION and METHOD and form elements



Container Objects



Properties

Each object has a set of unique properties and methods

Syntax: Object.Property

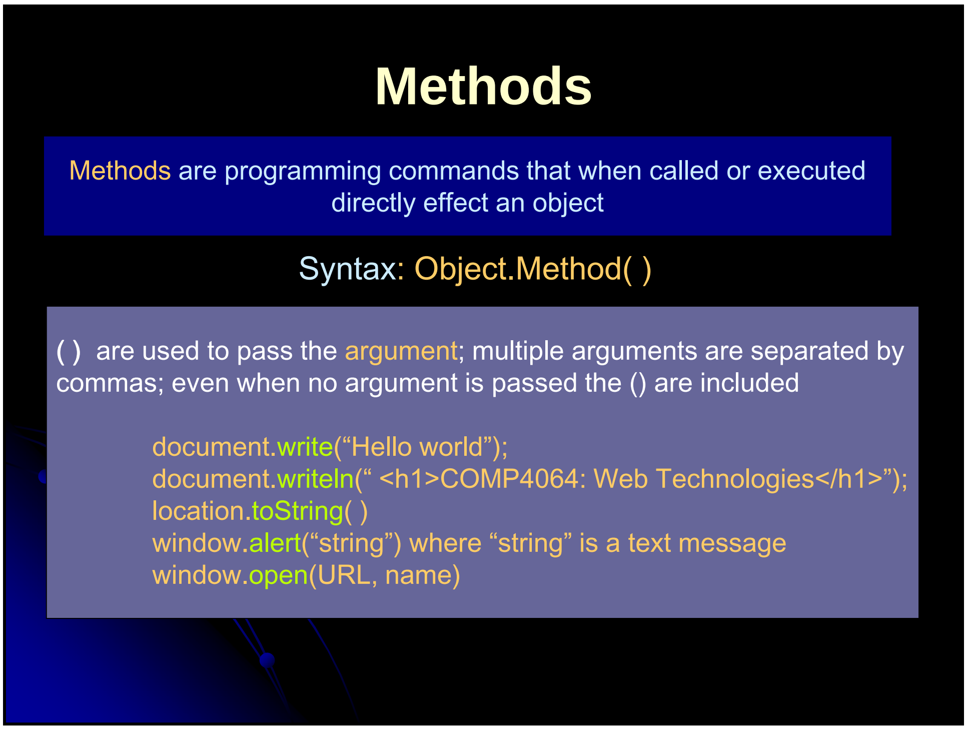
Properties are variables that hold values associated with an object

location.hostname -- contains the host and domain name or IP address

document.title -- reflects the content on <title> element

document.bgColor – reflects the background colour

form.action -- reflects the server URL



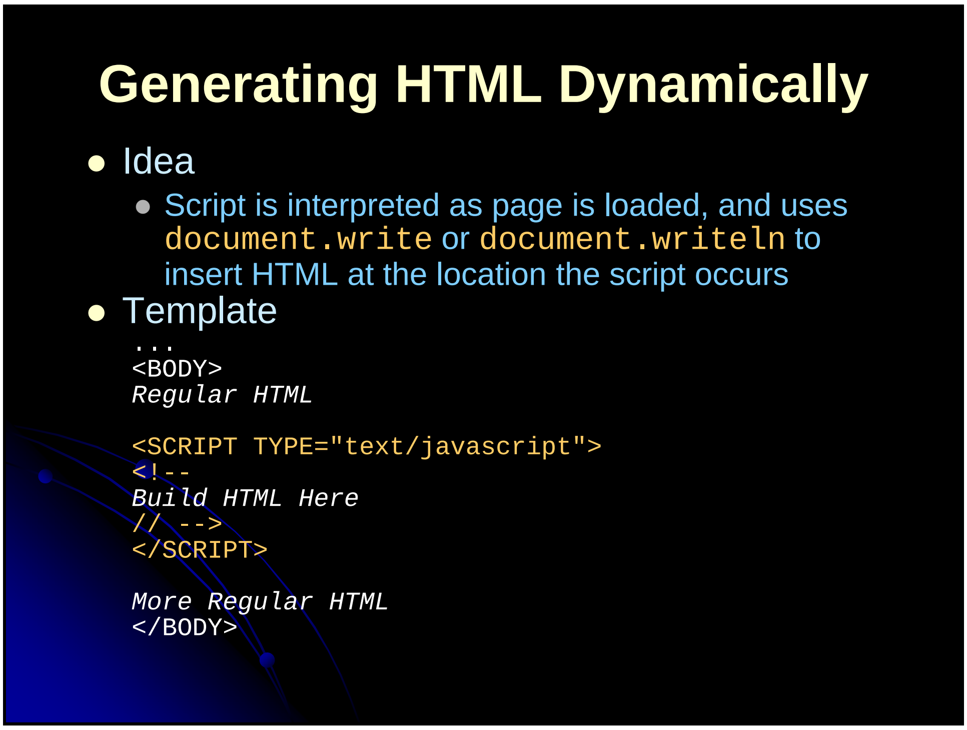
Methods

Methods are programming commands that when called or executed directly effect an object

Syntax: Object.Method( )

( ) are used to pass the argument; multiple arguments are separated by commas; even when no argument is passed the () are included

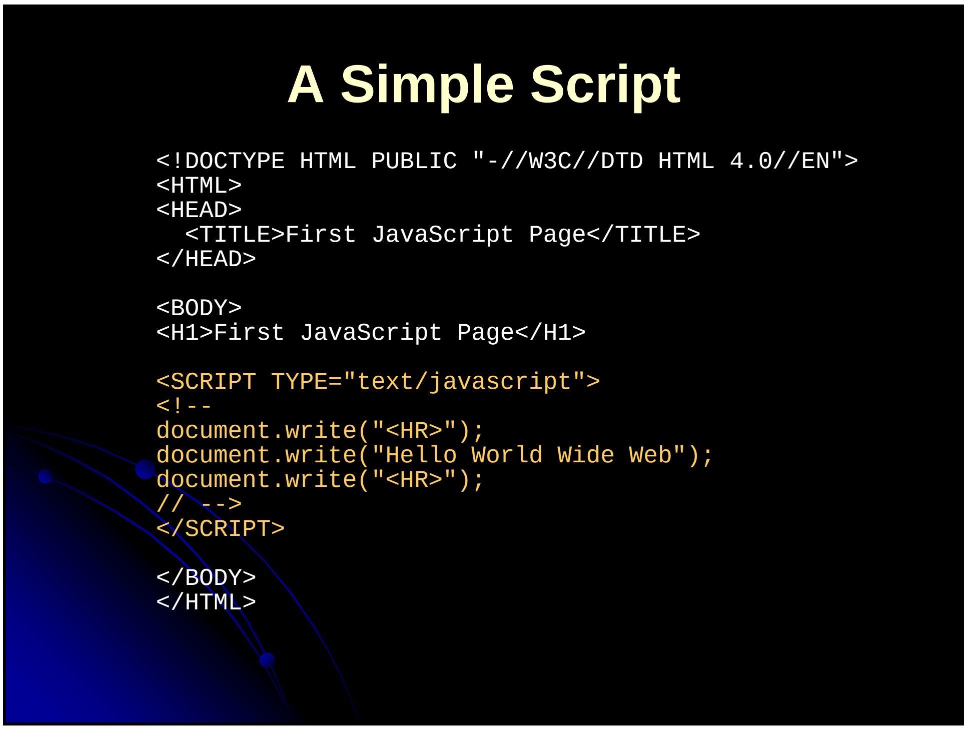
```
document.write(“Hello world”);
document.writeln(“ <h1>COMP4064: Web Technologies</h1>”);
location.toString( )
window.alert(“string”) where “string” is a text message
window.open(URL, name)
```



Generating HTML Dynamically

- Idea
  - Script is interpreted as page is loaded, and uses `document.write` or `document.writeln` to insert HTML at the location the script occurs
- Template

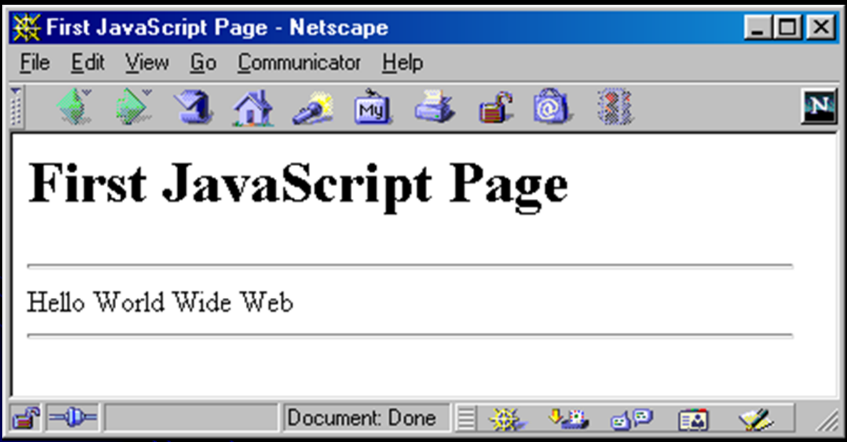
```
...
<BODY>
Regular HTML

<SCRIPT TYPE="text/javascript">
<!--
Build HTML Here
// -->
</SCRIPT>

More Regular HTML
</BODY>
```



A Simple Script

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
  <TITLE>First JavaScript Page</TITLE>
</HEAD>

<BODY>
<H1>First JavaScript Page</H1>

<SCRIPT TYPE="text/javascript">
<!--
document.write("<HR>");
document.write("Hello World Wide Web");
document.write("<HR>");
// -->
</SCRIPT>

</BODY>
</HTML>
```

Simple Script, Result


First JavaScript Page - Netscape
File Edit View Go Communicator Help
Document: Done

First JavaScript Page

Hello World Wide Web

Script Locations

External .js Source File

Text files with no HTML code

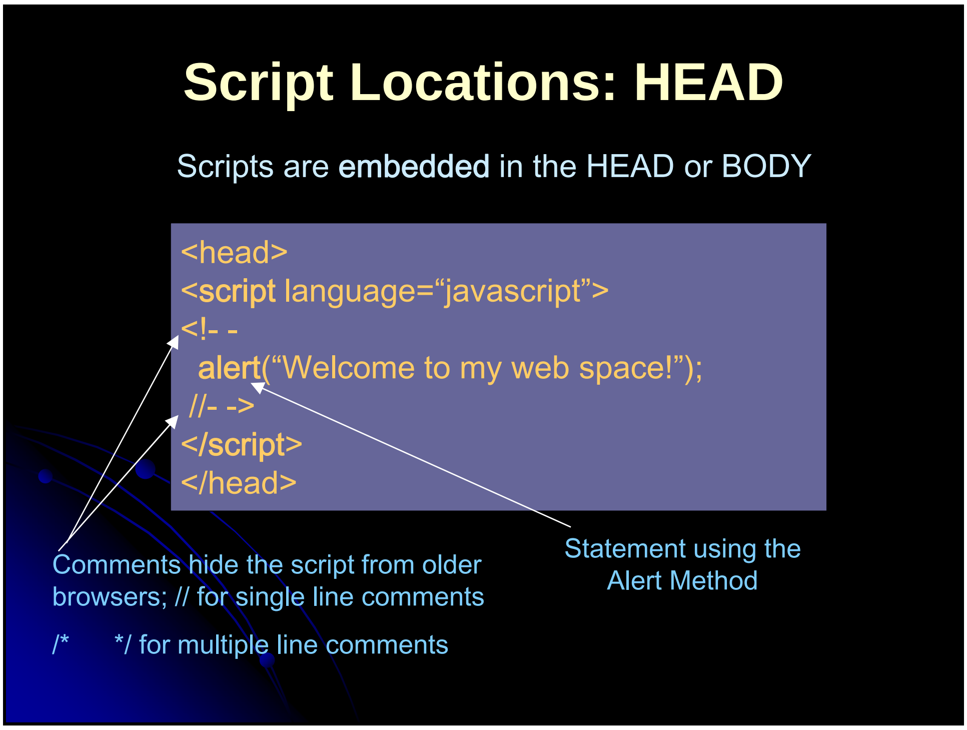
Stored on the server

Called from SRC
<script language=“javascript”
src=“myscript.js”>
</script>

Embedded JavaScript

Statements are read in order First in the HEAD and then in the body

You can combine JavaScript source code with embedded code



Script Locations: HEAD

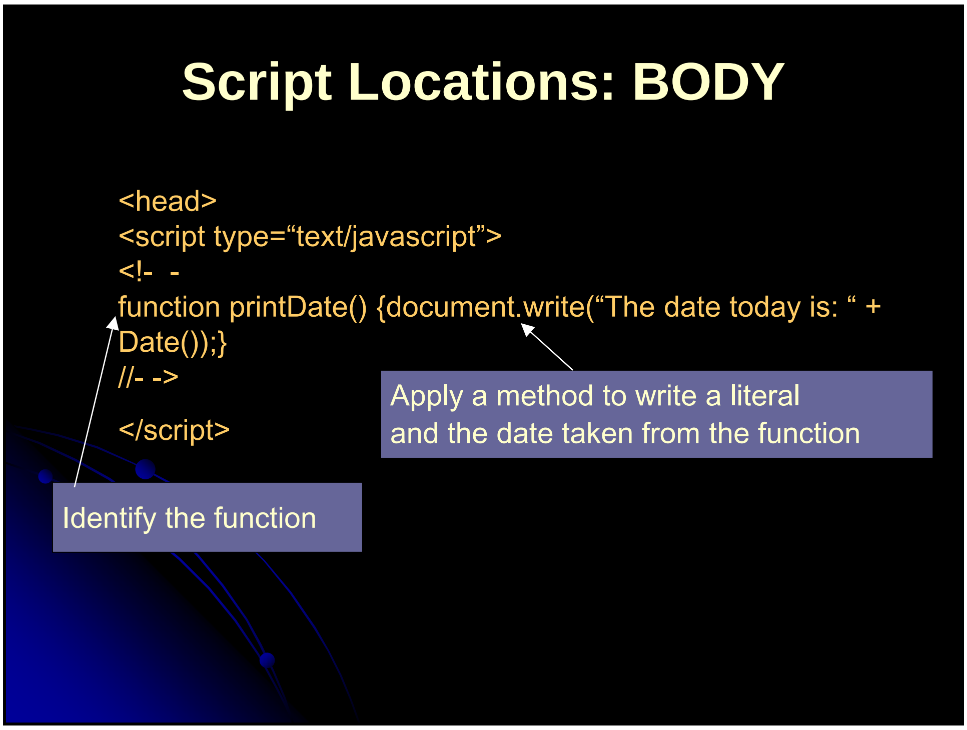
Scripts are **embedded** in the HEAD or BODY

```
<head>
<script language="javascript">
<!- -
  alert("Welcome to my web space!");
 //- ->
</script>
</head>
```

Comments hide the script from older browsers; // for single line comments

/* */ for multiple line comments

Statement using the Alert Method

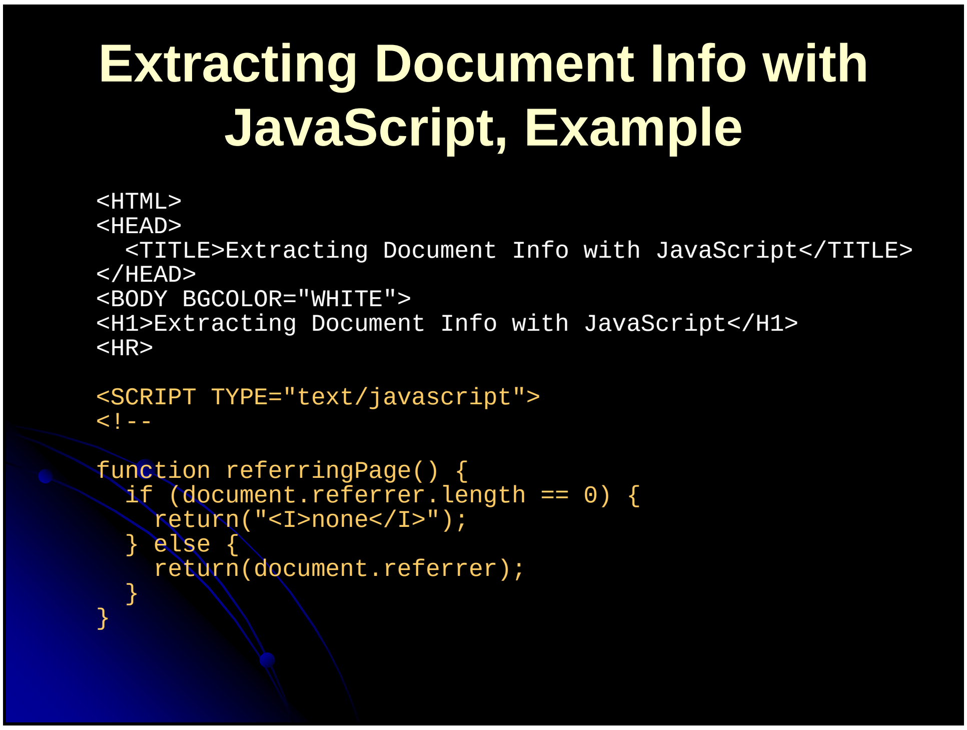


Script Locations: BODY

```
<head>
<script type="text/javascript">
<!-  -
function printDate() {document.write("The date today is: " +
Date());}
//- ->

</script>
```

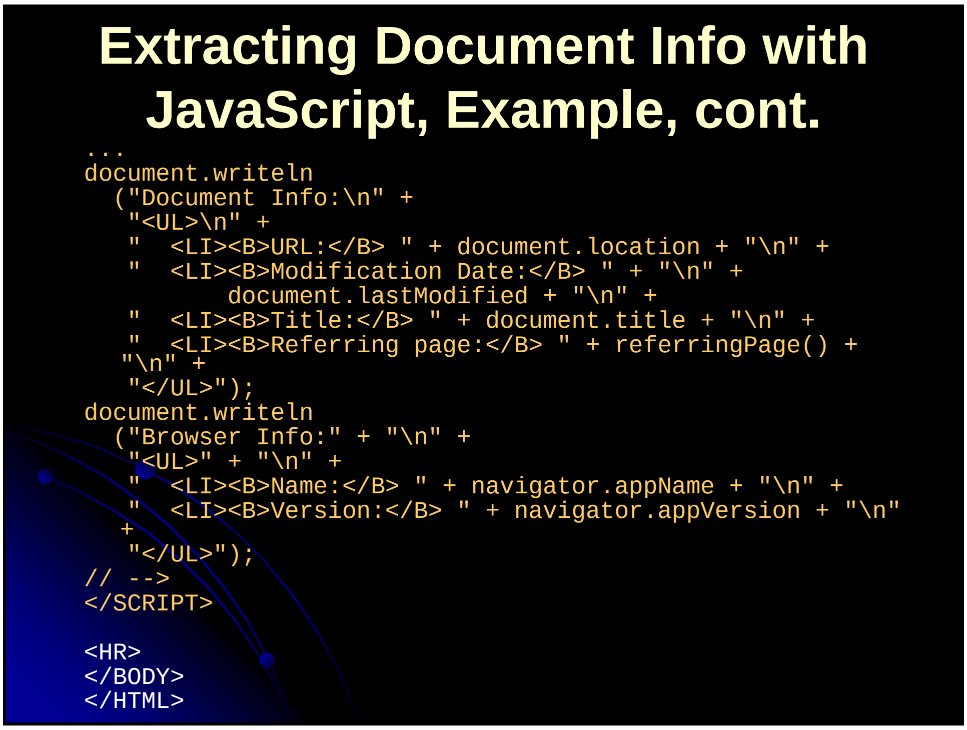
Apply a method to write a literal and the date taken from the function

Identify the function



Extracting Document Info with JavaScript, Example

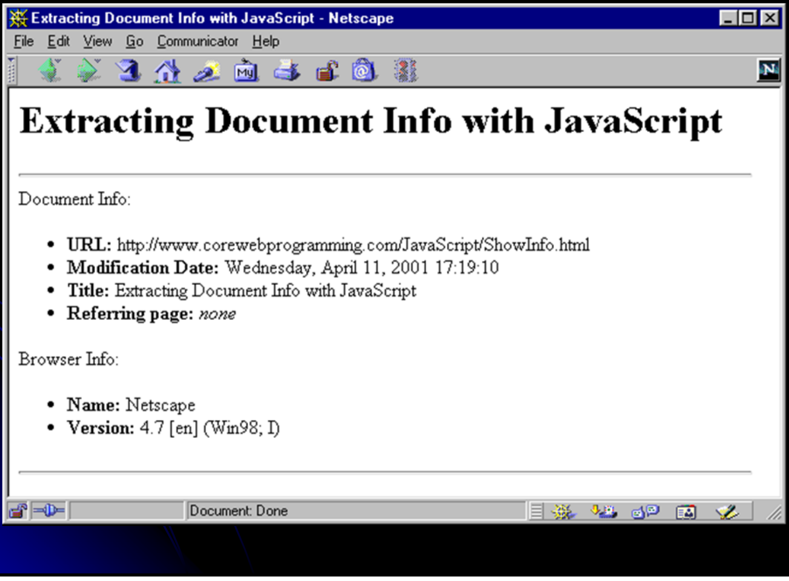
```
<HTML>
<HEAD>
  <TITLE>Extracting Document Info with JavaScript</TITLE>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>Extracting Document Info with JavaScript</H1>
<HR>

<SCRIPT TYPE="text/javascript">
<!--

function referringPage() {
  if (document.referrer.length == 0) {
    return("<I>none</I>");
  } else {
    return(document.referrer);
  }
}
```



Extracting Document Info with JavaScript, Example, cont.

```
...
document.writeln
  ("Document Info:\n" +
   "<UL>\n" +
   "  <LI><B>URL:</B> " + document.location + "\n" +
   "  <LI><B>Modification Date:</B> " + "\n" +
          document.lastModified + "\n" +
   "  <LI><B>Title:</B> " + document.title + "\n" +
   "  <LI><B>Referring page:</B> " + referringPage() +
  "\n" +
   "</UL>");
document.writeln
  ("Browser Info:" + "\n" +
   "<UL>" + "\n" +
   "  <LI><B>Name:</B> " + navigator.appName + "\n" +
   "  <LI><B>Version:</B> " + navigator.appVersion + "\n"
  +
   "</UL>");
// -->
</SCRIPT>

<HR>
</BODY>
</HTML>
```

Extracting Document Info with JavaScript, Result


Extracting Document Info with JavaScript - Netscape
File Edit View Go Communicator Help
Document: Done

Extracting Document Info with JavaScript

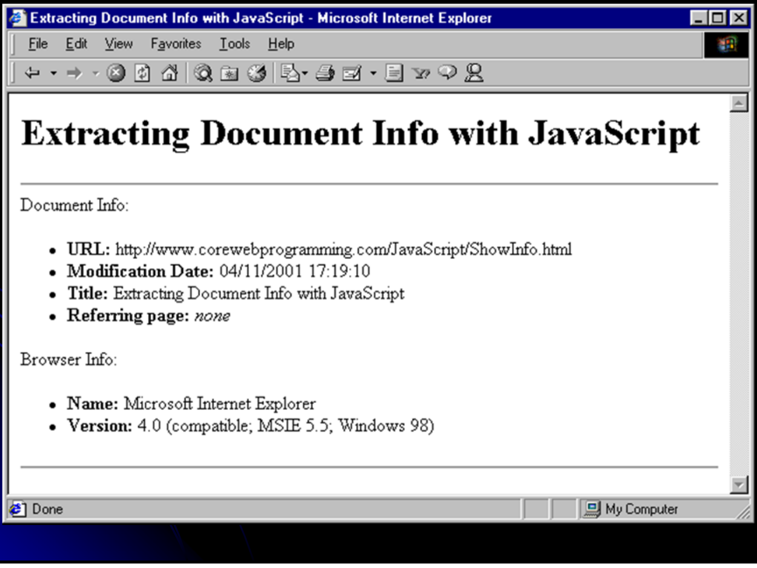
Document Info:

- **URL:** http://www.corewebprogramming.com/JavaScript/ShowInfo.html
- **Modification Date:** Wednesday, April 11, 2001 17:19:10
- **Title:** Extracting Document Info with JavaScript
- **Referring page:** *none*

Browser Info:

- **Name:** Netscape
- **Version:** 4.7 [en] (Win98; I)

Extracting Document Info with JavaScript, Result


Extracting Document Info with JavaScript - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Done
My Computer

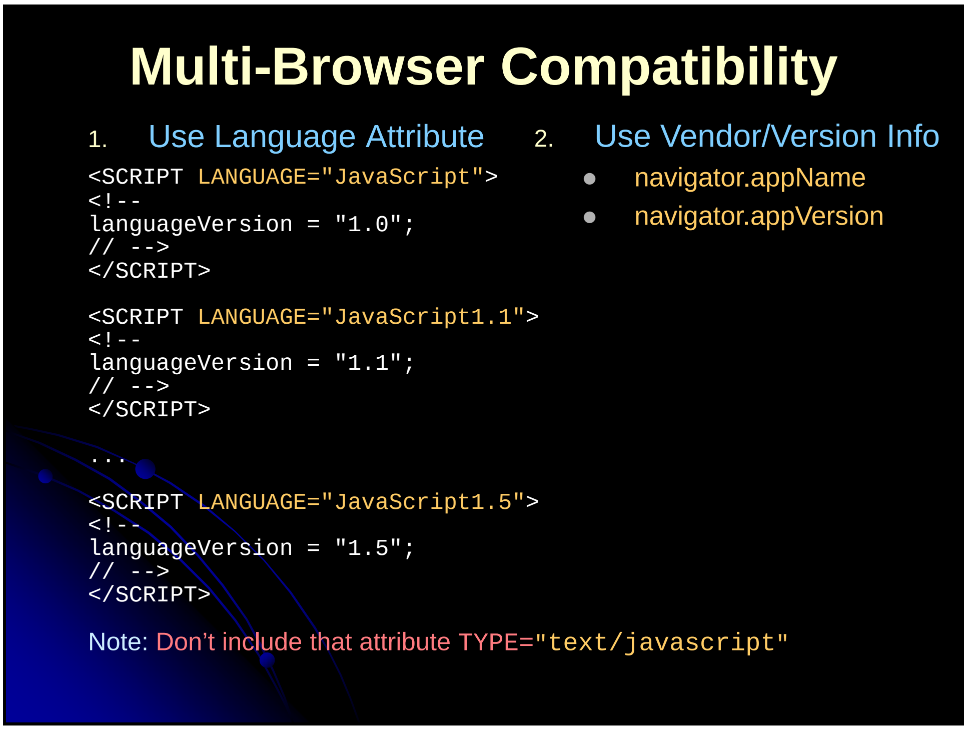
Extracting Document Info with JavaScript

Document Info:

- **URL:** http://www.corewebprogramming.com/JavaScript/ShowInfo.html
- **Modification Date:** 04/11/2001 17:19:10
- **Title:** Extracting Document Info with JavaScript
- **Referring page:** *none*

Browser Info:

- **Name:** Microsoft Internet Explorer
- **Version:** 4.0 (compatible; MSIE 5.5; Windows 98)



Multi-Browser Compatibility

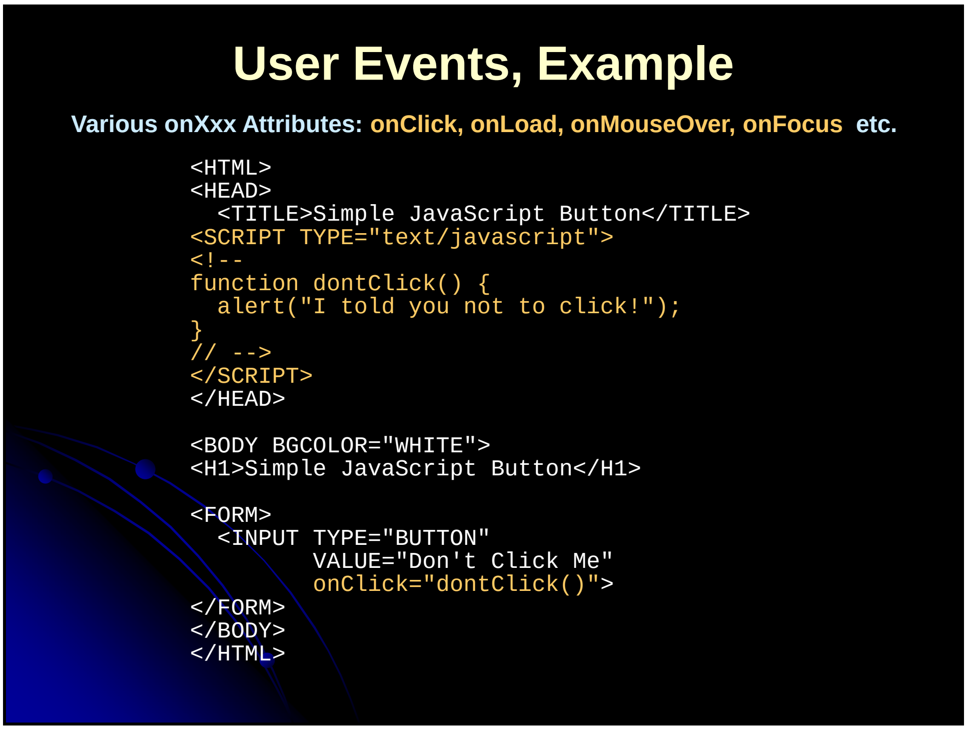
1. Use Language Attribute

```
<SCRIPT LANGUAGE="JavaScript">
<!--
languageVersion = "1.0";
// -->
</SCRIPT>

<SCRIPT LANGUAGE="JavaScript1.1">
<!--
languageVersion = "1.1";
// -->
</SCRIPT>

...
<SCRIPT LANGUAGE="JavaScript1.5">
<!--
languageVersion = "1.5";
// -->
</SCRIPT>
```

Note: Don't include that attribute TYPE="text/javascript"

2. Use Vendor/Version Info

- navigator.appName
- navigator.appVersion

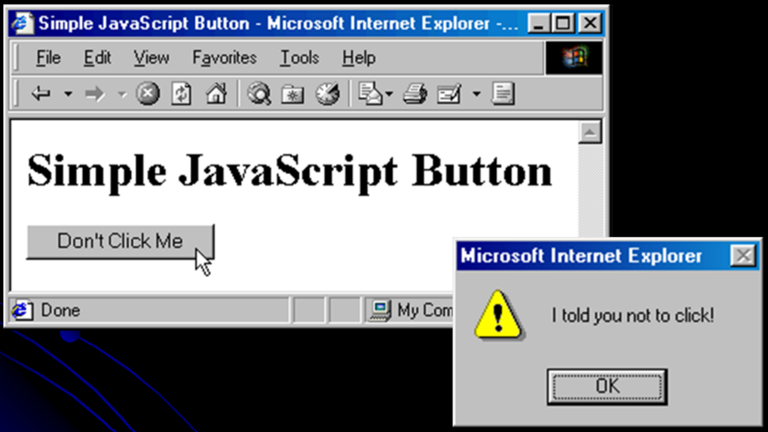


User Events, Example

Various onXxx Attributes: onClick, onLoad, onMouseOver, onFocus etc.

```
<HTML>
<HEAD>
  <TITLE>Simple JavaScript Button</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function dontClick() {
  alert("I told you not to click!");
}
// -->
</SCRIPT>
</HEAD>

<BODY BGCOLOR="WHITE">
<H1>Simple JavaScript Button</H1>

<FORM>
  <INPUT TYPE="BUTTON"
         VALUE="Don't Click Me"
         onClick="dontClick()">
</FORM>
</BODY>
</HTML>
```

User Events, Result



Simple JavaScript Button - Microsoft Internet Explorer -...

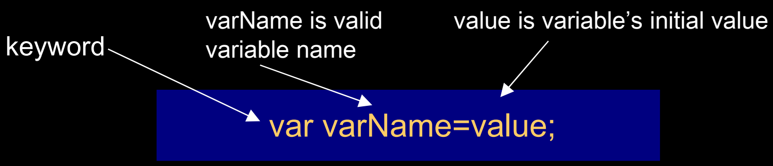
Simple JavaScript Button

Microsoft Internet Explorer

I told you not to click!

Variables

JavaScript is loosely typed – don’t need to specify data types when you declare variables. A value is only checked for proper type when it is operated upon.



keyword

varName is valid variable name

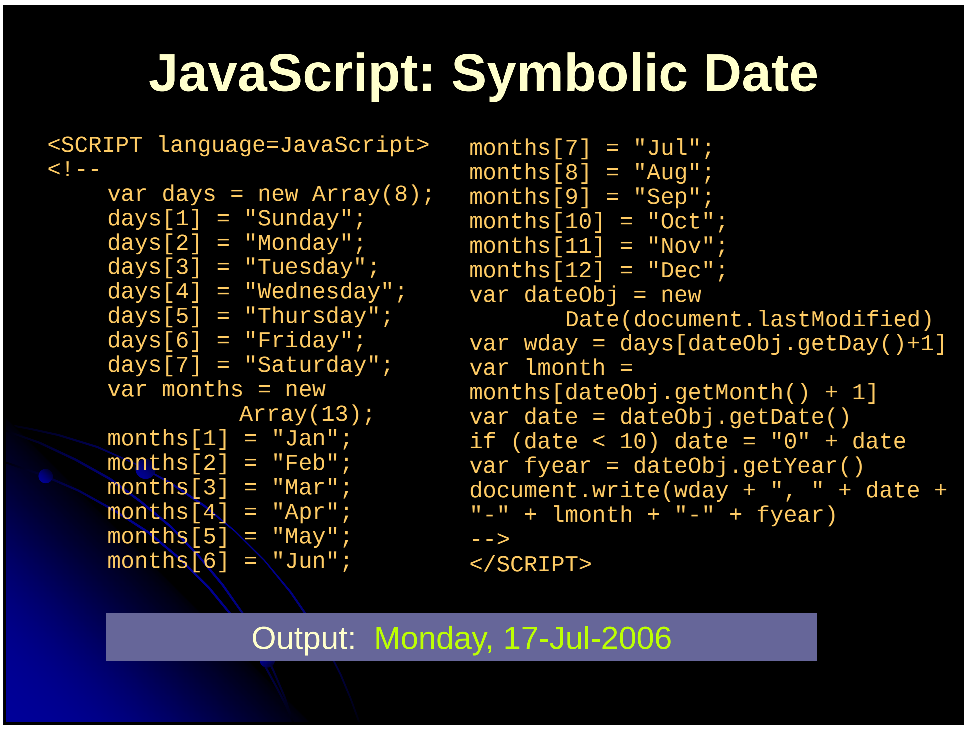
value is variable’s initial value

```
var varName=value;
```

varName can contain letters, numbers, and underscore but can’t begin with a number

```
var x = 5; // int
x = 5.5; // float
x = "five point five"; // String
```

Variables declared inside a function are LOCAL to that function

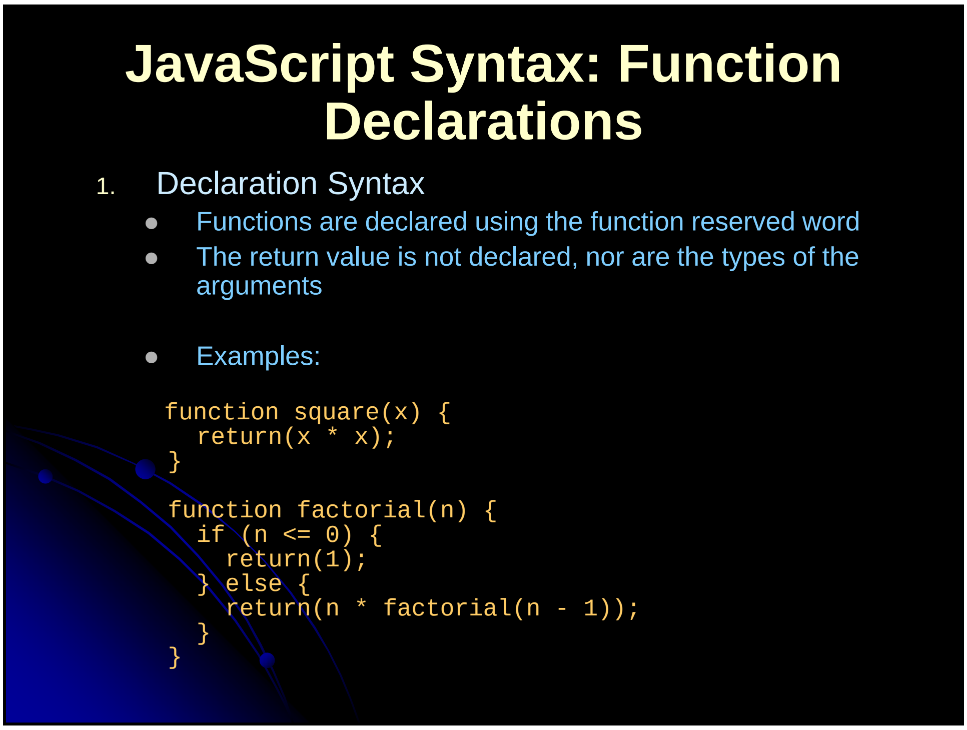


JavaScript: Symbolic Date

```
<SCRIPT language=JavaScript>
<!--
    var days = new Array(8);
    days[1] = "Sunday";
    days[2] = "Monday";
    days[3] = "Tuesday";
    days[4] = "Wednesday";
    days[5] = "Thursday";
    days[6] = "Friday";
    days[7] = "Saturday";
    var months = new
             Array(13);
    months[1] = "Jan";
    months[2] = "Feb";
    months[3] = "Mar";
    months[4] = "Apr";
    months[5] = "May";
    months[6] = "Jun";
```

```
months[7] = "Jul";
months[8] = "Aug";
months[9] = "Sep";
months[10] = "Oct";
months[11] = "Nov";
months[12] = "Dec";
var dateObj = new
       Date(document.lastModified)
var wday = days[dateObj.getDay()+1]
var lmonth =
months[dateObj.getMonth() + 1]
var date = dateObj.getDate()
if (date < 10) date = "0" + date
var fyear = dateObj.getYear()
document.write(wday + ", " + date +
"-" + lmonth + "-" + fyear)
-->
</SCRIPT>
```

Output: Monday, 17-Jul-2006



JavaScript Syntax: Function Declarations

1. Declaration Syntax
   - Functions are declared using the function reserved word
   - The return value is not declared, nor are the types of the arguments
   - Examples:

```
function square(x) {
  return(x * x);
}

function factorial(n) {
  if (n <= 0) {
    return(1);
  } else {
    return(n * factorial(n - 1));
  }
}
```

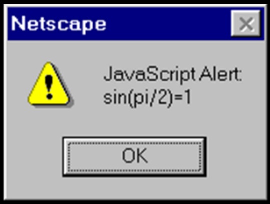
JavaScript Syntax: Function Declarations, cont.

2. First Class Functions

Functions can be passed and assigned to variables

Example

```
var fun = Math.sin;
alert("sin(pi/2)=" + fun(Math.PI/2));
```



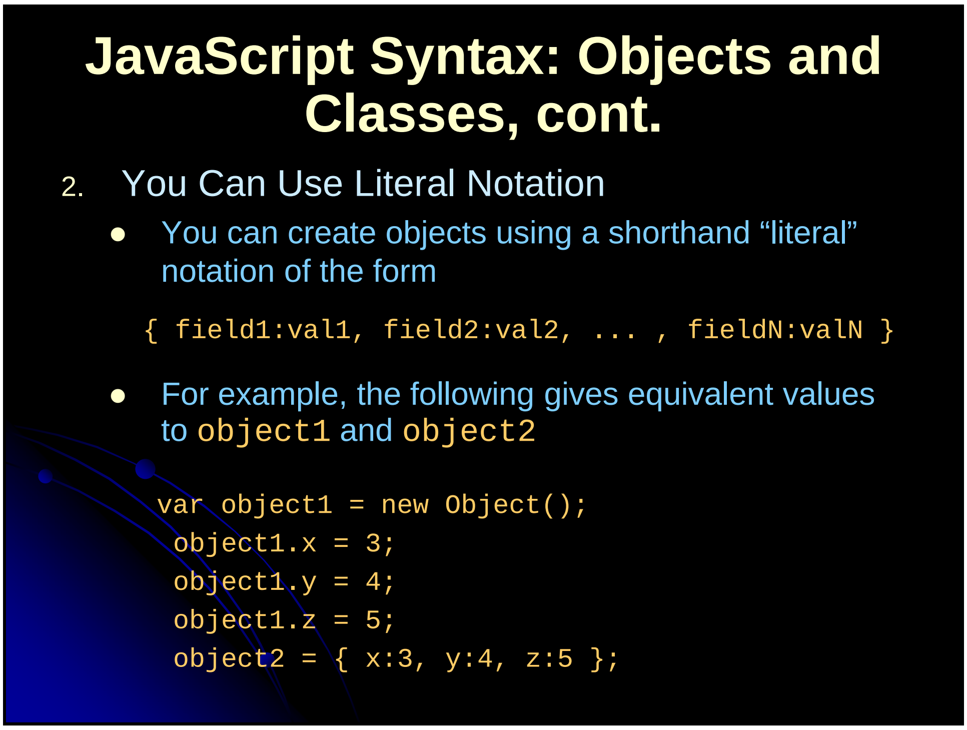
Netscape

JavaScript Alert:
sin(pi/2)=1

JavaScript Syntax: Objects and Classes

1. Fields Can Be Added On-the-Fly
   - Adding a new property (field) is a simple matter of assigning a value to one
   - If the field doesn’t already exist when you try to assign to it, JavaScript will create it automatically.
   - For instance:

```
var test = new Object();
test.field1 = "Value 1"; // Create field1 property
test.field2 = 7; // Create field2 property
```



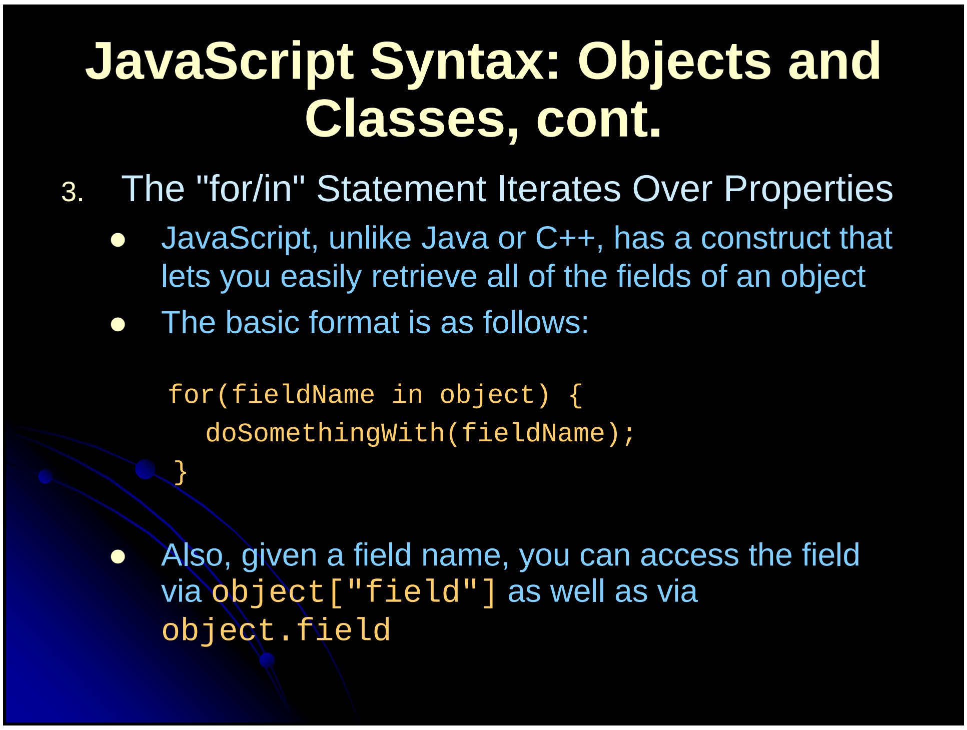
JavaScript Syntax: Objects and Classes, cont.

2. You Can Use Literal Notation
   - You can create objects using a shorthand “literal” notation of the form

```
{ field1:val1, field2:val2, ... , fieldN:valN }
```

   - For example, the following gives equivalent values to `object1` and `object2`

```
var object1 = new Object();
 object1.x = 3;
 object1.y = 4;
 object1.z = 5;
 object2 = { x:3, y:4, z:5 };
```

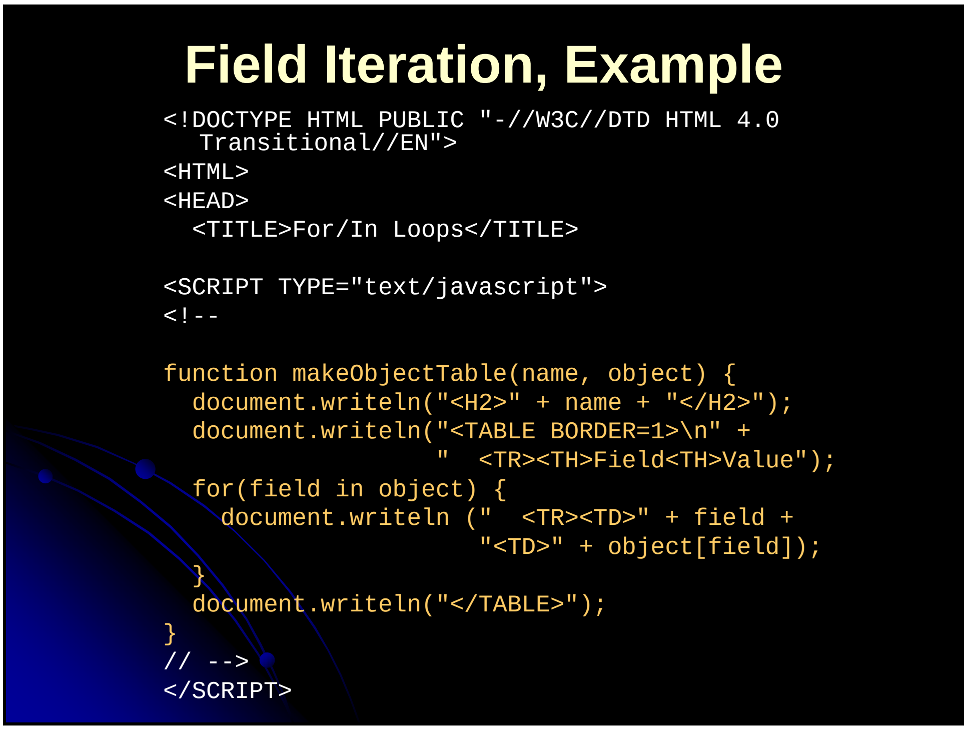


JavaScript Syntax: Objects and Classes, cont.

3. The "for/in" Statement Iterates Over Properties
   - JavaScript, unlike Java or C++, has a construct that lets you easily retrieve all of the fields of an object
   - The basic format is as follows:

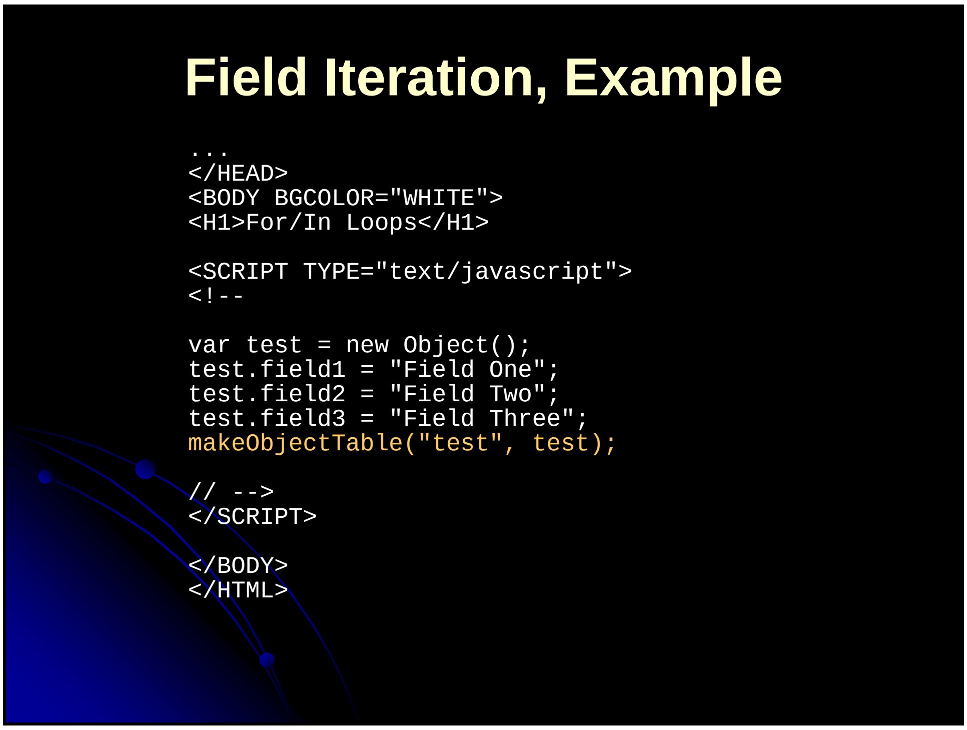
```
for(fieldName in object) {
  doSomethingWith(fieldName);
}
```

   - Also, given a field name, you can access the field via `object["field"]` as well as via `object.field`



Field Iteration, Example

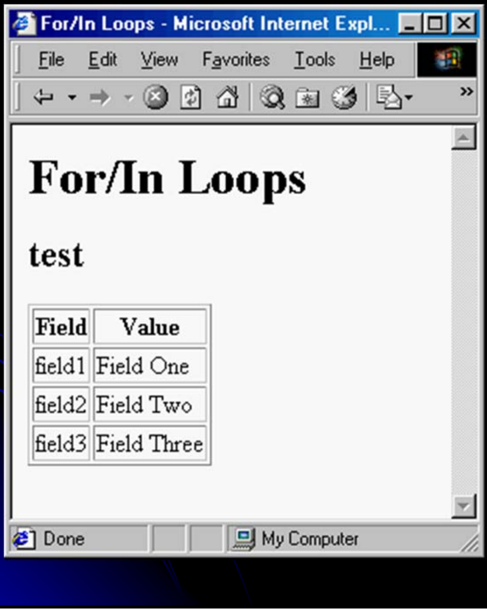
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
   Transitional//EN">
<HTML>
<HEAD>
  <TITLE>For/In Loops</TITLE>

<SCRIPT TYPE="text/javascript">
<!--

function makeObjectTable(name, object) {
  document.writeln("<H2>" + name + "</H2>");
  document.writeln("<TABLE BORDER=1>\n" +
                   "  <TR><TH>Field<TH>Value");
  for(field in object) {
    document.writeln ("  <TR><TD>" + field +
                      "<TD>" + object[field]);
  }
  document.writeln("</TABLE>");
}
// -->
</SCRIPT>
```



Field Iteration, Example

```
...
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>For/In Loops</H1>

<SCRIPT TYPE="text/javascript">
<!--

var test = new Object();
test.field1 = "Field One";
test.field2 = "Field Two";
test.field3 = "Field Three";
makeObjectTable("test", test);

// -->
</SCRIPT>

</BODY>
</HTML>
```

Field Iteration, Result


For/In Loops - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Done
My Computer

For/In Loops

test

Field	Value
field1	Field One
field2	Field Two
field3	Field Three

The for/in statement iterates over object properties

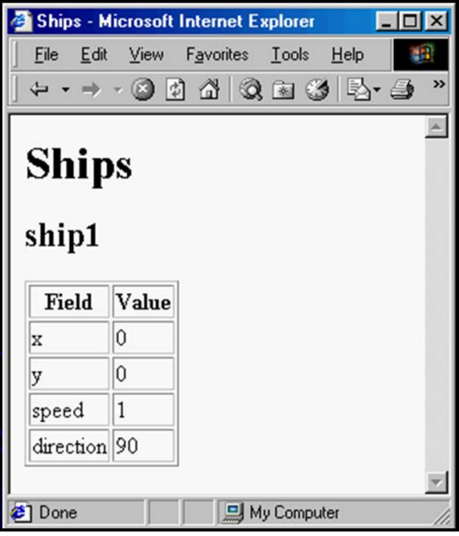
JavaScript Syntax: Objects and Classes

4. A “Constructor” is Just a Function that Assigns to “this”
   - JavaScript does not have an exact equivalent to Java’s class definition
   - The closest you get is when you define a function that assigns values to properties in the `this` reference
   - Calling this function using `new` binds `this` to a new `Object`
   - For example, following is a simple constructor for a `Ship` class

```
function Ship(x, y, speed, direction) {
   this.x = x;
   this.y = y;
   this.speed = speed;
   this.direction = direction;
}
```

Constructor, Example

```
var ship1 = new Ship(0, 0, 1, 90);
makeObjectTable("ship1", ship1);
```


Ships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Done
My Computer

Ships

ship1

Field	Value
x	0
y	0
speed	1
direction	90

JavaScript Syntax: Objects and Classes, cont.

5. Methods Are Function-Valued Properties
   - No special syntax for defining methods of objects
   - Instead, you simply assign a function to a property

Class Methods

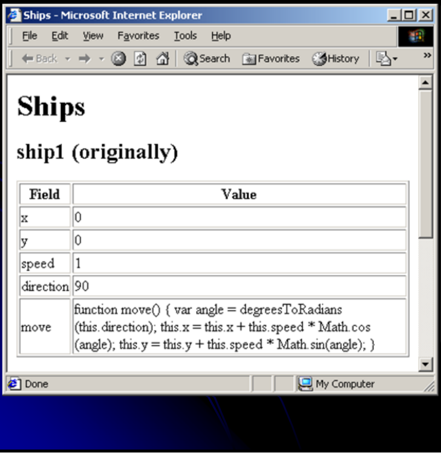
5. Methods Are Function-Valued Properties
   - No special syntax for defining methods of objects
   - Instead, you simply assign a function to a property

- Consider a version of the `Ship` class that includes a `move` method

```
function degreesToRadians(degrees) {
  return(degrees * Math.PI / 180.0);
}
function move() {
  var angle = degreesToRadians(this.direction);
  this.x = this.x + this.speed * Math.cos(angle);
  this.y = this.y + this.speed * Math.sin(angle);
}
function Ship(x, y, speed, direction) {
  this.x = x;
  this.y = y;
  this.speed = speed;
  this.direction = direction;
  this.move = move;
}
```

Class Methods, Result

```
var ship1 = new Ship(0, 0, 1, 90);
makeObjectTable("ship1 (originally)", ship1);
ship1.move();
makeObjectTable("ship1 (after move)", ship1);
```

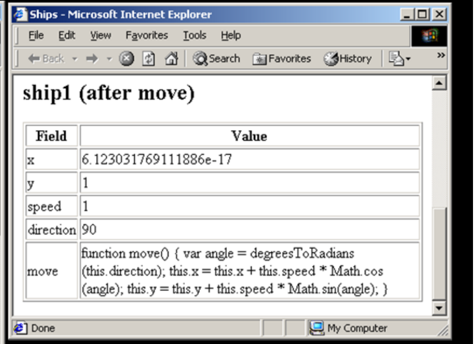


Ships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
My Computer

Ships

ship1 (originally)

Field	Value
x	0
y	0
speed	1
direction	90
move	function move() { var angle = degreesToRadians (this.direction); this.x = this.x + this.speed * Math.cos (angle); this.y = this.y + this.speed * Math.sin(angle); }



Ships - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
My Computer

ship1 (after move)

Field	Value
x	6.123031769111886e-17
y	1
speed	1
direction	90
move	function move() { var angle = degreesToRadians (this.direction); this.x = this.x + this.speed * Math.cos (angle); this.y = this.y + this.speed * Math.sin(angle); }

JavaScript Syntax: Objects and Classes, cont.

5. Arrays

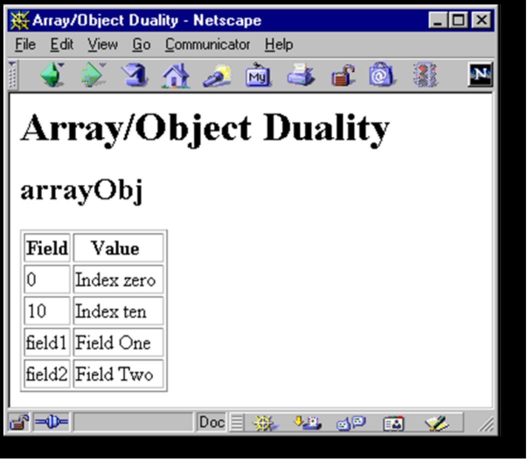
- For the most part, you can use arrays in JavaScript a lot like Java arrays.
  - Here are a few examples:

```
var squares = new Array(5);
for(var i=0; i<squares.length; i++) {
         vals[i] = i * i;
}
// Or, in one fell swoop:
var squares = new Array(0, 1, 4, 9, 16);
var array1 = new Array("fee", "fie", "fo", "fum");
// Literal Array notation for creating an array.
var array2 = [ "fee", "fie", "fo", "fum" ];
```

- Behind the scenes, however, JavaScript simply represents arrays as objects with numbered fields
  - You can access named fields using either `object.field` or `object["field"]`, but numbered fields only via `object[fieldNumber]`

Array, Example

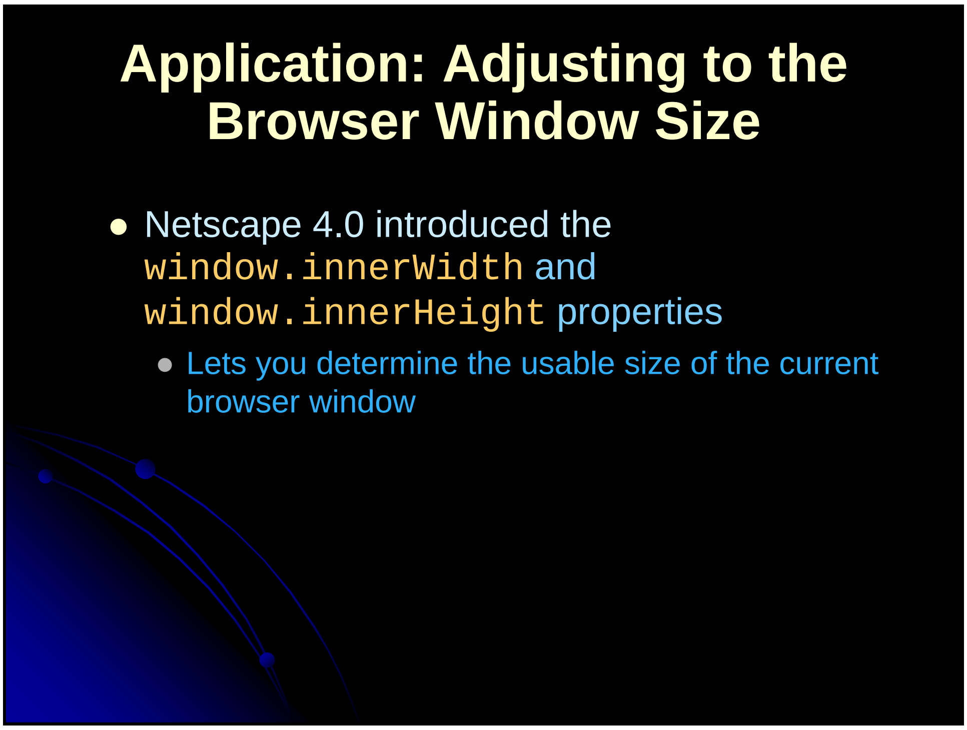
```
var arrayObj = new Object();
arrayObj[0] = "Index zero";
arrayObj[10] = "Index ten";
arrayObj.field1 = "Field One";
arrayObj["field2"] = "Field Two";

makeObjectTable("arrayObj",
                arrayObj);
```


Array/Object Duality - Netscape
File Edit View Go Communicator Help
Doc

Array/Object Duality

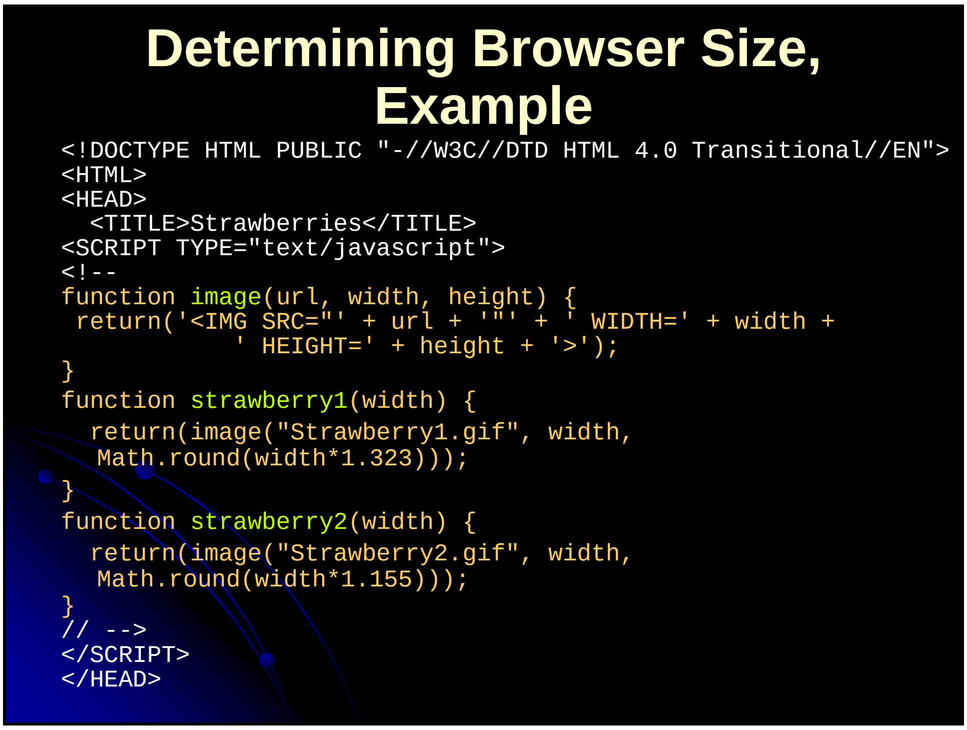
arrayObj

Field	Value
0	Index zero
10	Index ten
field1	Field One
field2	Field Two



Application: Adjusting to the Browser Window Size

- Netscape 4.0 introduced the `window.innerWidth` and `window.innerHeight` properties
  - Lets you determine the usable size of the current browser window



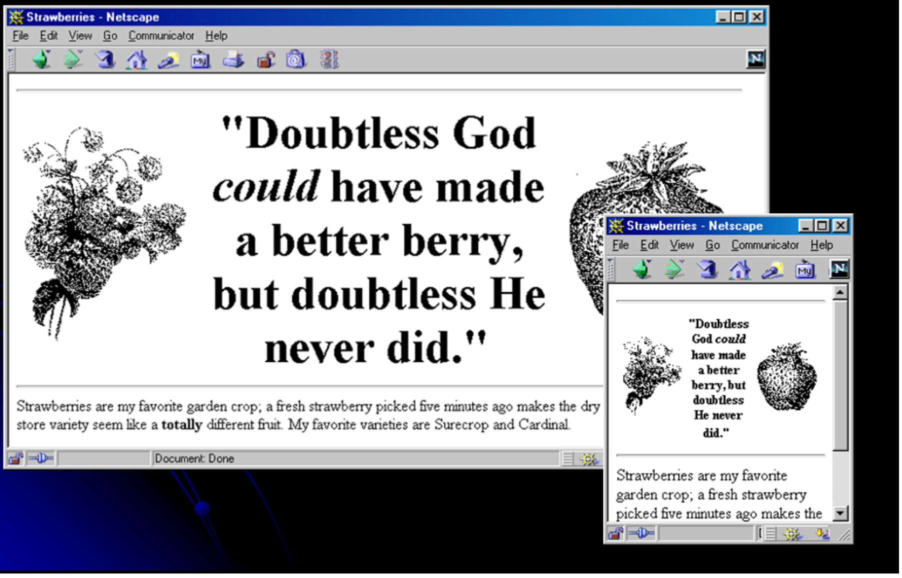
Determining Browser Size, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Strawberries</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function image(url, width, height) {
 return('<IMG SRC="' + url + '"' + ' WIDTH=' + width +
            ' HEIGHT=' + height + '>');
}
function strawberry1(width) {
  return(image("Strawberry1.gif", width,
   Math.round(width*1.323)));
}
function strawberry2(width) {
  return(image("Strawberry2.gif", width,
   Math.round(width*1.155)));
}
// -->
</SCRIPT>
</HEAD>
```

Determining Browser Size, Example, cont.

```
...
<SCRIPT TYPE="text/javascript">
<!--
  var imageWidth = window.innerWidth/4;
  var fontSize = Math.min(7,Math.round(window.innerWidth/100));

document.writeln
  ('<TABLE>\n' +
   '  <TR><TD>' + strawberry1(imageWidth) + '\n' +
   '      <TH><FONT SIZE=' + fontSize + '>\n' +
   '          "Doubtless God <I>could</I> have made\n' +
   '          a better berry, but doubtless He\n' +
   '          never did."</FONT>\n' +
   '      <TD>'  + strawberry2(imageWidth) + '\n' +
   '</TABLE>');
// -->
</SCRIPT>
<HR>

Strawberries are my favorite garden crop; a fresh ...
</BODY>
</HTML>
```

Determining Browser Size, Results

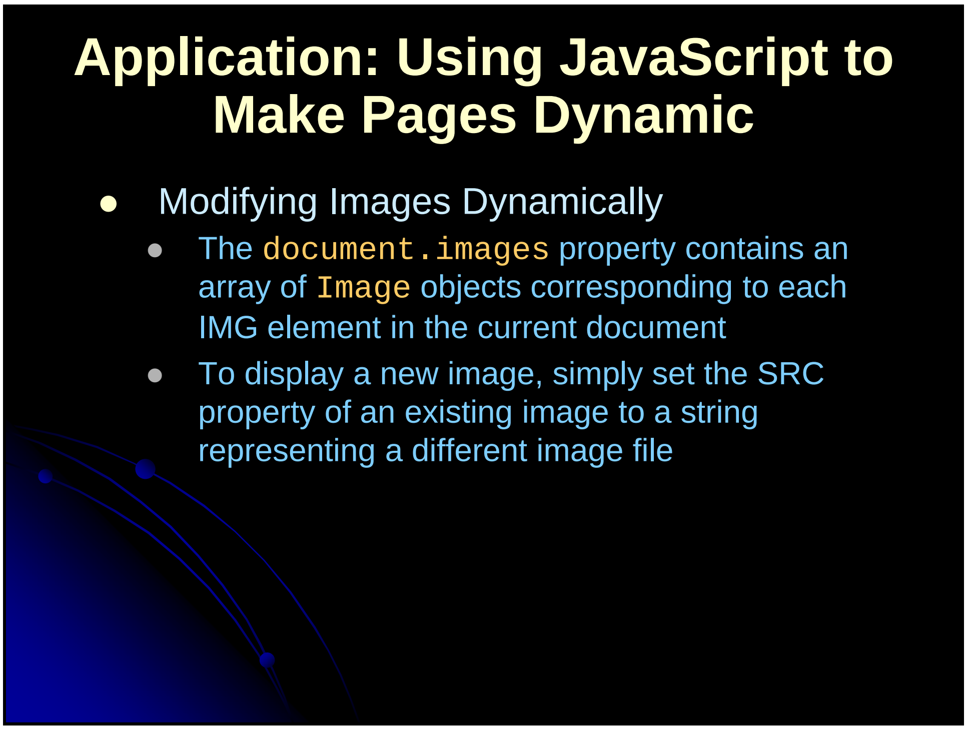

Strawberries - Netscape
File Edit View Go Communicator Help
Document: Done
Strawberries - Netscape
File Edit View Go Communicator Help

"Doubtless God *could* have made a better berry, but doubtless He never did."

Strawberries are my favorite garden crop; a fresh strawberry picked five minutes ago makes the dry store variety seem like a **totally** different fruit. My favorite varieties are Surecrop and Cardinal.

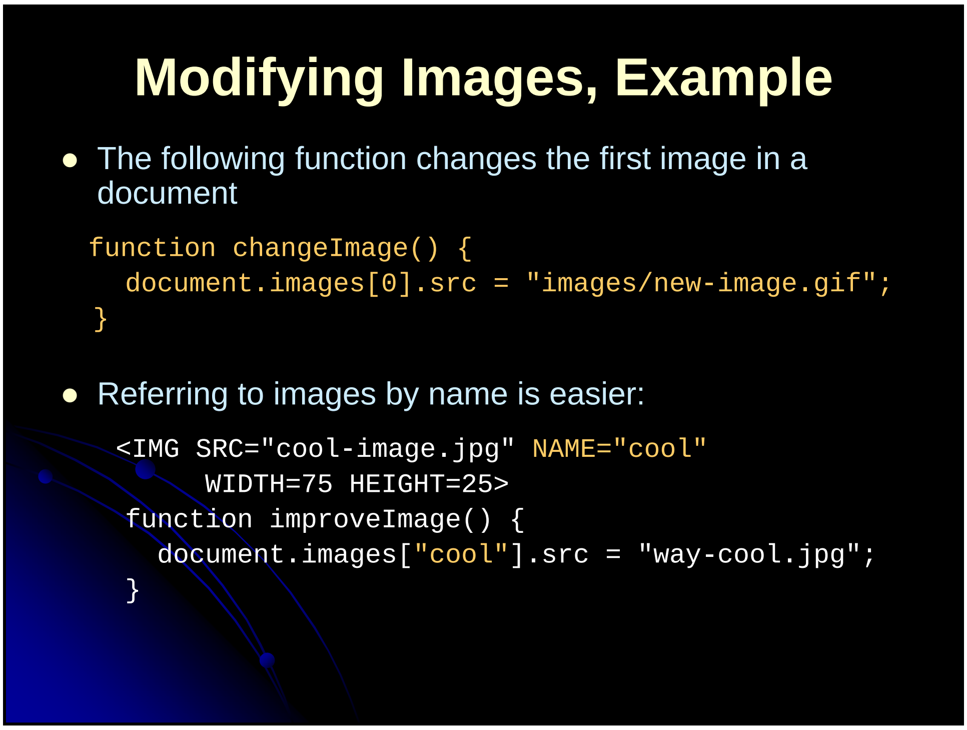
"Doubtless God *could* have made a better berry, but doubtless He never did."

Strawberries are my favorite garden crop; a fresh strawberry picked five minutes ago makes the



Application: Using JavaScript to Make Pages Dynamic

- Modifying Images Dynamically
  - The `document.images` property contains an array of `Image` objects corresponding to each IMG element in the current document
  - To display a new image, simply set the SRC property of an existing image to a string representing a different image file



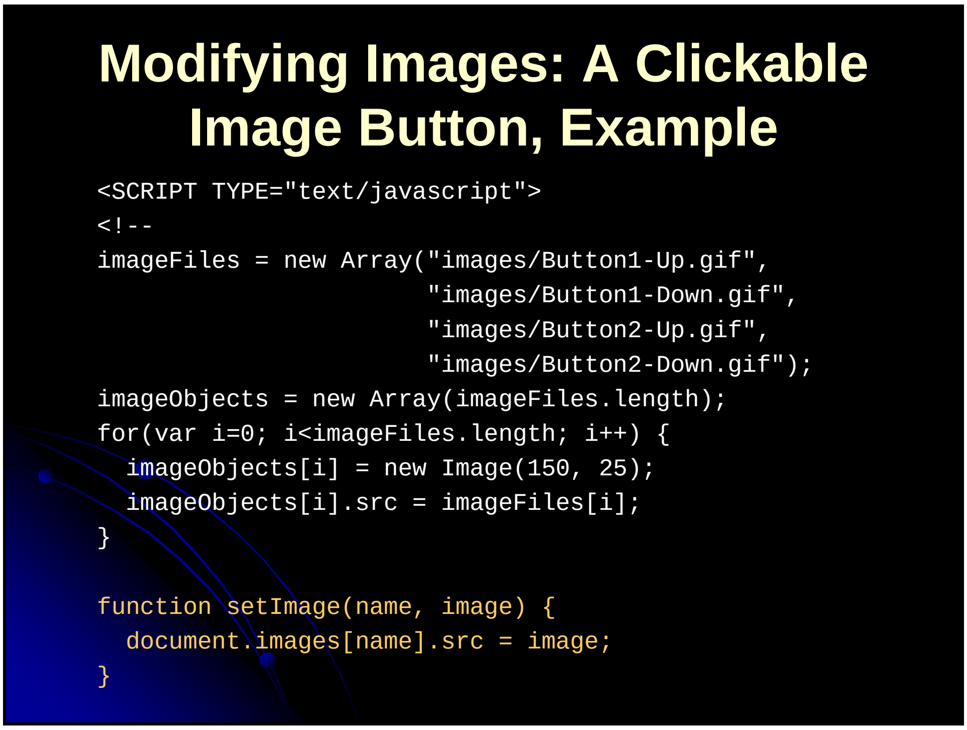
Modifying Images, Example

- The following function changes the first image in a document

```
function changeImage() {
  document.images[0].src = "images/new-image.gif";
}
```

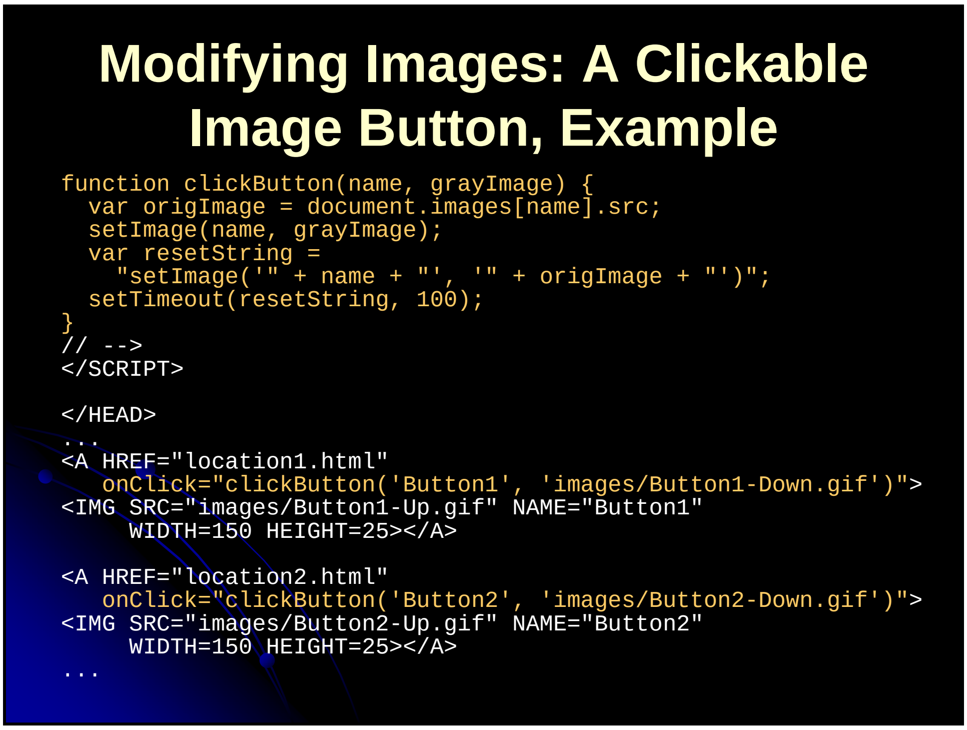
- Referring to images by name is easier:

```
<IMG SRC="cool-image.jpg" NAME="cool"
     WIDTH=75 HEIGHT=25>
function improveImage() {
  document.images["cool"].src = "way-cool.jpg";
}
```



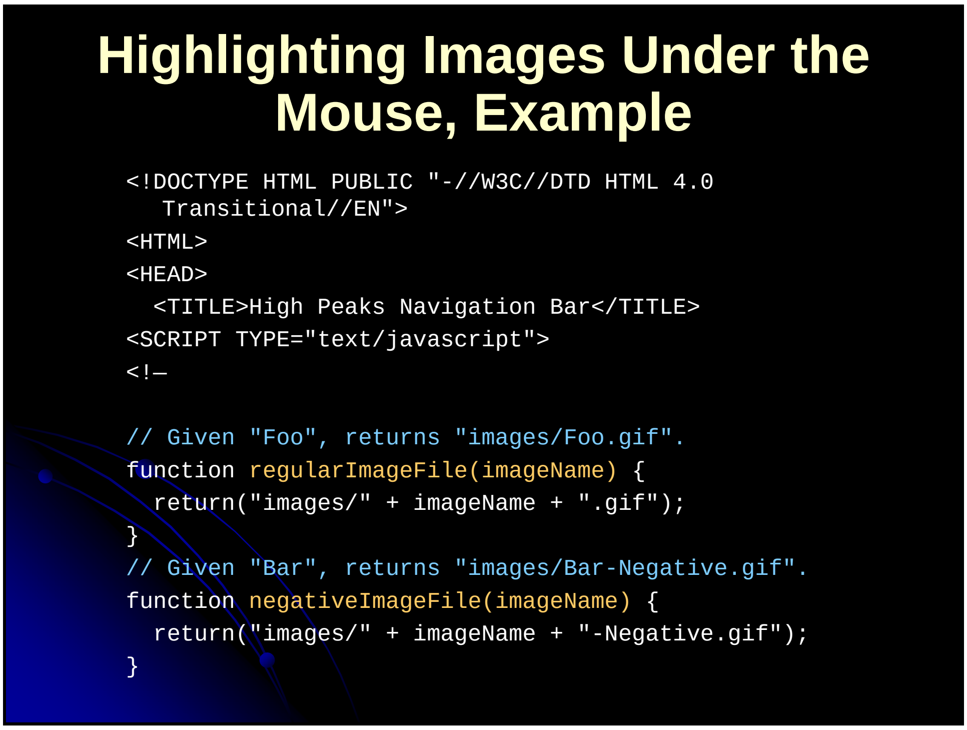
Modifying Images: A Clickable Image Button, Example

```
<SCRIPT TYPE="text/javascript">
<!--
imageFiles = new Array("images/Button1-Up.gif",
                       "images/Button1-Down.gif",
                       "images/Button2-Up.gif",
                       "images/Button2-Down.gif");
imageObjects = new Array(imageFiles.length);
for(var i=0; i<imageFiles.length; i++) {
  imageObjects[i] = new Image(150, 25);
  imageObjects[i].src = imageFiles[i];
}

function setImage(name, image) {
  document.images[name].src = image;
}
```



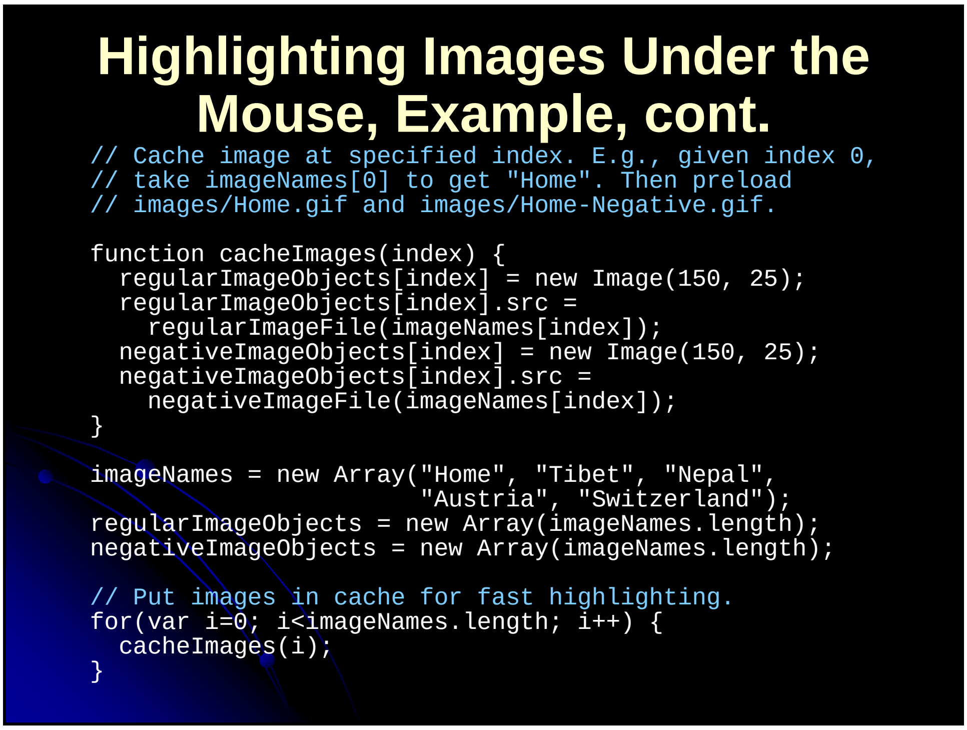
Modifying Images: A Clickable Image Button, Example

```
function clickButton(name, grayImage) {
  var origImage = document.images[name].src;
  setImage(name, grayImage);
  var resetString =
    "setImage('" + name + "', '" + origImage + "')";
  setTimeout(resetString, 100);
}
// -->
</SCRIPT>

</HEAD>
...
<A HREF="location1.html"
   onClick="clickButton('Button1', 'images/Button1-Down.gif')">
<IMG SRC="images/Button1-Up.gif" NAME="Button1"
     WIDTH=150 HEIGHT=25></A>

<A HREF="location2.html"
   onClick="clickButton('Button2', 'images/Button2-Down.gif')">
<IMG SRC="images/Button2-Up.gif" NAME="Button2"
     WIDTH=150 HEIGHT=25></A>
...
```



Highlighting Images Under the Mouse, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
   Transitional//EN">
<HTML>
<HEAD>
  <TITLE>High Peaks Navigation Bar</TITLE>
<SCRIPT TYPE="text/javascript">
<!–

// Given "Foo", returns "images/Foo.gif".
function regularImageFile(imageName) {
  return("images/" + imageName + ".gif");
}
// Given "Bar", returns "images/Bar-Negative.gif".
function negativeImageFile(imageName) {
  return("images/" + imageName + "-Negative.gif");
}
```



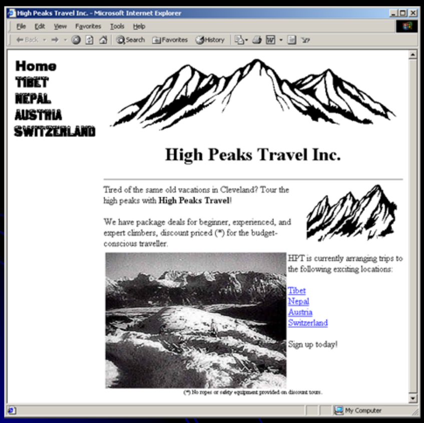
Highlighting Images Under the Mouse, Example, cont.

```
// Cache image at specified index. E.g., given index 0,
// take imageNames[0] to get "Home". Then preload
// images/Home.gif and images/Home-Negative.gif.

function cacheImages(index) {
  regularImageObjects[index] = new Image(150, 25);
  regularImageObjects[index].src =
    regularImageFile(imageNames[index]);
  negativeImageObjects[index] = new Image(150, 25);
  negativeImageObjects[index].src =
    negativeImageFile(imageNames[index]);
}

imageNames = new Array("Home", "Tibet", "Nepal",
                       "Austria", "Switzerland");
regularImageObjects = new Array(imageNames.length);
negativeImageObjects = new Array(imageNames.length);

// Put images in cache for fast highlighting.
for(var i=0; i<imageNames.length; i++) {
  cacheImages(i);
}
```

Highlighting Images Under the Mouse, Example, cont.

```
...
function highlight(imageName) {
  document.images[imageName].src = negativeImageFile(imageName);
}
function unHighlight(imageName) {
  document.images[imageName].src = regularImageFile(imageName);
}
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<TABLE BORDER=0 WIDTH=150 BGCOLOR="WHITE"
       CELLPADDING=0 CELLSPACING=0>
 <TR><TD><A HREF="Tibet.html"
            TARGET="Main"
            onMouseOver="highlight('Tibet')"
            onMouseOut="unHighlight('Tibet')">
            <IMG SRC="images/Tibet.gif"
                 NAME="Tibet"
                 WIDTH=150 HEIGHT=25 BORDER=0>
         </A>
...
```

Highlighting Images Under the Mouse, Result


High Peaks Travel Inc. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Home
TIBET
NEPAL
AUSTRIA
SWITZERLAND
My Computer

High Peaks Travel Inc.

Tired of the same old vacations in Cleveland? Tour the high peaks with **High Peaks Travel**!

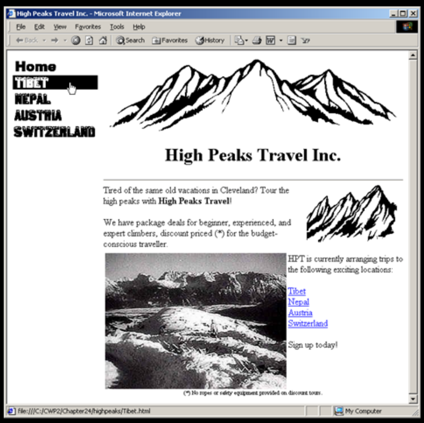
We have package deals for beginner, experienced, and expert climbers, discount priced (*) for the budget-conscious traveller.

HPT is currently arranging trips to the following exciting locations:

Tibet
Nepal
Austria
Switzerland

Sign up today!

(*) No ropes or safety equipment provided on discount tours.


High Peaks Travel Inc. - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Home
TIBET
NEPAL
AUSTRIA
SWITZERLAND
file:///C:/CWP2/Chapter24/highpeaks/Tibet.html
My Computer

High Peaks Travel Inc.

Tired of the same old vacations in Cleveland? Tour the high peaks with **High Peaks Travel**!

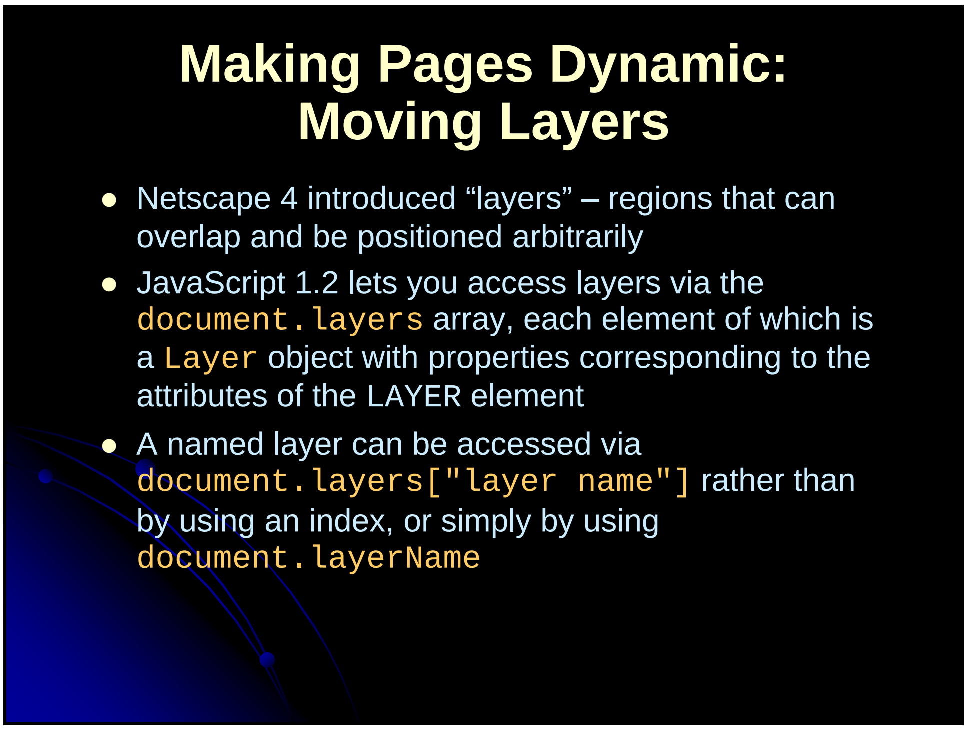
We have package deals for beginner, experienced, and expert climbers, discount priced (*) for the budget-conscious traveller.

HPT is currently arranging trips to the following exciting locations:

Tibet
Nepal
Austria
Switzerland

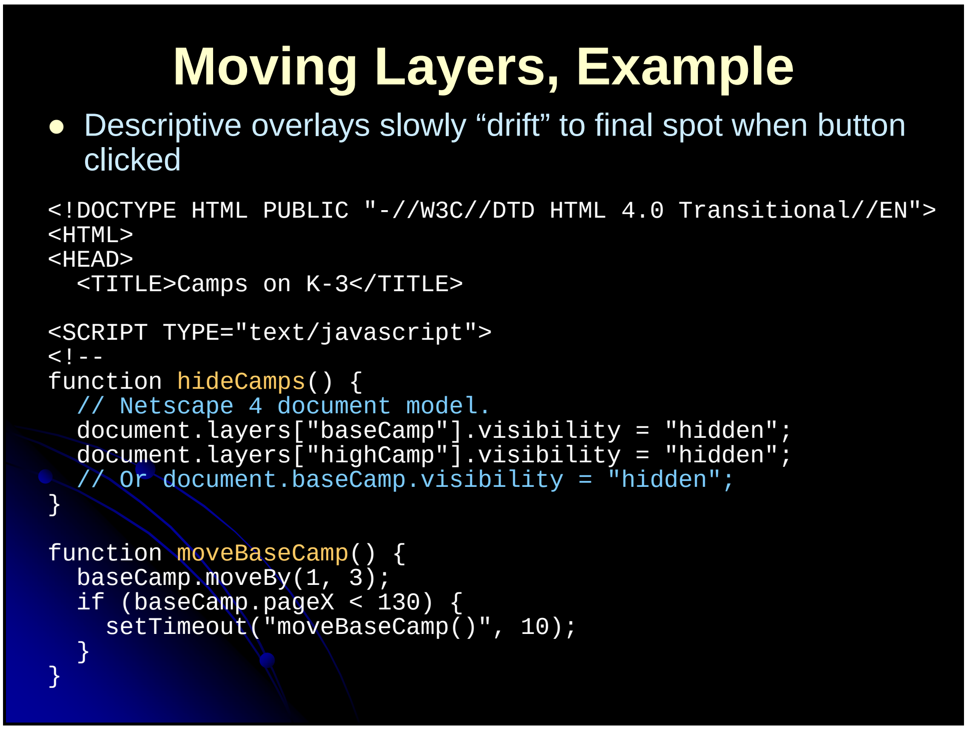
Sign up today!

(*) No ropes or safety equipment provided on discount tours.



Making Pages Dynamic: Moving Layers

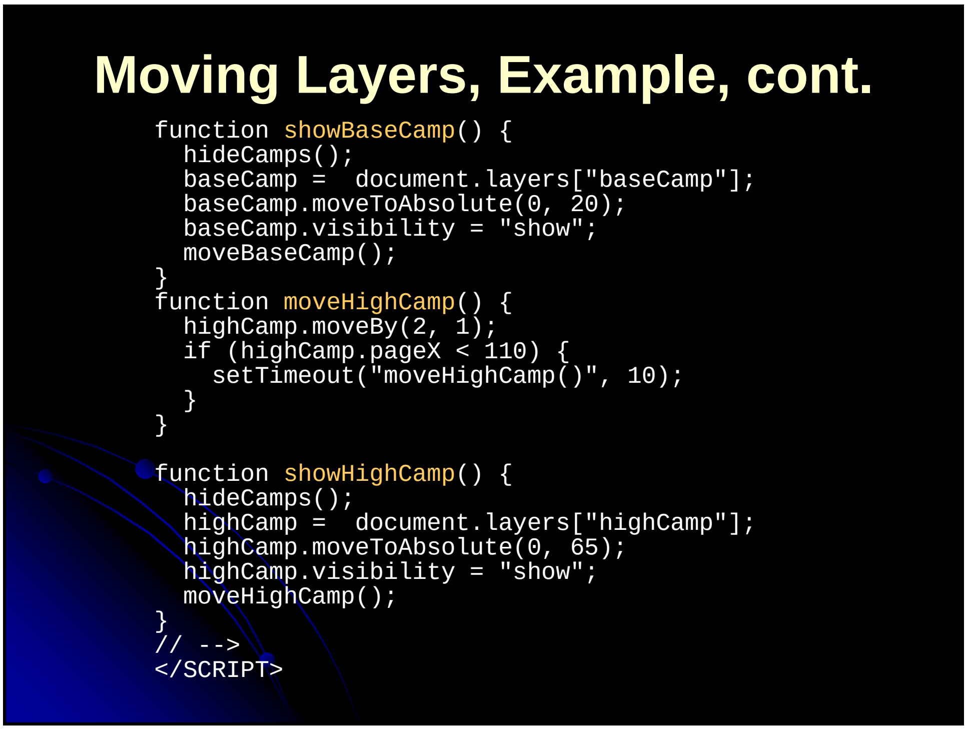
- Netscape 4 introduced “layers” – regions that can overlap and be positioned arbitrarily
- JavaScript 1.2 lets you access layers via the `document.layers` array, each element of which is a `Layer` object with properties corresponding to the attributes of the LAYER element
- A named layer can be accessed via `document.layers["layer name"]` rather than by using an index, or simply by using `document.layerName`



Moving Layers, Example

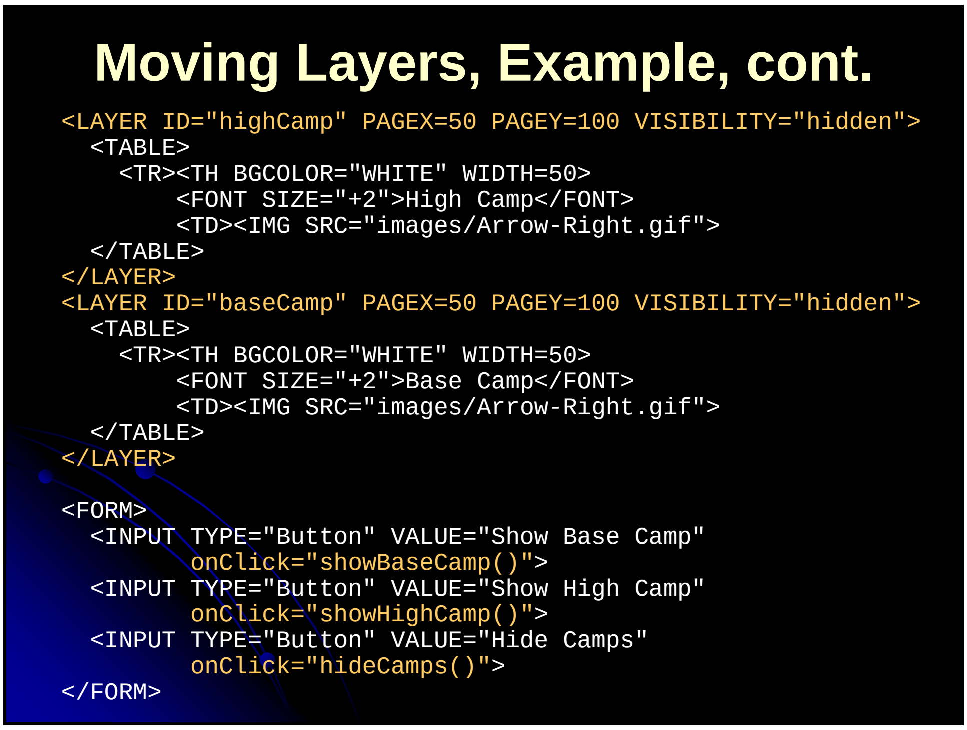
- Descriptive overlays slowly “drift” to final spot when button clicked

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camps on K-3</TITLE>

<SCRIPT TYPE="text/javascript">
<!--
function hideCamps() {
  // Netscape 4 document model.
  document.layers["baseCamp"].visibility = "hidden";
  document.layers["highCamp"].visibility = "hidden";
  // Or document.baseCamp.visibility = "hidden";
}

function moveBaseCamp() {
  baseCamp.moveBy(1, 3);
  if (baseCamp.pageX < 130) {
    setTimeout("moveBaseCamp()", 10);
  }
}
```



Moving Layers, Example, cont.

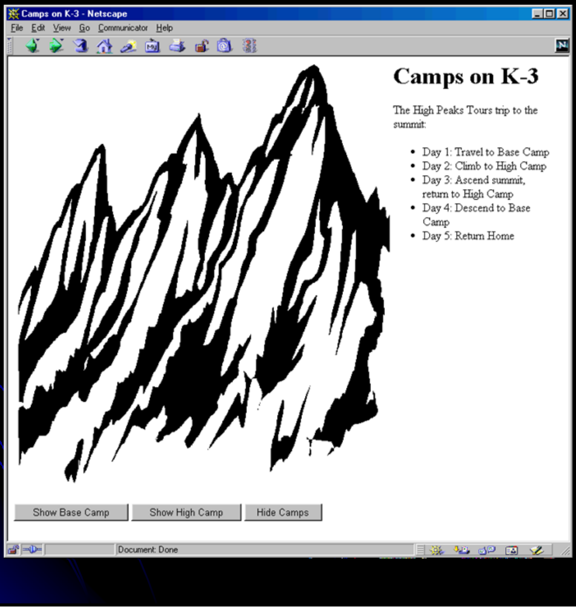
```
function showBaseCamp() {
  hideCamps();
  baseCamp =  document.layers["baseCamp"];
  baseCamp.moveToAbsolute(0, 20);
  baseCamp.visibility = "show";
  moveBaseCamp();
}
function moveHighCamp() {
  highCamp.moveBy(2, 1);
  if (highCamp.pageX < 110) {
    setTimeout("moveHighCamp()", 10);
  }
}

function showHighCamp() {
  hideCamps();
  highCamp =  document.layers["highCamp"];
  highCamp.moveToAbsolute(0, 65);
  highCamp.visibility = "show";
  moveHighCamp();
}
// -->
</SCRIPT>
```



Moving Layers, Example, cont.

```
<LAYER ID="highCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
  <TABLE>
    <TR><TH BGCOLOR="WHITE" WIDTH=50>
        <FONT SIZE="+2">High Camp</FONT>
        <TD><IMG SRC="images/Arrow-Right.gif">
  </TABLE>
</LAYER>
<LAYER ID="baseCamp" PAGEX=50 PAGEY=100 VISIBILITY="hidden">
  <TABLE>
    <TR><TH BGCOLOR="WHITE" WIDTH=50>
        <FONT SIZE="+2">Base Camp</FONT>
        <TD><IMG SRC="images/Arrow-Right.gif">
  </TABLE>
</LAYER>

<FORM>
  <INPUT TYPE="Button" VALUE="Show Base Camp"
         onClick="showBaseCamp()">
  <INPUT TYPE="Button" VALUE="Show High Camp"
         onClick="showHighCamp()">
  <INPUT TYPE="Button" VALUE="Hide Camps"
         onClick="hideCamps()">
</FORM>
```

Moving Layers, Result

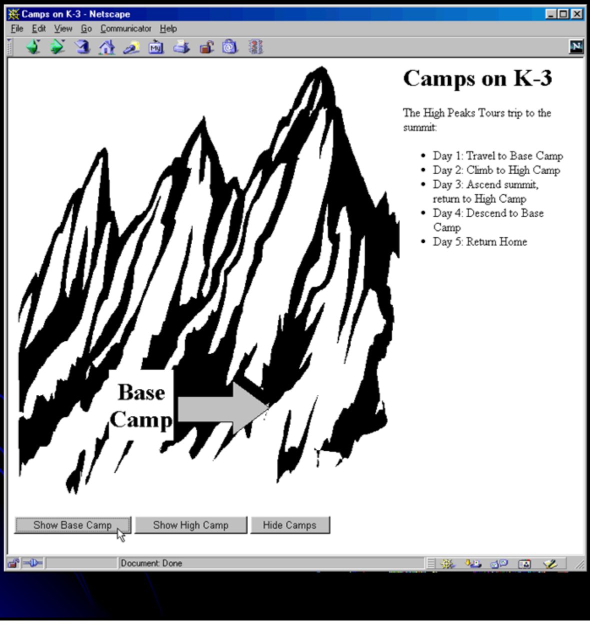


Camps on K-3

The High Peaks Tours trip to the summit:

- Day 1: Travel to Base Camp
- Day 2: Climb to High Camp
- Day 3: Ascend summit, return to High Camp
- Day 4: Descend to Base Camp
- Day 5: Return Home

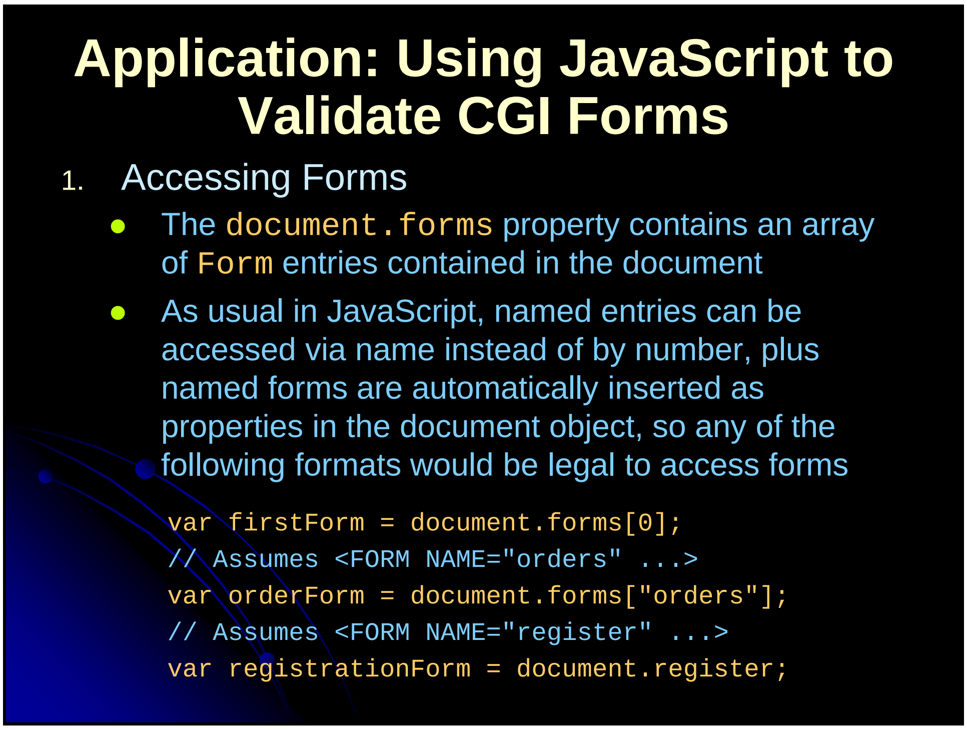
Moving Layers, Result



Camps on K-3

The High Peaks Tours trip to the summit:

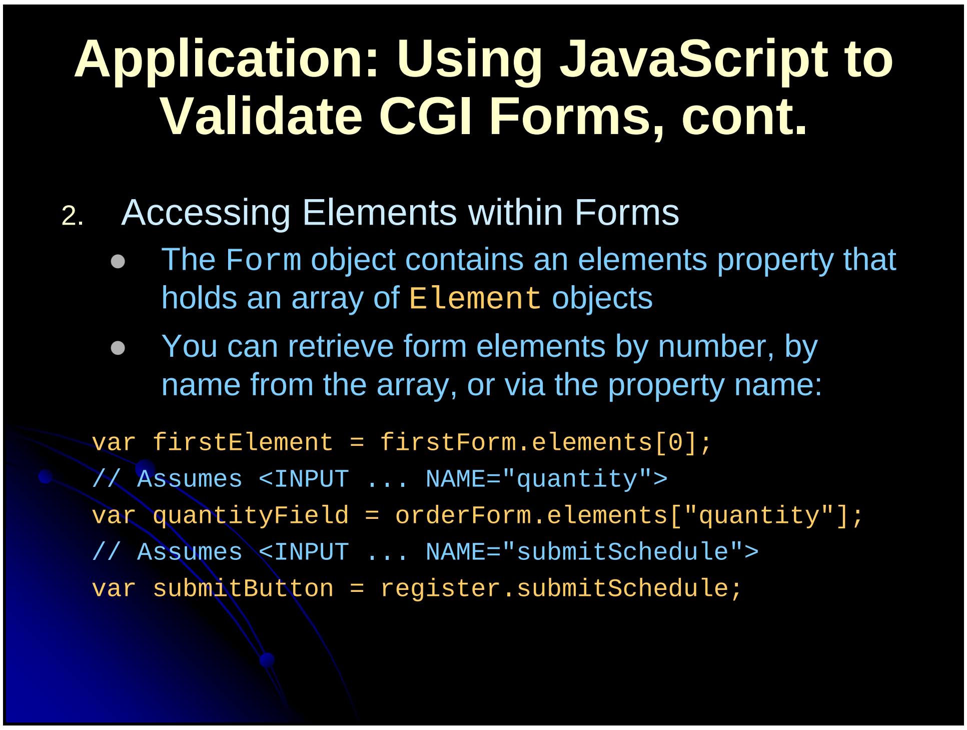
- Day 1: Travel to Base Camp
- Day 2: Climb to High Camp
- Day 3: Ascend summit, return to High Camp
- Day 4: Descend to Base Camp
- Day 5: Return Home



Application: Using JavaScript to Validate CGI Forms

1. Accessing Forms
   - The `document.forms` property contains an array of `Form` entries contained in the document
   - As usual in JavaScript, named entries can be accessed via name instead of by number, plus named forms are automatically inserted as properties in the document object, so any of the following formats would be legal to access forms

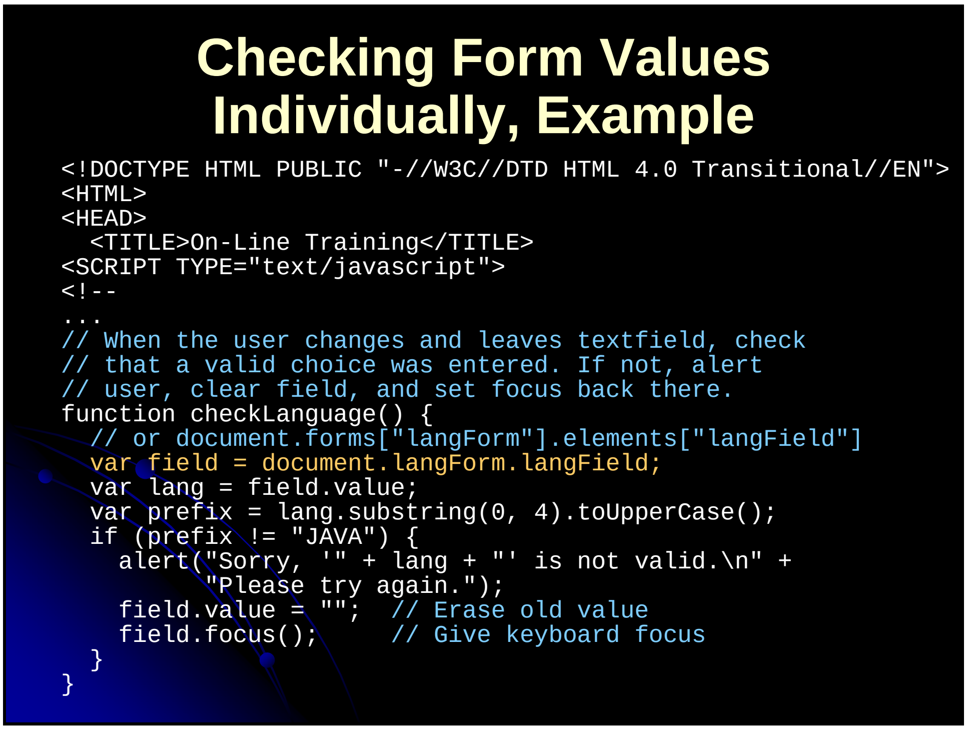
```
var firstForm = document.forms[0];
// Assumes <FORM NAME="orders" ...>
var orderForm = document.forms["orders"];
// Assumes <FORM NAME="register" ...>
var registrationForm = document.register;
```



Application: Using JavaScript to Validate CGI Forms, cont.

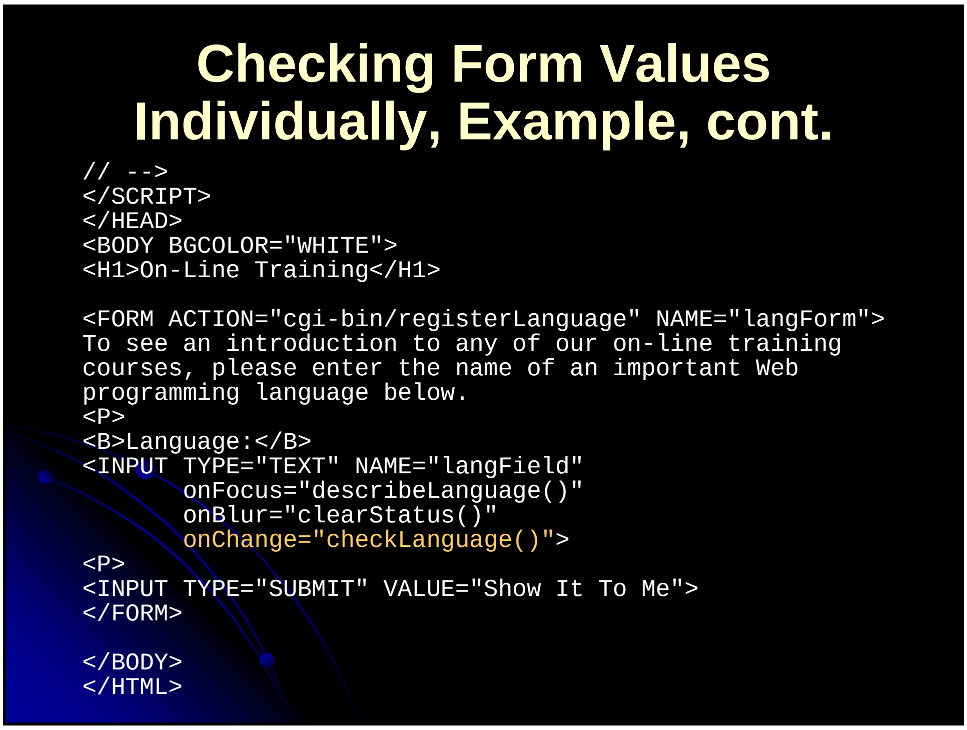
2. Accessing Elements within Forms
   - The `Form` object contains an elements property that holds an array of `Element` objects
   - You can retrieve form elements by number, by name from the array, or via the property name:

```
var firstElement = firstForm.elements[0];
// Assumes <INPUT ... NAME="quantity">
var quantityField = orderForm.elements["quantity"];
// Assumes <INPUT ... NAME="submitSchedule">
var submitButton = register.submitSchedule;
```



Checking Form Values Individually, Example

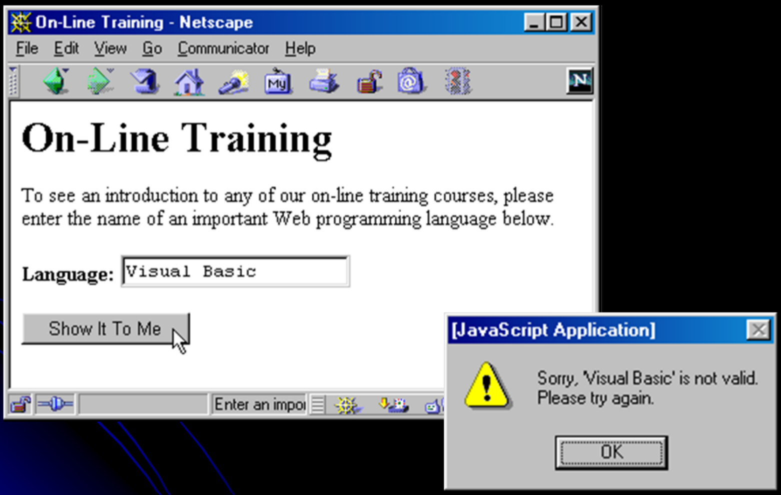
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>On-Line Training</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
...
// When the user changes and leaves textfield, check
// that a valid choice was entered. If not, alert
// user, clear field, and set focus back there.
function checkLanguage() {
  // or document.forms["langForm"].elements["langField"]
  var field = document.langForm.langField;
  var lang = field.value;
  var prefix = lang.substring(0, 4).toUpperCase();
  if (prefix != "JAVA") {
    alert("Sorry, '" + lang + "' is not valid.\n" +
          "Please try again.");
    field.value = "";  // Erase old value
    field.focus();     // Give keyboard focus
  }
}
```



Checking Form Values Individually, Example, cont.

```
// -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR="WHITE">
<H1>On-Line Training</H1>

<FORM ACTION="cgi-bin/registerLanguage" NAME="langForm">
To see an introduction to any of our on-line training
courses, please enter the name of an important Web
programming language below.
<P>
<B>Language:</B>
<INPUT TYPE="TEXT" NAME="langField"
       onFocus="describeLanguage()"
       onBlur="clearStatus()"
       onChange="checkLanguage()">
<P>
<INPUT TYPE="SUBMIT" VALUE="Show It To Me">
</FORM>

</BODY>
</HTML>
```

Checking Form Values Individually, Results


On-Line Training - Netscape
File Edit View Go Communicator Help
Show It To Me
Enter an impo
[JavaScript Application]
OK

On-Line Training

To see an introduction to any of our on-line training courses, please enter the name of an important Web programming language below.

Language: Visual Basic

Sorry, 'Visual Basic' is not valid.
Please try again.

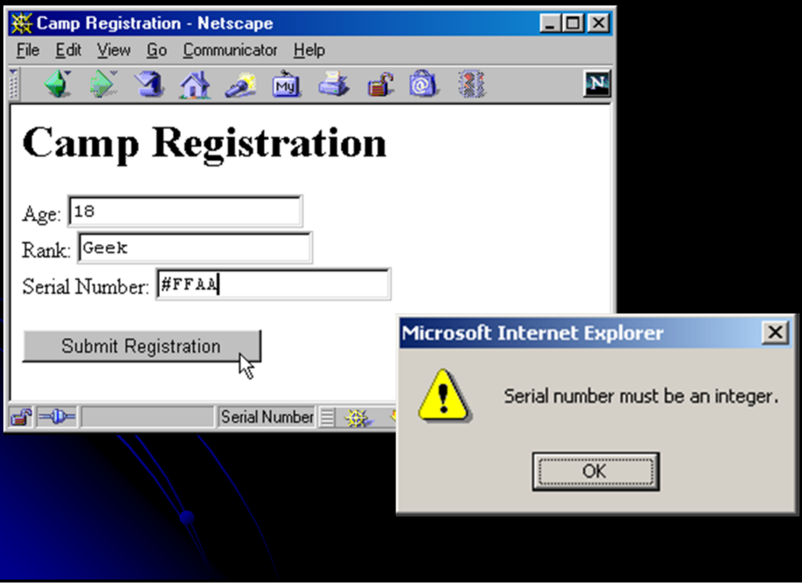
Checking Values When Form is Submitted, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Camp Registration</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function isInt(string) {
  var val = parseInt(string);
  return(val > 0);
}
function checkRegistration() {
  var ageField = document.registerForm.ageField;
  if (!isInt(ageField.value)) {
    alert("Age must be an integer.");
    return(false);
  }
  ...
  // Format looks OK. Submit form.
  return(true);
}
// -->
</SCRIPT>
```

Checking Values When Form is Submitted, Example, cont.

```
<BODY BGCOLOR="WHITE">
<H1>Camp Registration</H1>

<FORM ACTION="cgi-bin/register"
      NAME="registerForm"
      onSubmit="return(checkRegistration())">
Age: <INPUT TYPE="TEXT" NAME="ageField"
            onFocus="promptAge()"
            onBlur="clearStatus()">
<BR>
Rank: <INPUT TYPE="TEXT" NAME="rankField"
             onFocus="promptRank()"
             onBlur="clearStatus()">
<BR>
Serial Number: <INPUT TYPE="TEXT" NAME="serialField"
                      onFocus="promptSerial()"
                      onBlur="clearStatus()">
<P>
<INPUT TYPE="SUBMIT" VALUE="Submit Registration">
</FORM>

</BODY>
</HTML>
```

Checking Values When Form is Submitted, Results



Camp Registration - Netscape
File Edit View Go Communicator Help
Serial Number

Camp Registration

Age: 18

Rank: Geek

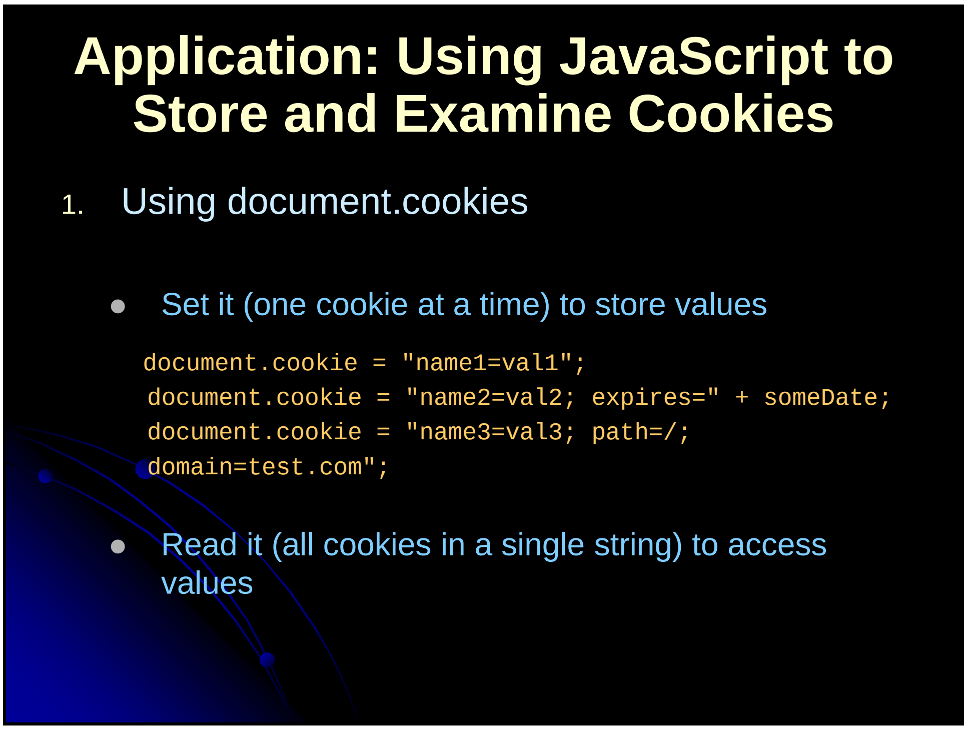
Serial Number: #FFAA

Submit Registration

Microsoft Internet Explorer

Serial number must be an integer.

OK



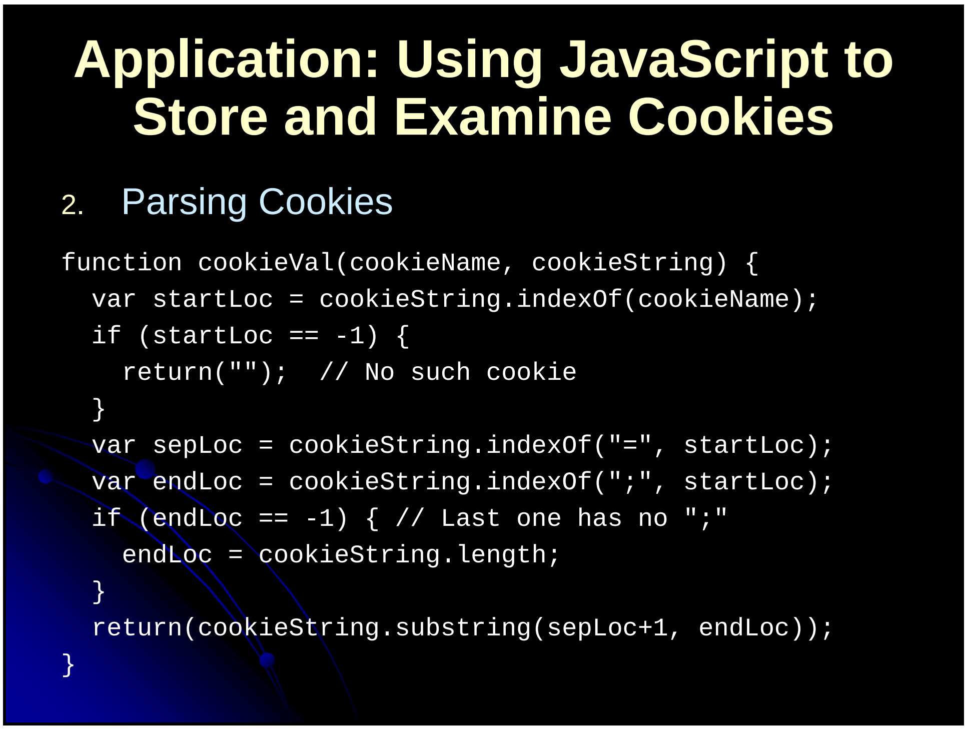
Application: Using JavaScript to Store and Examine Cookies

1. Using document.cookies

- Set it (one cookie at a time) to store values

```
document.cookie = "name1=val1";
document.cookie = "name2=val2; expires=" + someDate;
document.cookie = "name3=val3; path=/;
domain=test.com";
```

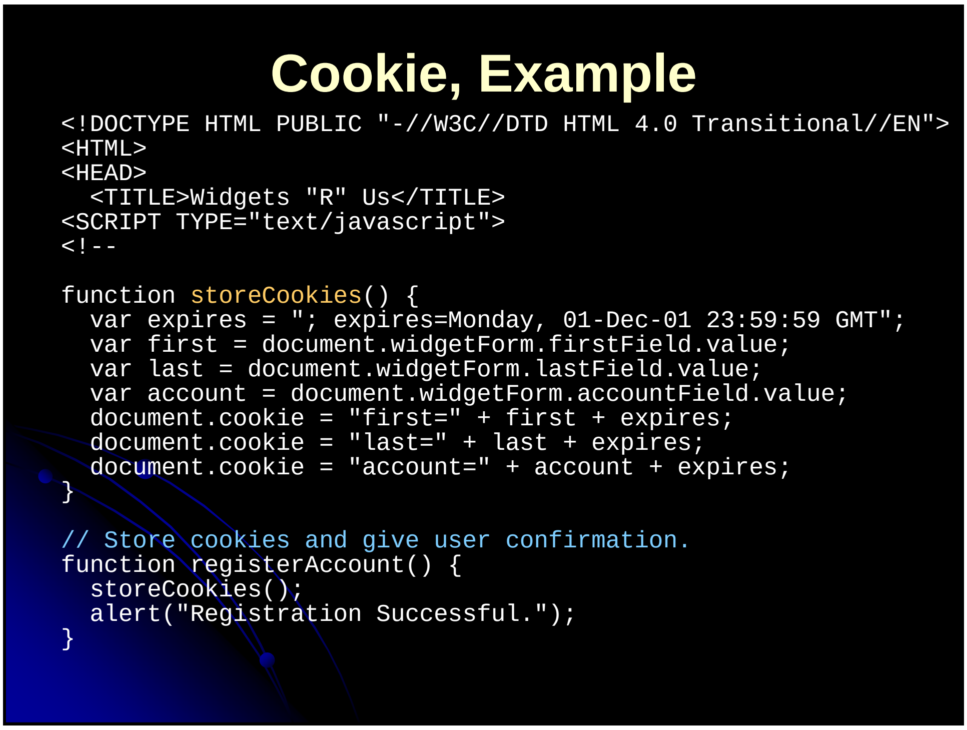
- Read it (all cookies in a single string) to access values



Application: Using JavaScript to Store and Examine Cookies

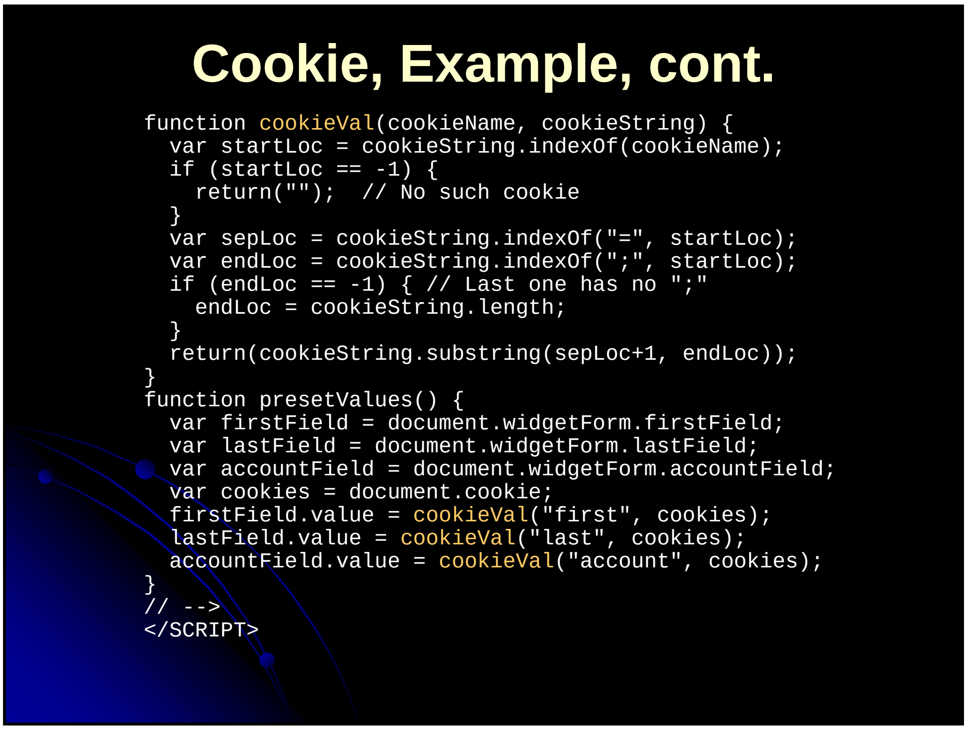
2. Parsing Cookies

```
function cookieVal(cookieName, cookieString) {
  var startLoc = cookieString.indexOf(cookieName);
  if (startLoc == -1) {
    return("");  // No such cookie
  }
  var sepLoc = cookieString.indexOf("=", startLoc);
  var endLoc = cookieString.indexOf(";", startLoc);
  if (endLoc == -1) { // Last one has no ";"
    endLoc = cookieString.length;
  }
  return(cookieString.substring(sepLoc+1, endLoc));
}
```



Cookie, Example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Widgets "R" Us</TITLE>
<SCRIPT TYPE="text/javascript">
<!--

function storeCookies() {
  var expires = "; expires=Monday, 01-Dec-01 23:59:59 GMT";
  var first = document.widgetForm.firstField.value;
  var last = document.widgetForm.lastField.value;
  var account = document.widgetForm.accountField.value;
  document.cookie = "first=" + first + expires;
  document.cookie = "last=" + last + expires;
  document.cookie = "account=" + account + expires;
}

// Store cookies and give user confirmation.
function registerAccount() {
  storeCookies();
  alert("Registration Successful.");
}
```



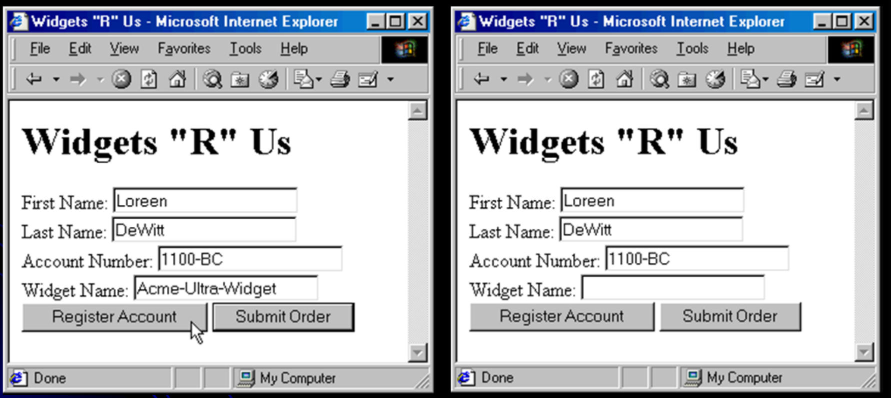
Cookie, Example, cont.

```
function cookieVal(cookieName, cookieString) {
  var startLoc = cookieString.indexOf(cookieName);
  if (startLoc == -1) {
    return("");  // No such cookie
  }
  var sepLoc = cookieString.indexOf("=", startLoc);
  var endLoc = cookieString.indexOf(";", startLoc);
  if (endLoc == -1) { // Last one has no ";"
    endLoc = cookieString.length;
  }
  return(cookieString.substring(sepLoc+1, endLoc));
}
function presetValues() {
  var firstField = document.widgetForm.firstField;
  var lastField = document.widgetForm.lastField;
  var accountField = document.widgetForm.accountField;
  var cookies = document.cookie;
  firstField.value = cookieVal("first", cookies);
  lastField.value = cookieVal("last", cookies);
  accountField.value = cookieVal("account", cookies);
}
// -->
</SCRIPT>
```

Cookie, Examaple, cont.

```
</HEAD>
<BODY BGCOLOR="WHITE" onLoad="presetValues()">

<H1>Widgets "R" Us</H1>

<FORM ACTION="servlet/cwp.Widgets"
      NAME="widgetForm"
      onSubmit="storeCookies()">
First Name: <INPUT TYPE="TEXT" NAME="firstField">
<BR>
Last Name: <INPUT TYPE="TEXT" NAME="lastField">
<BR>
Account Number: <INPUT TYPE="TEXT" NAME="accountField">
<BR>
Widget Name: <INPUT TYPE="TEXT" NAME="widgetField">
<BR>
<INPUT TYPE="BUTTON" VALUE="Register Account"
       onClick="registerAccount()">
<INPUT TYPE="SUBMIT" VALUE="Submit Order">

</FORM>
</BODY>
</HTML>
```

Cookie, Example, Result



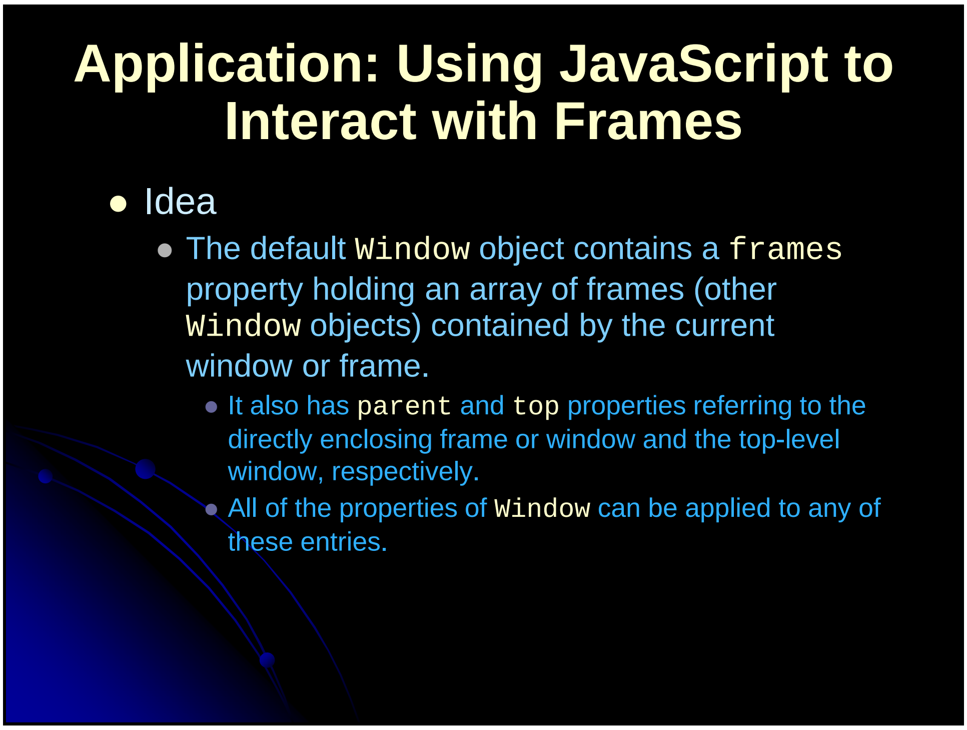
Widgets "R" Us - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Register Account
Submit Order
Done
My Computer
Widgets "R" Us - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Register Account
Submit Order
Done
My Computer

Widgets "R" Us

First Name: Loreen
Last Name: DeWitt
Account Number: 1100-BC
Widget Name: Acme-Ultra-Widget

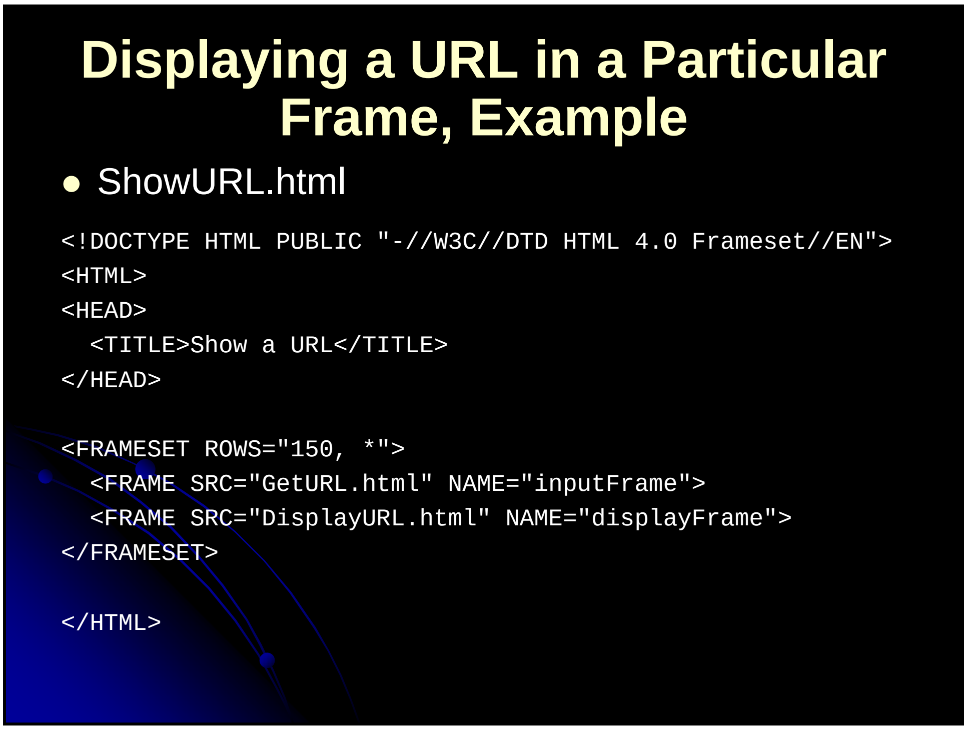
Widgets "R" Us

First Name: Loreen
Last Name: DeWitt
Account Number: 1100-BC
Widget Name:



Application: Using JavaScript to Interact with Frames

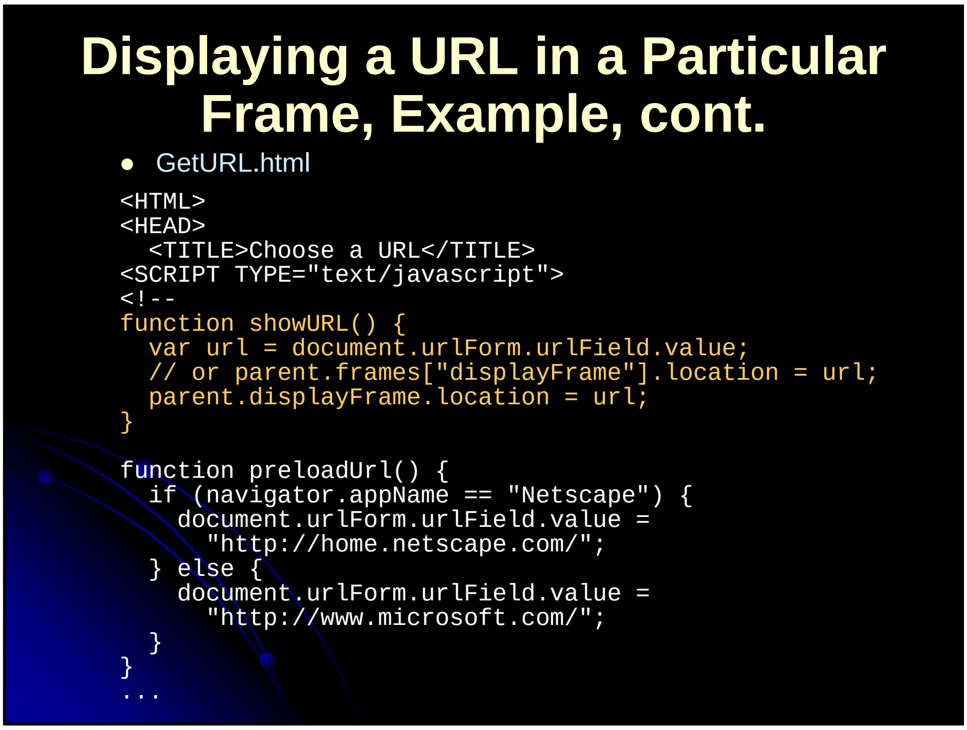
- Idea
  - The default `Window` object contains a `frames` property holding an array of frames (other `Window` objects) contained by the current window or frame.
    - It also has `parent` and `top` properties referring to the directly enclosing frame or window and the top-level window, respectively.
    - All of the properties of `Window` can be applied to any of these entries.



Displaying a URL in a Particular Frame, Example

- ShowURL.html

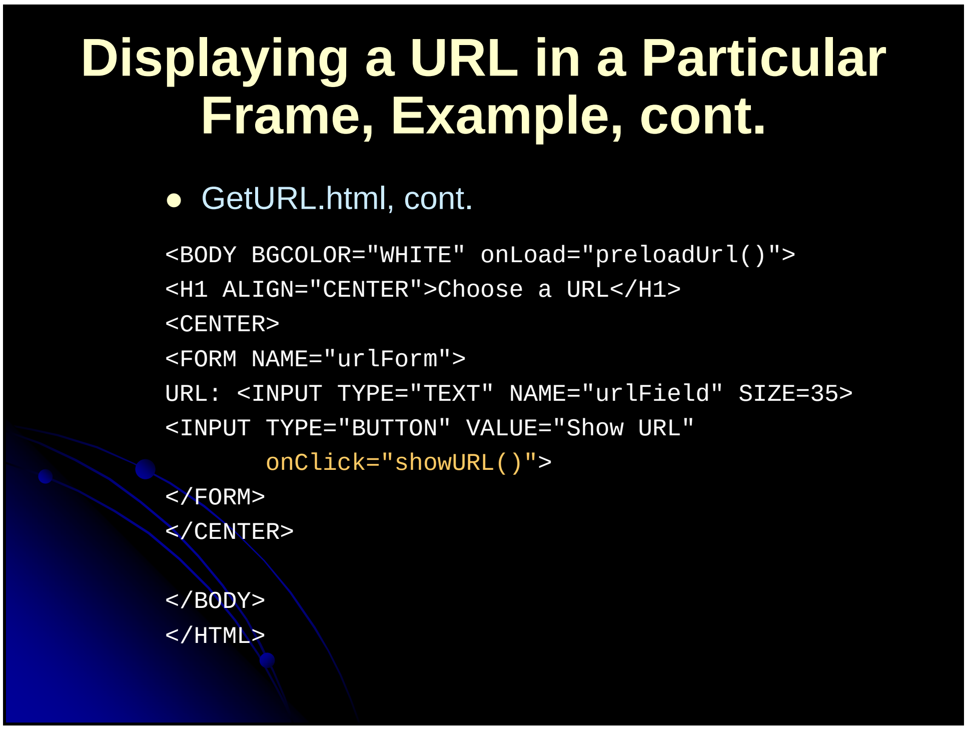
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN">
<HTML>
<HEAD>
  <TITLE>Show a URL</TITLE>
</HEAD>

<FRAMESET ROWS="150, *">
  <FRAME SRC="GetURL.html" NAME="inputFrame">
  <FRAME SRC="DisplayURL.html" NAME="displayFrame">
</FRAMESET>

</HTML>
```



Displaying a URL in a Particular Frame, Example, cont.

- GetURL.html

```
<HTML>
<HEAD>
  <TITLE>Choose a URL</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function showURL() {
  var url = document.urlForm.urlField.value;
  // or parent.frames["displayFrame"].location = url;
  parent.displayFrame.location = url;
}

function preloadUrl() {
  if (navigator.appName == "Netscape") {
    document.urlForm.urlField.value =
      "http://home.netscape.com/";
  } else {
    document.urlForm.urlField.value =
      "http://www.microsoft.com/";
  }
}
...
```

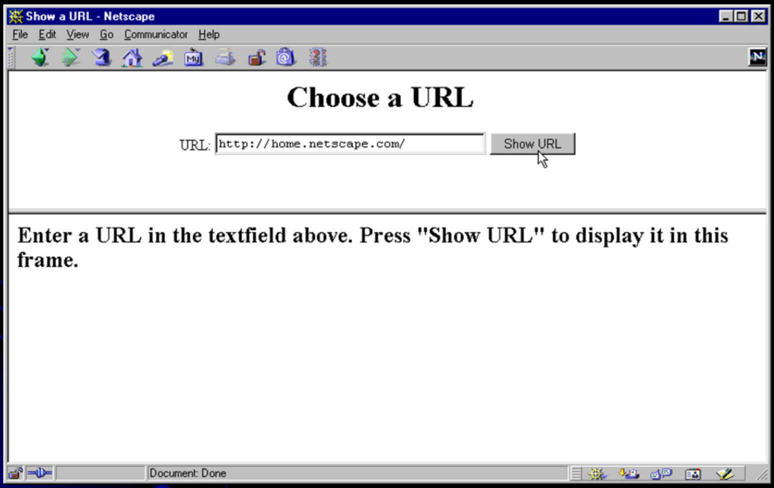


Displaying a URL in a Particular Frame, Example, cont.

- GetURL.html, cont.

```
<BODY BGCOLOR="WHITE" onLoad="preloadUrl()">
<H1 ALIGN="CENTER">Choose a URL</H1>
<CENTER>
<FORM NAME="urlForm">
URL: <INPUT TYPE="TEXT" NAME="urlField" SIZE=35>
<INPUT TYPE="BUTTON" VALUE="Show URL"
       onClick="showURL()">
</FORM>
</CENTER>

</BODY>
</HTML>
```

Displaying a URL in a Particular Frame, Result

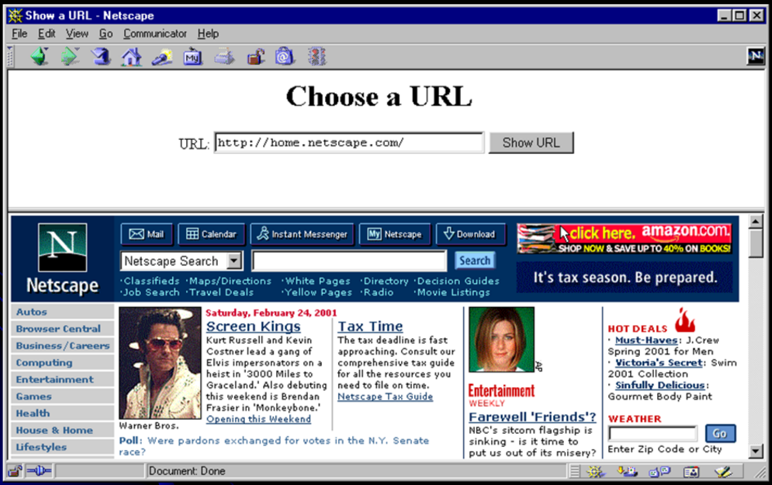


Choose a URL

URL: http://home.netscape.com/

Enter a URL in the textfield above. Press "Show URL" to display it in this frame.

Displaying a URL in a Particular Frame, Result, cont.



Choose a URL

URL: http://home.netscape.com/

Saturday, February 24, 2001

Screen Kings

Kurt Russell and Kevin Costner lead a gang of Elvis impersonators on a heist in '3000 Miles to Graceland.' Also debuting this weekend is Brendan Frasier in 'Monkeybone.' Opening this Weekend

Tax Time

The tax deadline is fast approaching. Consult our comprehensive tax guide for all the resources you need to file on time. Netscape Tax Guide

Farewell 'Friends'?

NBC's sitcom flagship is sinking - is it time to put us out of its misery?

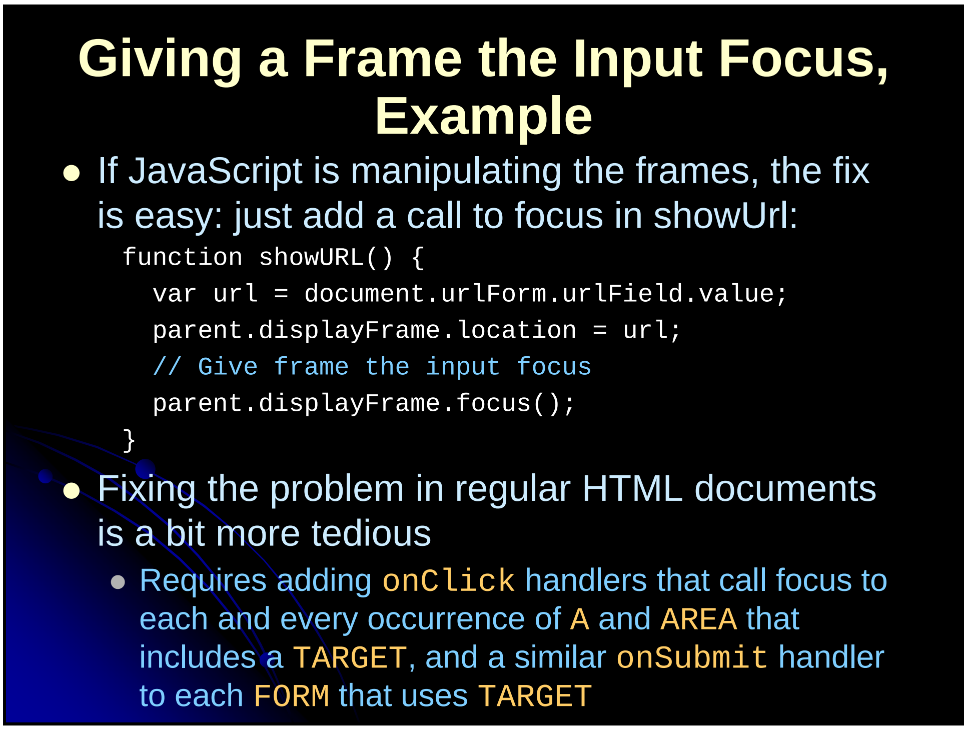
HOT DEALS

- Must-Haves: J.Crew Spring 2001 for Men
- Victoria's Secret: Swim 2001 Collection
- Sinfully Delicious: Gourmet Body Paint

WEATHER

Enter Zip Code or City

Poll: Were pardons exchanged for votes in the N.Y. Senate race?

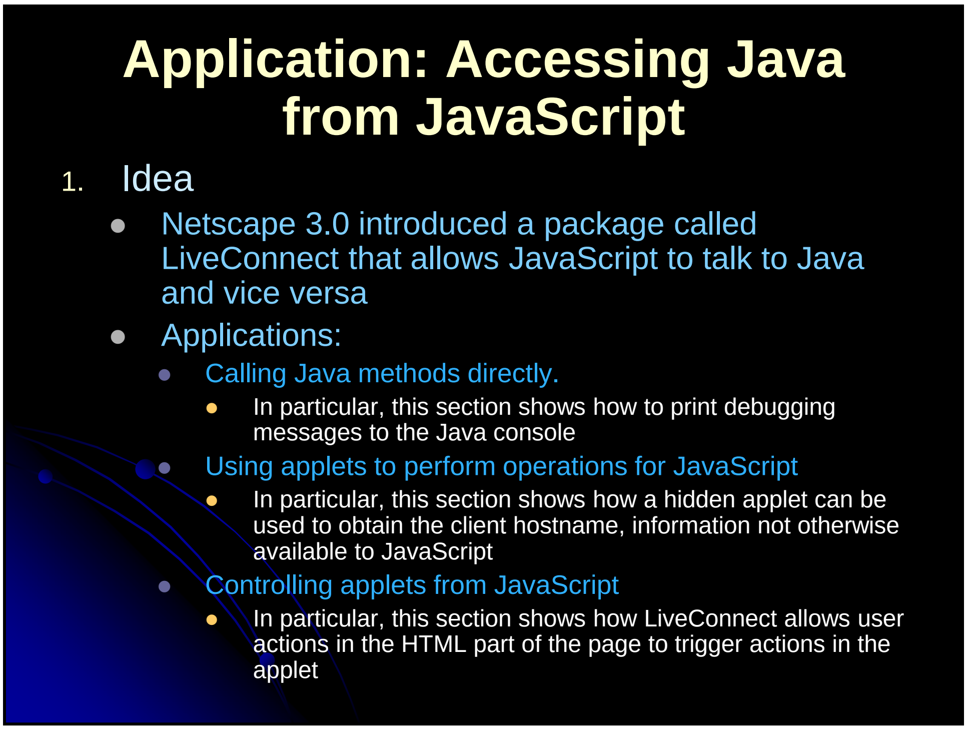


Giving a Frame the Input Focus, Example

- If JavaScript is manipulating the frames, the fix is easy: just add a call to focus in showUrl:

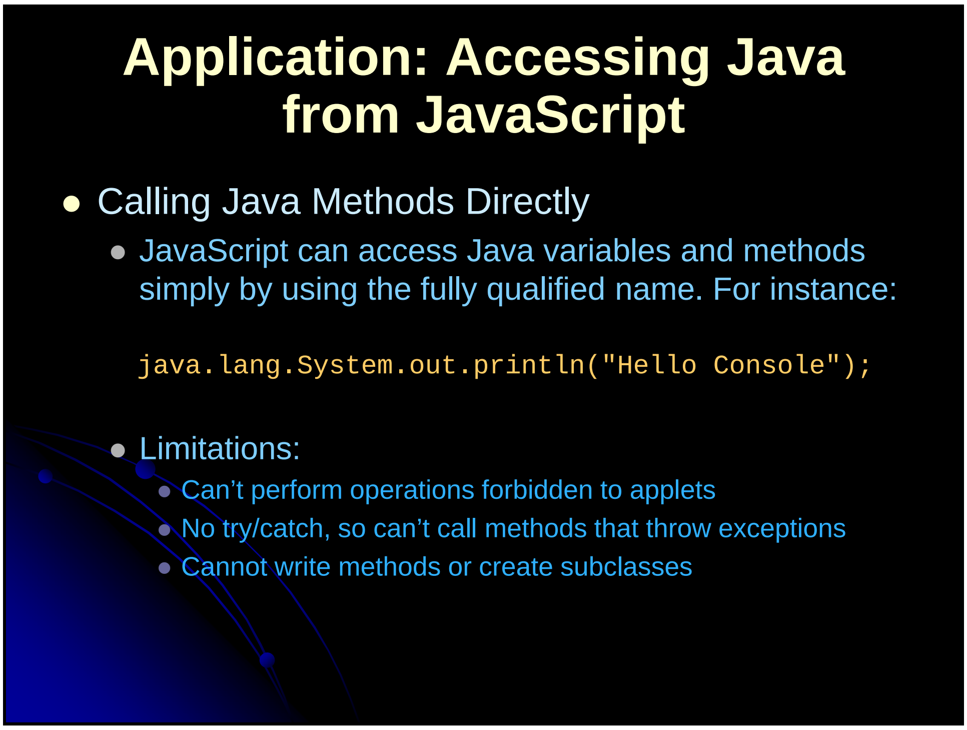
```
function showURL() {
  var url = document.urlForm.urlField.value;
  parent.displayFrame.location = url;
  // Give frame the input focus
  parent.displayFrame.focus();
}
```

- Fixing the problem in regular HTML documents is a bit more tedious
  - Requires adding `onClick` handlers that call focus to each and every occurrence of `A` and `AREA` that includes a `TARGET`, and a similar `onSubmit` handler to each `FORM` that uses `TARGET`



Application: Accessing Java from JavaScript

1. Idea
   - Netscape 3.0 introduced a package called LiveConnect that allows JavaScript to talk to Java and vice versa
   - Applications:
     - Calling Java methods directly.
       - In particular, this section shows how to print debugging messages to the Java console
     - Using applets to perform operations for JavaScript
       - In particular, this section shows how a hidden applet can be used to obtain the client hostname, information not otherwise available to JavaScript
     - Controlling applets from JavaScript
       - In particular, this section shows how LiveConnect allows user actions in the HTML part of the page to trigger actions in the applet

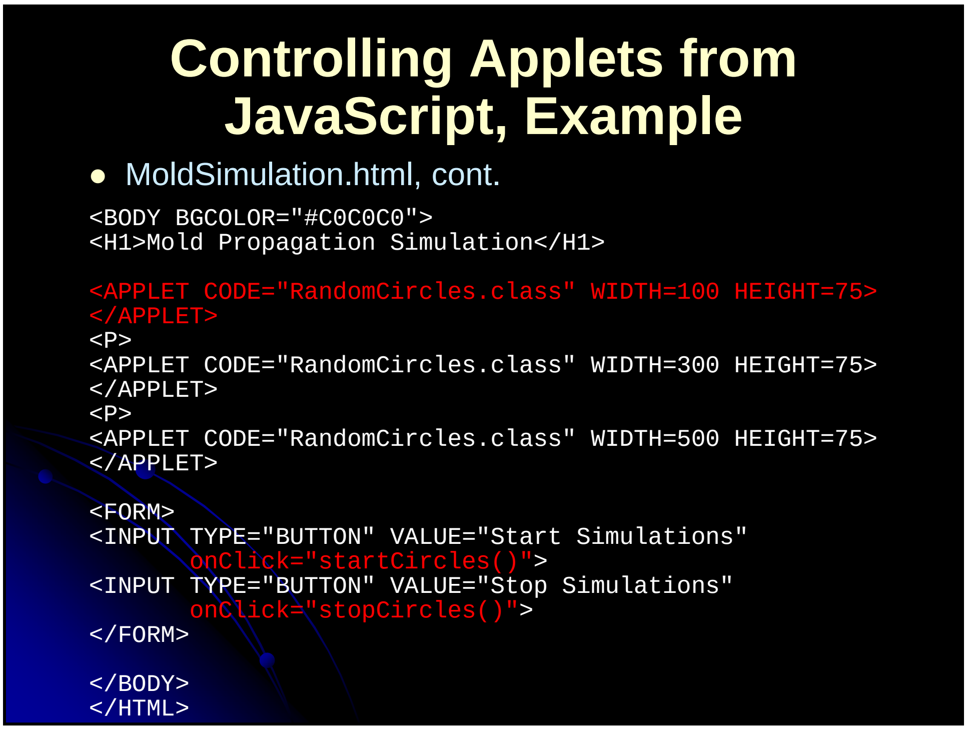


Application: Accessing Java from JavaScript

- Calling Java Methods Directly
  - JavaScript can access Java variables and methods simply by using the fully qualified name. For instance:

    ```
    java.lang.System.out.println("Hello Console");
    ```

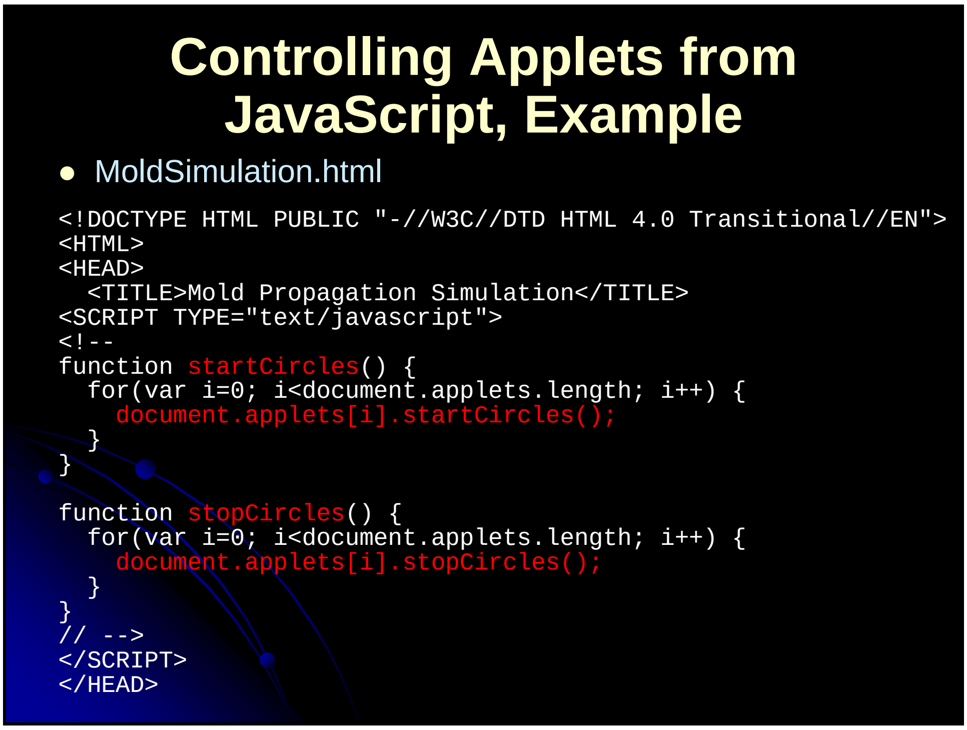
  - Limitations:
    - Can’t perform operations forbidden to applets
    - No try/catch, so can’t call methods that throw exceptions
    - Cannot write methods or create subclasses



Controlling Applets from JavaScript, Example

- MoldSimulation.html, cont.

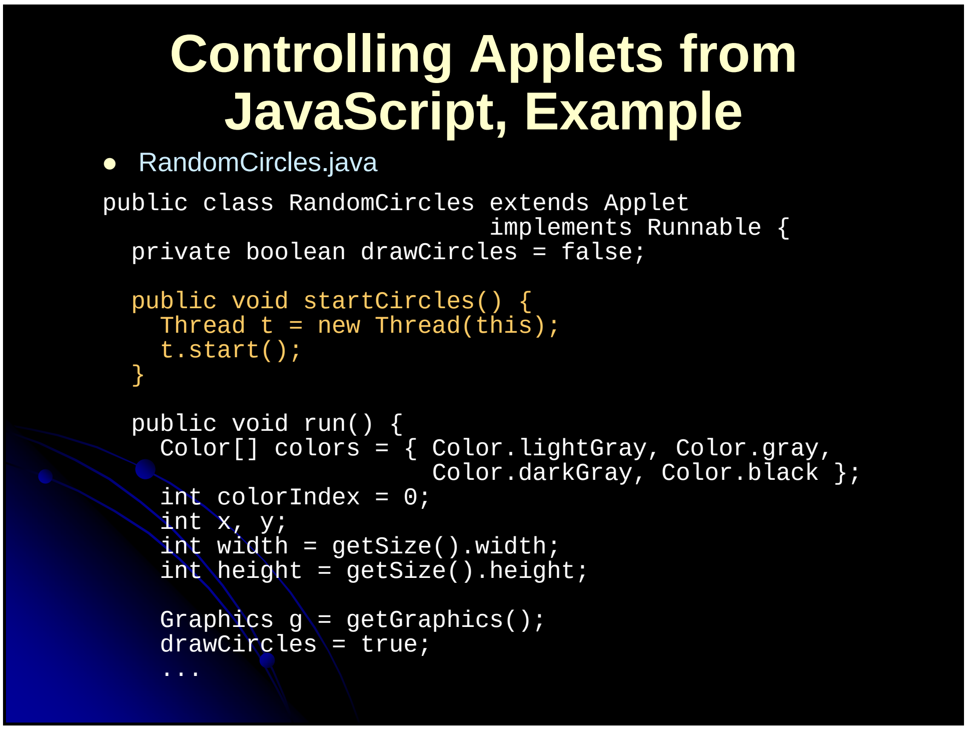
```
<BODY BGCOLOR="#C0C0C0">
<H1>Mold Propagation Simulation</H1>

<APPLET CODE="RandomCircles.class" WIDTH=100 HEIGHT=75>
</APPLET>
<P>
<APPLET CODE="RandomCircles.class" WIDTH=300 HEIGHT=75>
</APPLET>
<P>
<APPLET CODE="RandomCircles.class" WIDTH=500 HEIGHT=75>
</APPLET>

<FORM>
<INPUT TYPE="BUTTON" VALUE="Start Simulations"
       onClick="startCircles()">
<INPUT TYPE="BUTTON" VALUE="Stop Simulations"
       onClick="stopCircles()">
</FORM>

</BODY>
</HTML>
```



Controlling Applets from JavaScript, Example

- MoldSimulation.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Mold Propagation Simulation</TITLE>
<SCRIPT TYPE="text/javascript">
<!--
function startCircles() {
  for(var i=0; i<document.applets.length; i++) {
    document.applets[i].startCircles();
  }
}

function stopCircles() {
  for(var i=0; i<document.applets.length; i++) {
    document.applets[i].stopCircles();
  }
}
// -->
</SCRIPT>
</HEAD>
```



Controlling Applets from JavaScript, Example

- RandomCircles.java

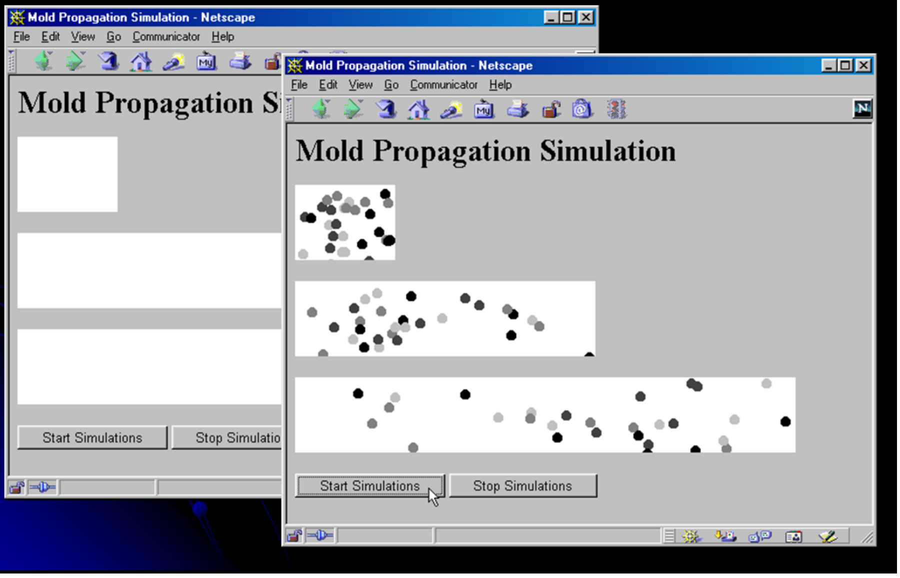
```
public class RandomCircles extends Applet
                           implements Runnable {
  private boolean drawCircles = false;

  public void startCircles() {
    Thread t = new Thread(this);
    t.start();
  }

  public void run() {
    Color[] colors = { Color.lightGray, Color.gray,
                       Color.darkGray, Color.black };
    int colorIndex = 0;
    int x, y;
    int width = getSize().width;
    int height = getSize().height;

    Graphics g = getGraphics();
    drawCircles = true;
    ...
```

Controlling Applets from JavaScript, Example

- RandomCircles.java, cont.

```
   while(drawCircles) {
      x = (int)Math.round(width * Math.random());
      y = (int)Math.round(height * Math.random());
      g.setColor(colors[colorIndex]);
      colorIndex = (colorIndex + 1) % colors.length;
      g.fillOval(x, y, 10, 10);
      pause(0.1);
    }
  }

  public void stopCircles() {
    drawCircles = false;
  }

  private void pause(double seconds) {
    try {
      Thread.sleep((int)(Math.round(seconds * 1000.0)));
    } catch(InterruptedException ie) {}
  }
}
```

Controlling Applets from JavaScript, Results


Mold Propagation Simulation - Netscape
File Edit View Go Communicator Help
Mold Propagation Simulation
Start Simulations
Stop Simulations

Accessing JavaScript from Java

- Steps
  1. Obtain and install the `JSObject` class
     - Installed with Netscape 4 (`javar40.jar`)
     - JDK 1.4 includes `JSObject` in `jaws.jar`
       - See Chapter 24 in http://java.sun.com/j2se/1.4.1/docs/guide/plugin/developer_guide/contents.html
  2. Import it in your applet

     ```
     import netscape.javascript.JSObject
     ```
  3. From the applet, obtain a JavaScript reference to the current window

     ```
     JSObject window = JSObject.getWindow(this);
     ```

Accessing JavaScript from Java, cont.

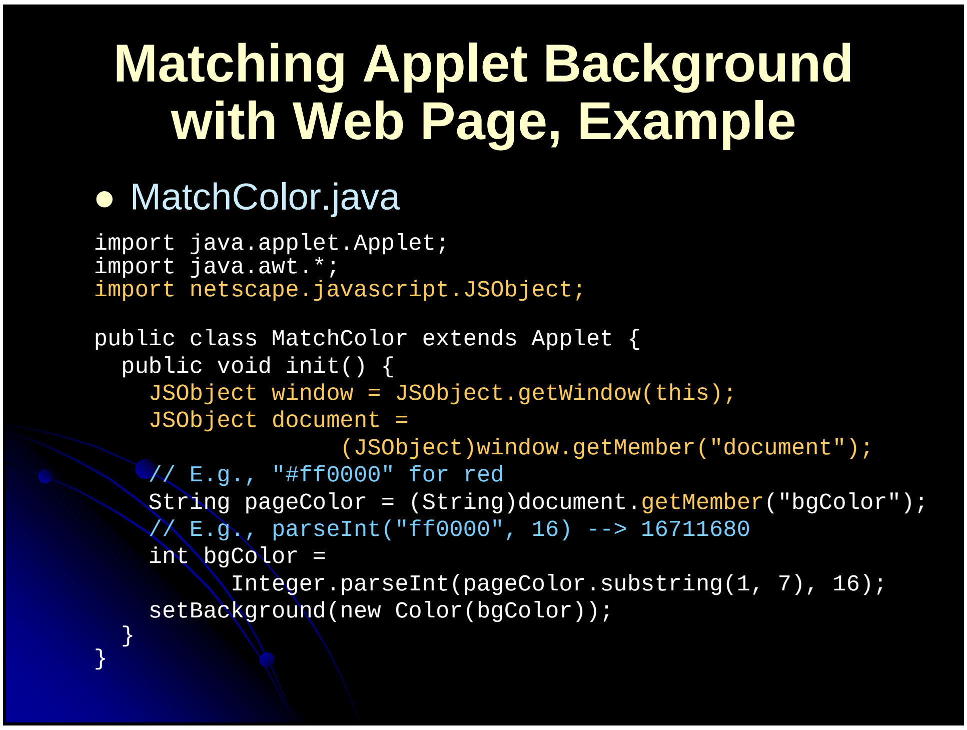
- Steps, cont.
  4. Read the JavaScript properties of interest
     - Use `getMember` to access properties of the `JSObject`

     ```
     JSObject someForm =
       (JSObject)document.getMember("someFormName");
     ```
  5. Set the JavaScript properties of interest
     - Use setMember to set properties of the `JSObject`

     ```
     document.setMember("bgColor", "red");
     ```
  6. Call the JavaScript methods of interest

     ```
     String[] message = { "An alert message" };
     window.call("alert", message);
     window.eval("alert('An alert message')");
     ```
  7. Give the applet permission to access its Web page

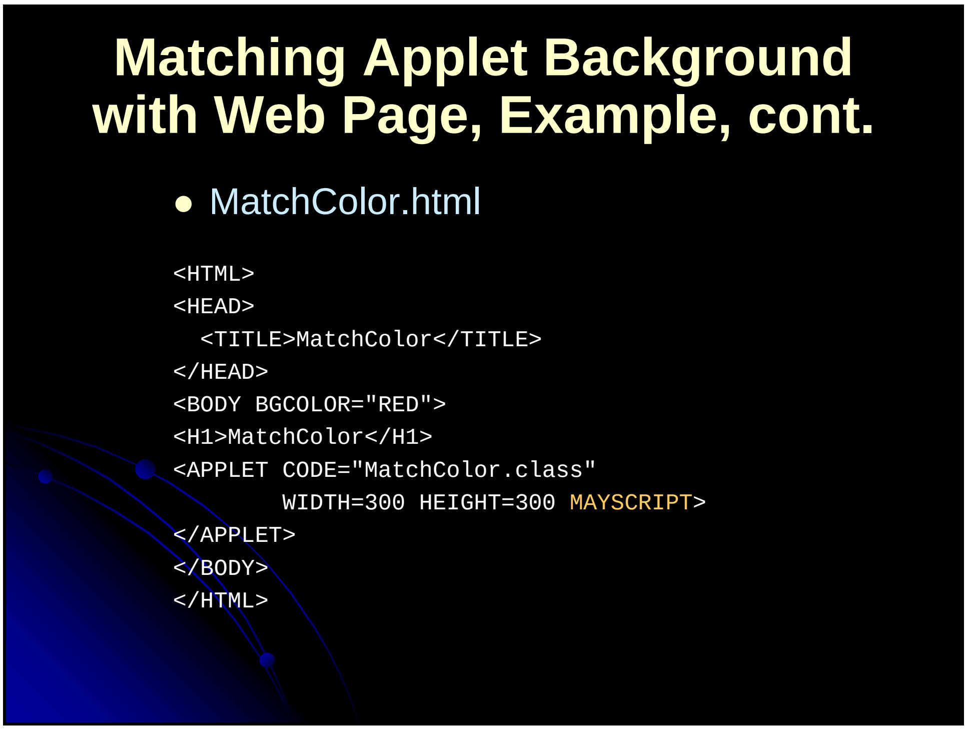
     ```
     <APPLET CODE=... WIDTH=... HEIGHT=... MAYSCRIPT>
     ...
     </APPLET>
     ```



Matching Applet Background with Web Page, Example

- MatchColor.java

```
import java.applet.Applet;
import java.awt.*;
import netscape.javascript.JSObject;

public class MatchColor extends Applet {
  public void init() {
    JSObject window = JSObject.getWindow(this);
    JSObject document =
              (JSObject)window.getMember("document");
    // E.g., "#ff0000" for red
    String pageColor = (String)document.getMember("bgColor");
    // E.g., parseInt("ff0000", 16) --> 16711680
    int bgColor =
          Integer.parseInt(pageColor.substring(1, 7), 16);
    setBackground(new Color(bgColor));
  }
}
```



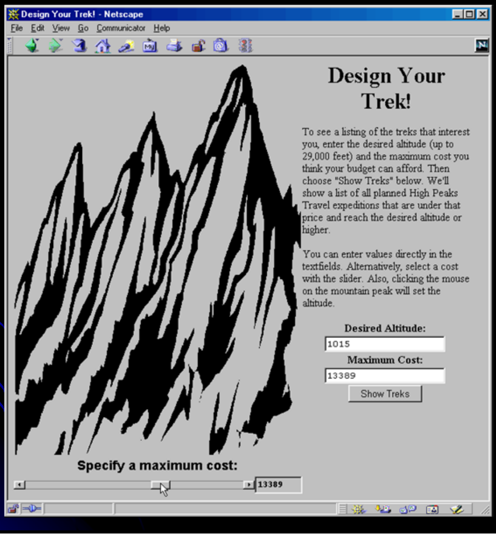
Matching Applet Background with Web Page, Example, cont.

- MatchColor.html

```
<HTML>
<HEAD>
  <TITLE>MatchColor</TITLE>
</HEAD>
<BODY BGCOLOR="RED">
<H1>MatchColor</H1>
<APPLET CODE="MatchColor.class"
        WIDTH=300 HEIGHT=300 MAYSCRIPT>
</APPLET>
</BODY>
</HTML>
```

Applet That Controls HTML Form Values, Example

- See on-line example for Everest.html


Design Your Trek! - Netscape
File Edit View Go Communicator Help

Design Your Trek!

To see a listing of the treks that interest you, enter the desired altitude (up to 29,000 feet) and the maximum cost you think your budget can afford. Then choose "Show Treks" below. We'll show a list of all planned High Peaks Travel expeditions that are under that price and reach the desired altitude or higher.

You can enter values directly in the textfields. Alternatively, select a cost with the slider. Also, clicking the mouse on the mountain peak will set the altitude.

Desired Altitude: 1015

Maximum Cost: 13389

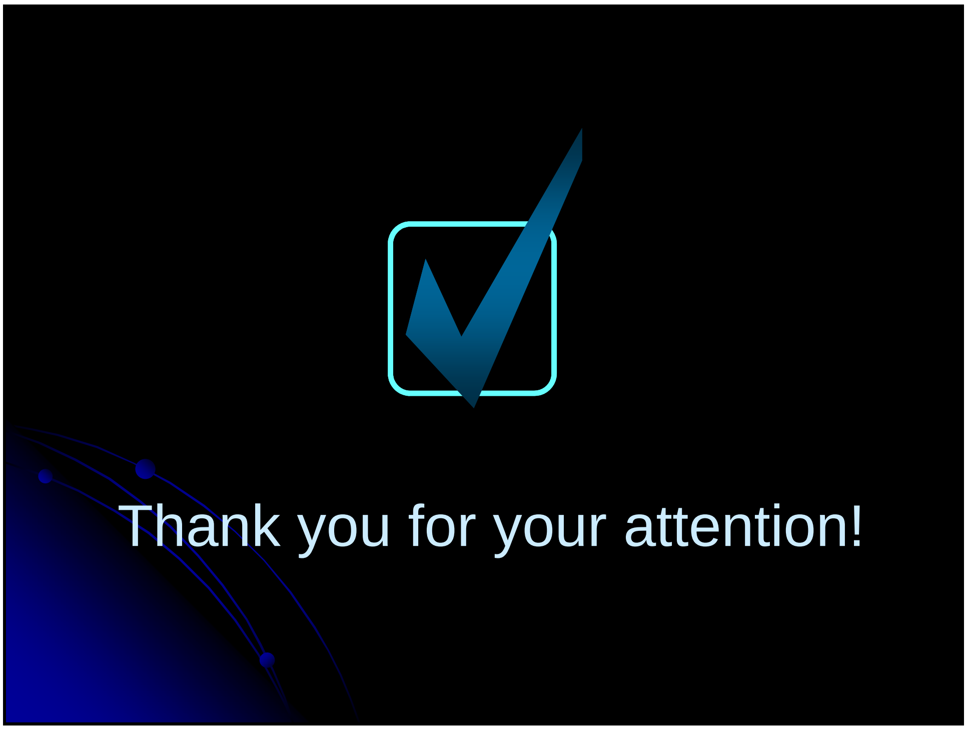
Show Treks

Specify a maximum cost: 13389

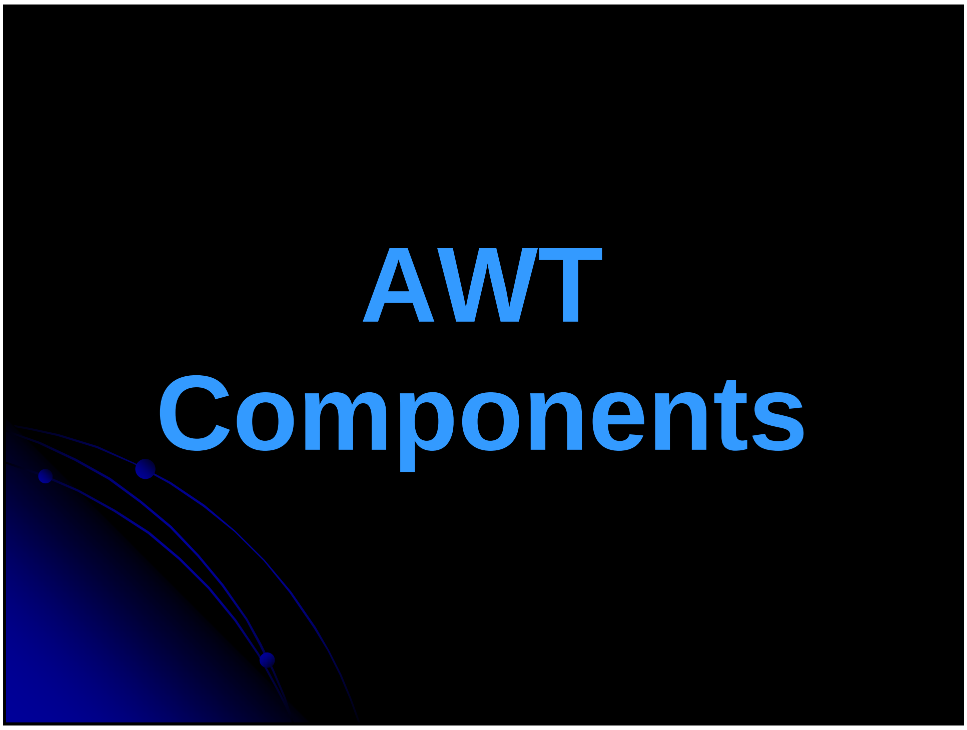
Summary

- JavaScript permits you to:
  - Customize Web pages based on the situation
  - Make pages more dynamic
  - Validate HTML form input
  - Manipulate cookies
  - Control frames
  - Integrate Java and JavaScript
- Web resources:

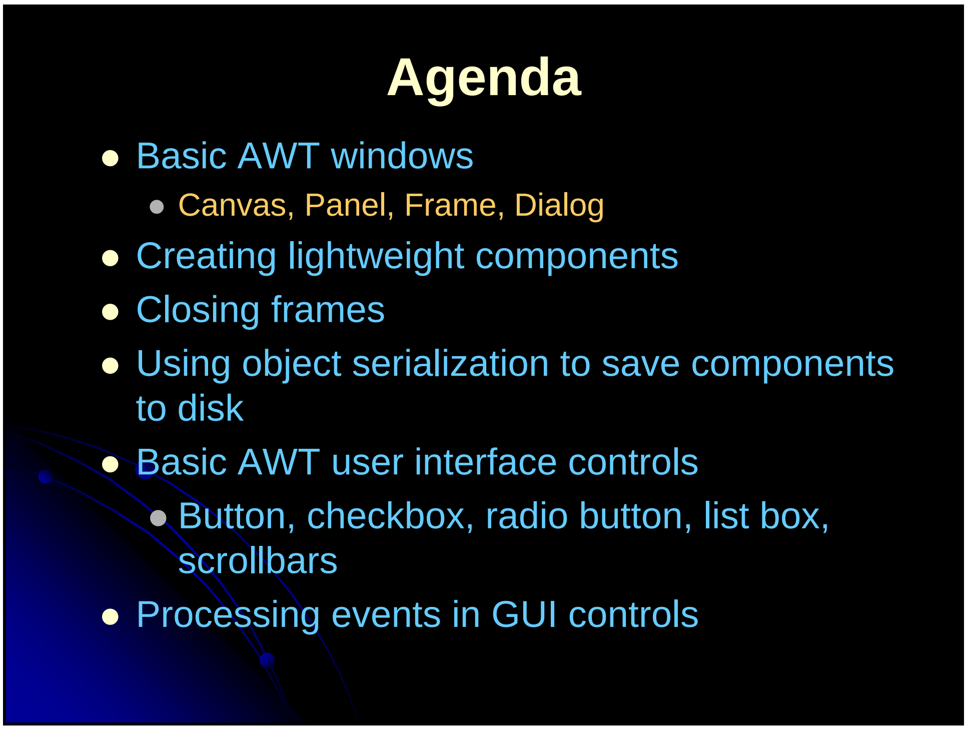
  http://www.javascriptsource.com



Thank you for your attention!

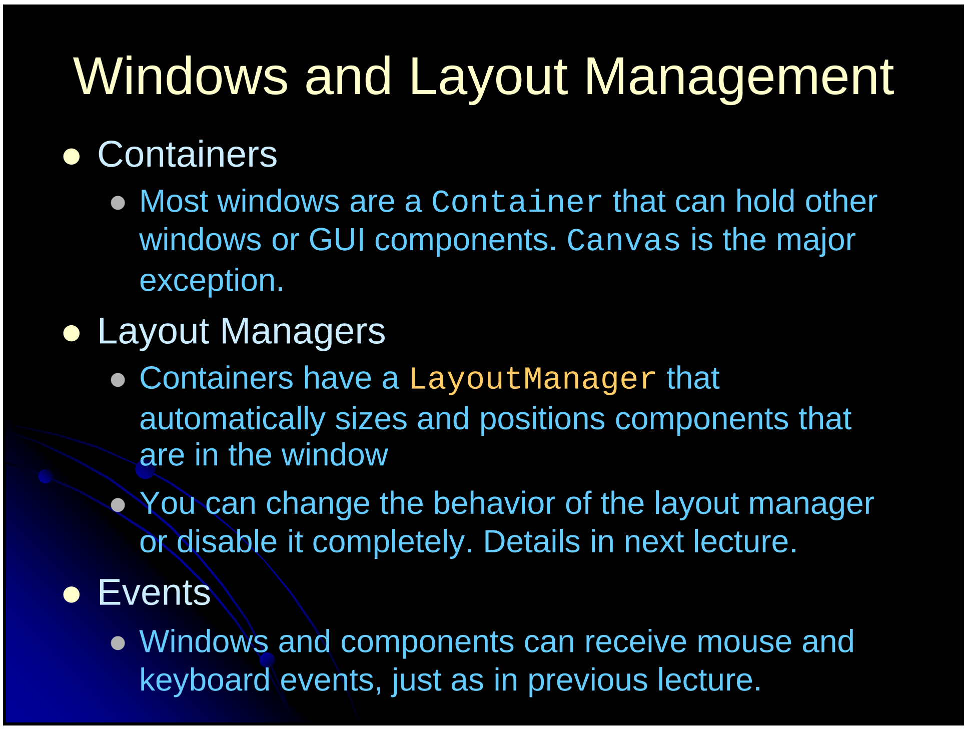


AWT Components



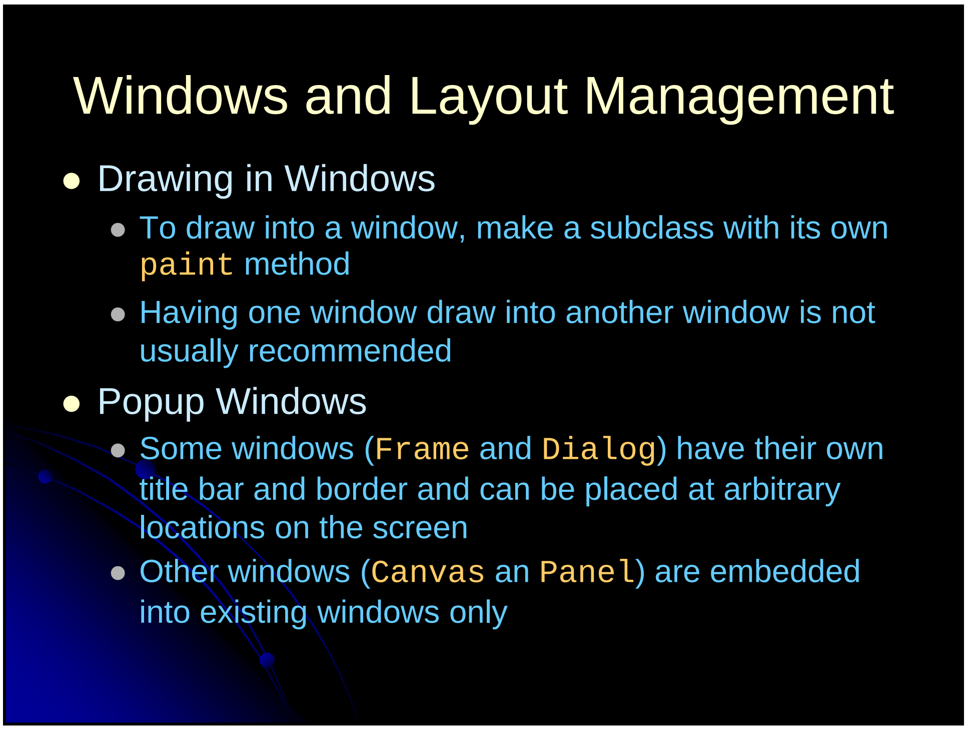
Agenda

- Basic AWT windows
  - Canvas, Panel, Frame, Dialog
- Creating lightweight components
- Closing frames
- Using object serialization to save components to disk
- Basic AWT user interface controls
  - Button, checkbox, radio button, list box, scrollbars
- Processing events in GUI controls



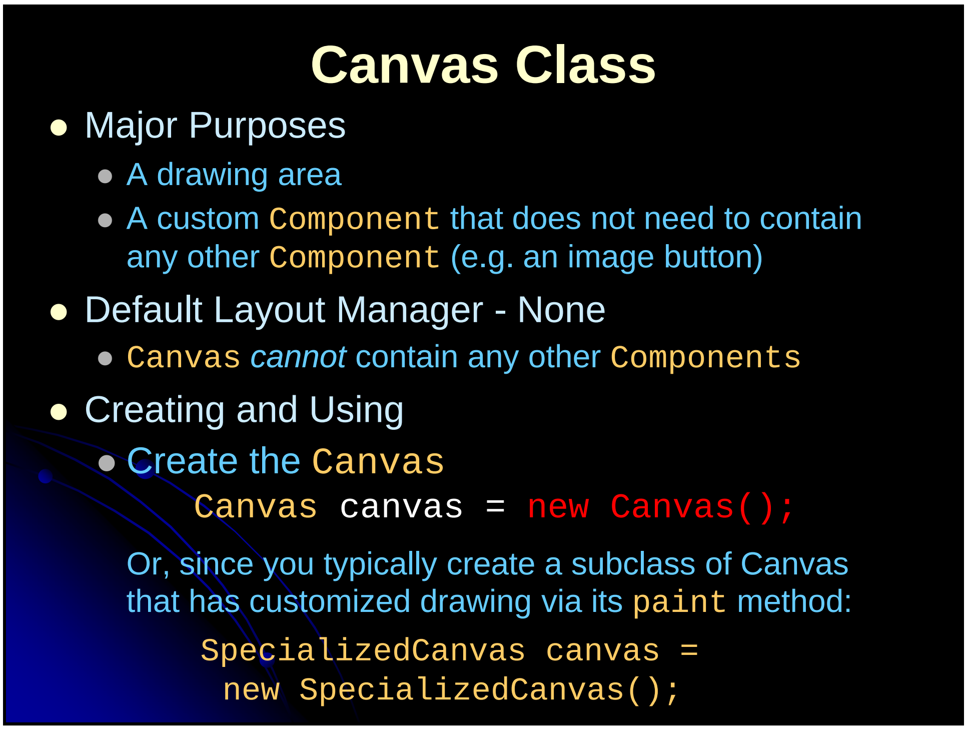
Windows and Layout Management

- Containers
  - Most windows are a `Container` that can hold other windows or GUI components. `Canvas` is the major exception.
- Layout Managers
  - Containers have a `LayoutManager` that automatically sizes and positions components that are in the window
  - You can change the behavior of the layout manager or disable it completely. Details in next lecture.
- Events
  - Windows and components can receive mouse and keyboard events, just as in previous lecture.



Windows and Layout Management

- Drawing in Windows
  - To draw into a window, make a subclass with its own `paint` method
  - Having one window draw into another window is not usually recommended
- Popup Windows
  - Some windows (`Frame` and `Dialog`) have their own title bar and border and can be placed at arbitrary locations on the screen
  - Other windows (`Canvas` an `Panel`) are embedded into existing windows only



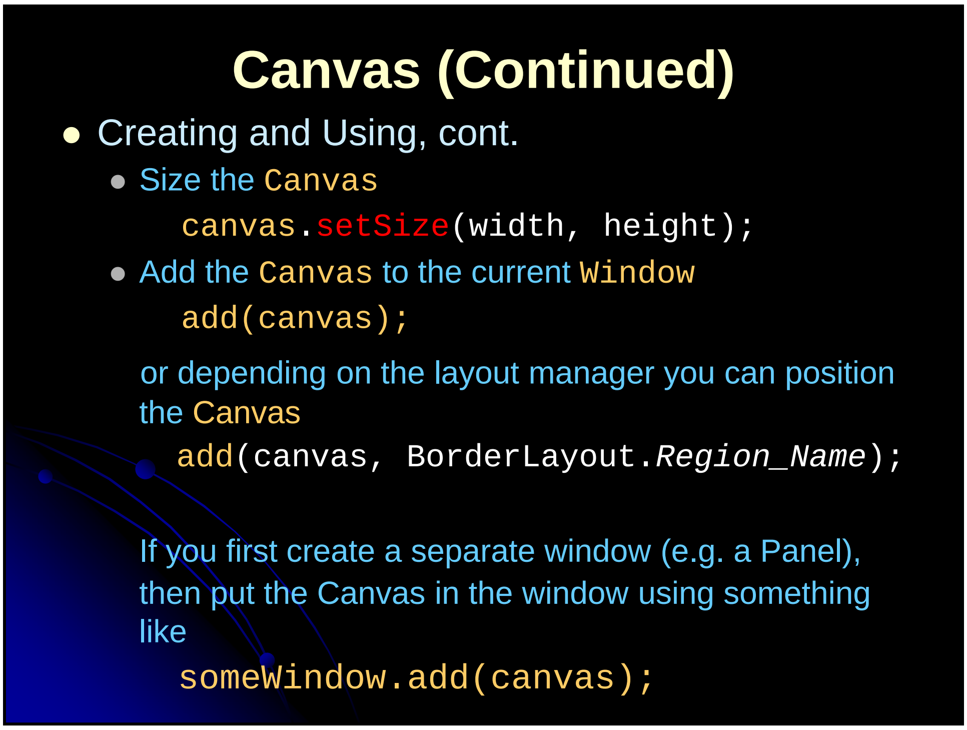
Canvas Class

- Major Purposes
  - A drawing area
  - A custom Component that does not need to contain any other Component (e.g. an image button)
- Default Layout Manager - None
  - Canvas *cannot* contain any other Components
- Creating and Using
  - Create the Canvas

```
    Canvas canvas = new Canvas();
```

Or, since you typically create a subclass of Canvas that has customized drawing via its paint method:

```
     SpecializedCanvas canvas =
       new SpecializedCanvas();
```



Canvas (Continued)

- Creating and Using, cont.
  - Size the Canvas

```
   canvas.setSize(width, height);
```

  - Add the Canvas to the current Window

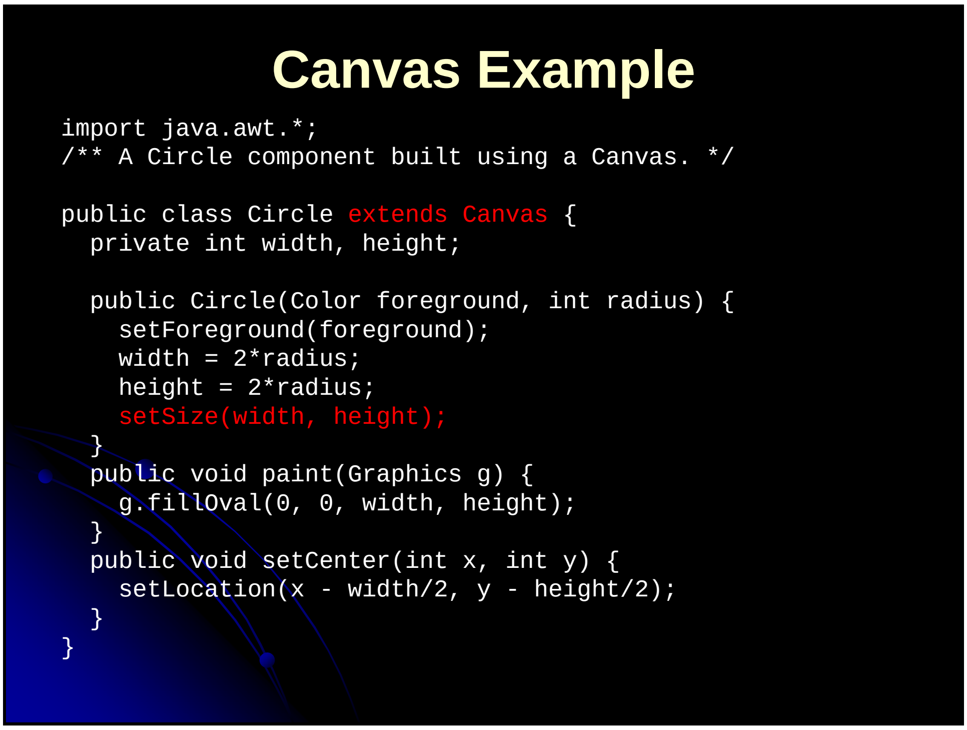
```
   add(canvas);
```

or depending on the layout manager you can position the Canvas

```
   add(canvas, BorderLayout.Region_Name);
```

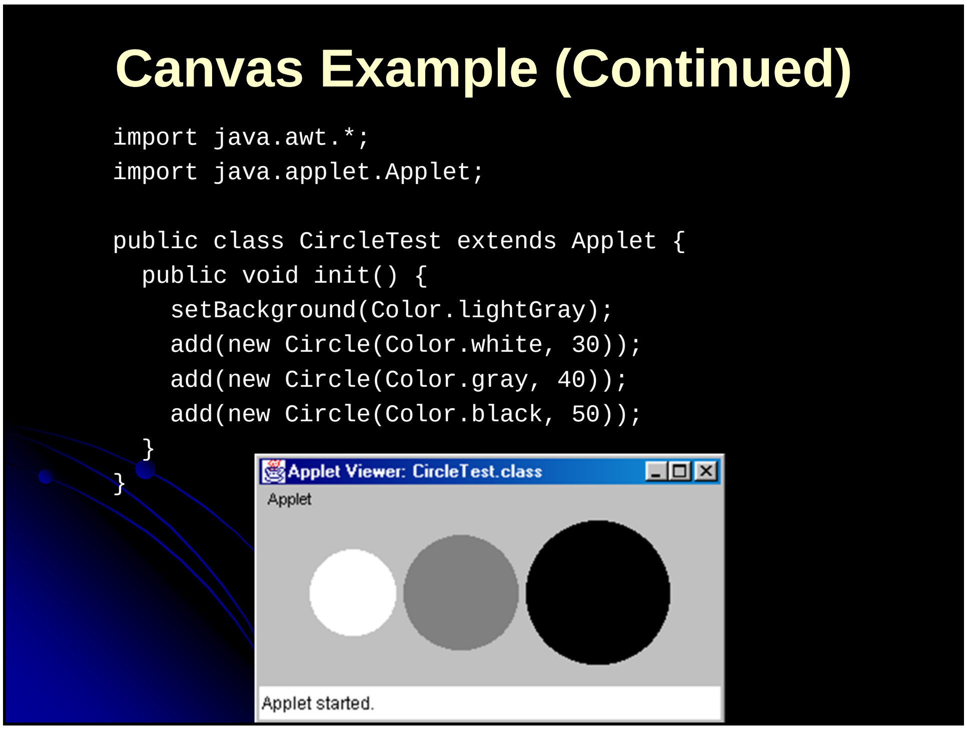
If you first create a separate window (e.g. a Panel), then put the Canvas in the window using something like

```
   someWindow.add(canvas);
```



Canvas Example

```
import java.awt.*;
/** A Circle component built using a Canvas. */

public class Circle extends Canvas {
  private int width, height;

  public Circle(Color foreground, int radius) {
    setForeground(foreground);
    width = 2*radius;
    height = 2*radius;
    setSize(width, height);
  }
  public void paint(Graphics g) {
    g.fillOval(0, 0, width, height);
  }
  public void setCenter(int x, int y) {
    setLocation(x - width/2, y - height/2);
  }
}
```



Canvas Example (Continued)

```
import java.awt.*;
import java.applet.Applet;

public class CircleTest extends Applet {
  public void init() {
    setBackground(Color.lightGray);
    add(new Circle(Color.white, 30));
    add(new Circle(Color.gray, 40));
    add(new Circle(Color.black, 50));
  }
}
```

Applet Viewer: CircleTest.class

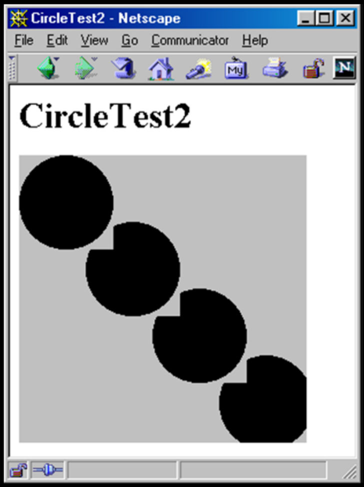
Applet

Applet started.

Canvases are Rectangular and Opaque: Example

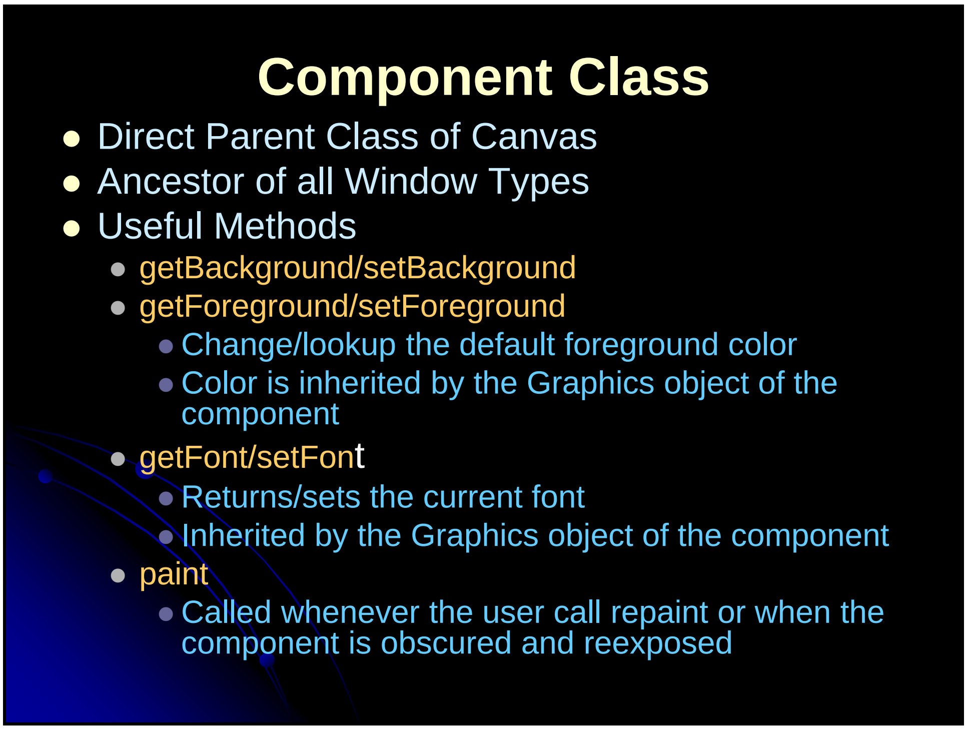
```
public class CircleTest2 extends Applet {
  public void init() {
    setBackground(Color.lightGray);
    setLayout(null); // Turn off layout manager.
    Circle circle;
    int radius = getSize().width/6;
    int deltaX = round(2.0*(double)radius/Math.sqrt(2.0));
    for (int x=radius; x<6*radius; x=x+deltaX) {
      circle = new Circle(Color.black, radius);
      add(circle);      circle.setCenter(x, x);
    }
  }
  private int round(double num) {
    return((int)Math.round(num));
  }
}
```

Canvases are Rectangular and Opaque: Result



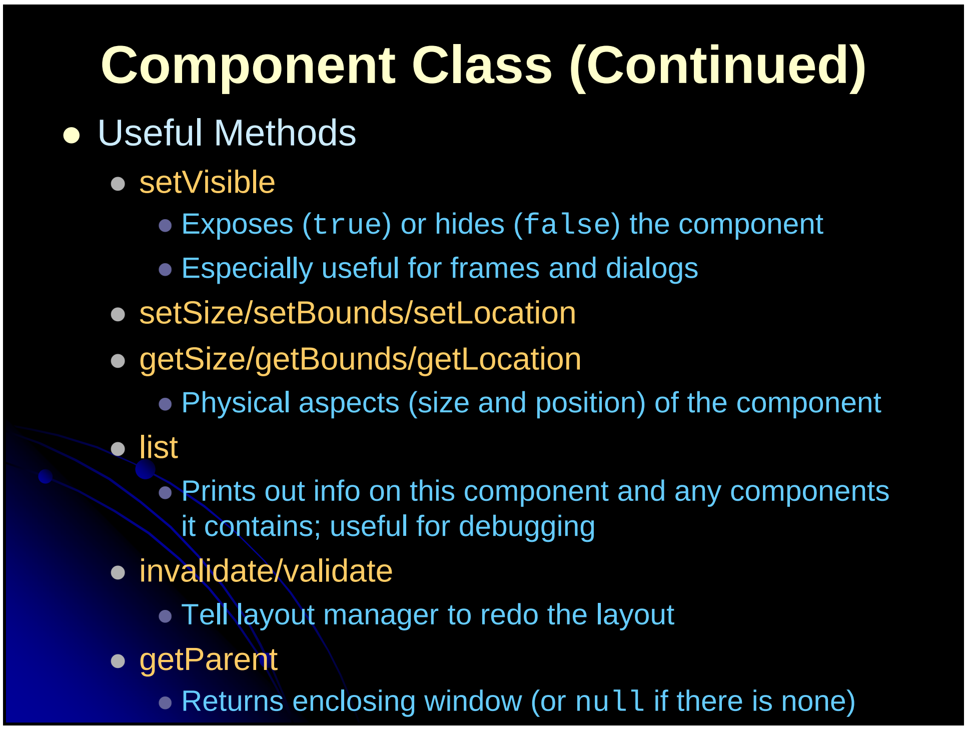
CircleTest2

Standard components have an associated peer (native window system object).



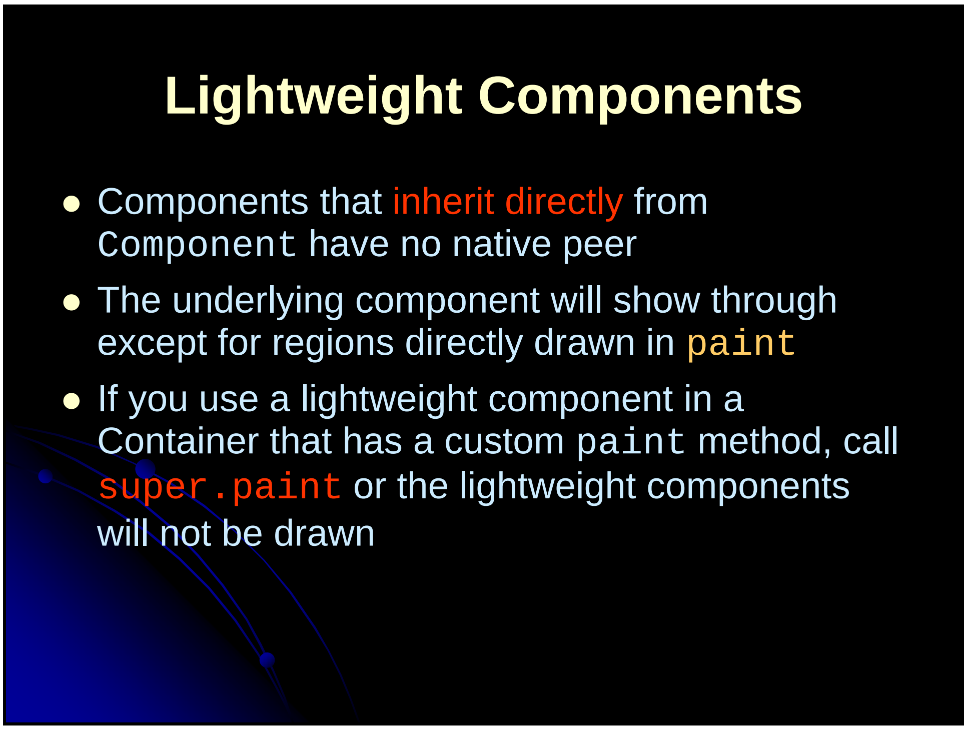
Component Class

- Direct Parent Class of Canvas
- Ancestor of all Window Types
- Useful Methods
  - getBackground/setBackground
  - getForeground/setForeground
    - Change/lookup the default foreground color
    - Color is inherited by the Graphics object of the component
  - getFont/setFont
    - Returns/sets the current font
    - Inherited by the Graphics object of the component
  - paint
    - Called whenever the user call repaint or when the component is obscured and reexposed



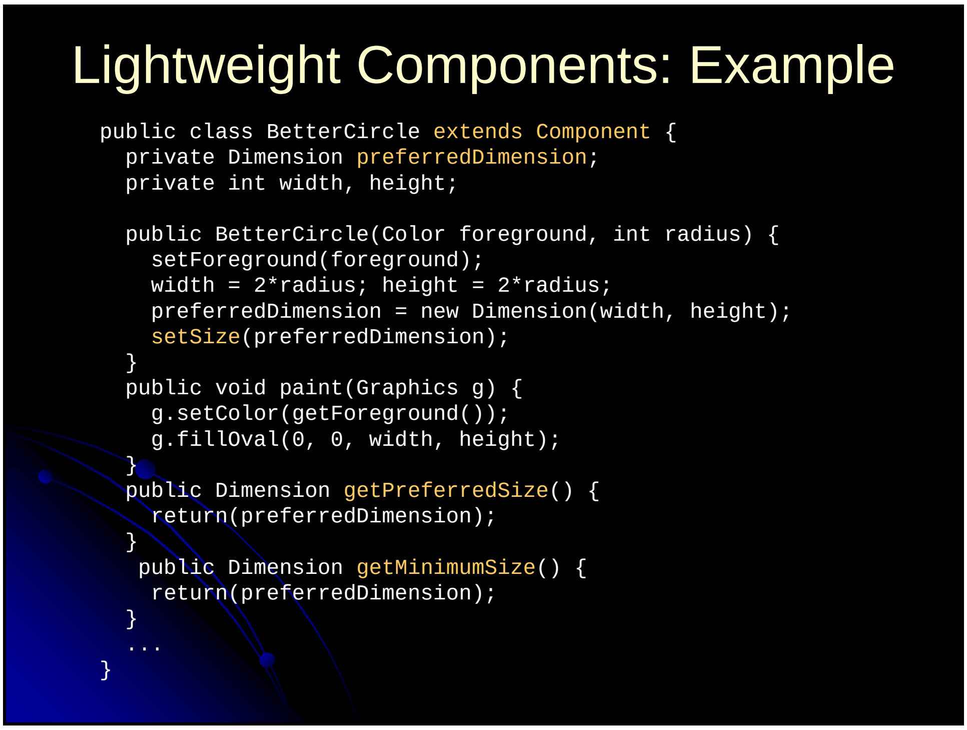
Component Class (Continued)

- Useful Methods
  - setVisible
    - Exposes (`true`) or hides (`false`) the component
    - Especially useful for frames and dialogs
  - setSize/setBounds/setLocation
  - getSize/getBounds/getLocation
    - Physical aspects (size and position) of the component
  - list
    - Prints out info on this component and any components it contains; useful for debugging
  - invalidate/validate
    - Tell layout manager to redo the layout
  - getParent
    - Returns enclosing window (or `null` if there is none)



Lightweight Components

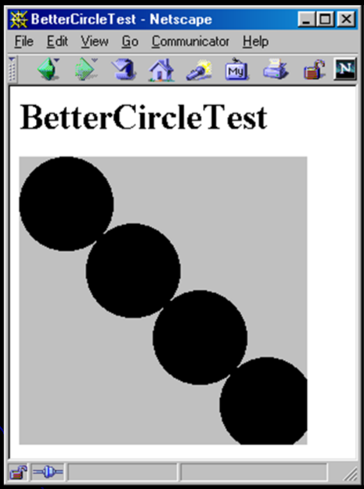
- Components that inherit directly from `Component` have no native peer
- The underlying component will show through except for regions directly drawn in `paint`
- If you use a lightweight component in a Container that has a custom `paint` method, call `super.paint` or the lightweight components will not be drawn



Lightweight Components: Example

```
public class BetterCircle extends Component {
  private Dimension preferredDimension;
  private int width, height;

  public BetterCircle(Color foreground, int radius) {
    setForeground(foreground);
    width = 2*radius; height = 2*radius;
    preferredDimension = new Dimension(width, height);
    setSize(preferredDimension);
  }
  public void paint(Graphics g) {
    g.setColor(getForeground());
    g.fillOval(0, 0, width, height);
  }
  public Dimension getPreferredSize() {
    return(preferredDimension);
  }
   public Dimension getMinimumSize() {
    return(preferredDimension);
  }
  ...
}
```

Lightweight Components: Result


BetterCircleTest - Netscape
File Edit View Go Communicator Help

BetterCircleTest

Lightweight components can be transparent

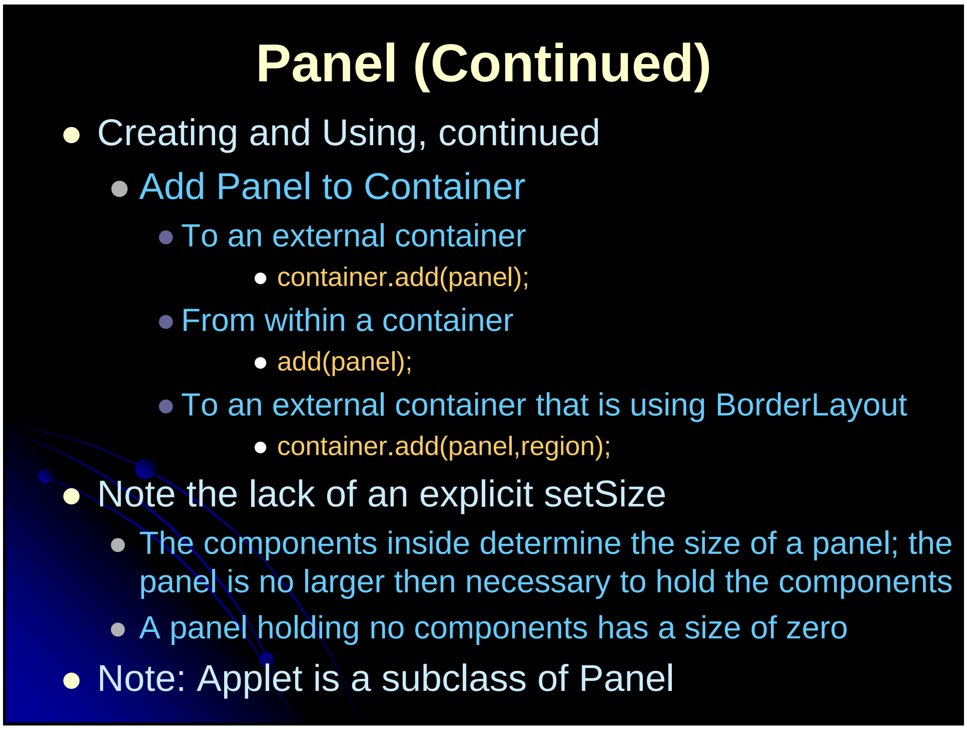
Panel Class

- Major Purposes
  - To group/organize components
  - A custom component that requires embedded components
- Default Layout Manager - FlowLayout
  - Shrinks components to their preferred (minimum) size
  - Places them left to right in centered rows
- Creating and Using
  - Create the Panel

```
Panel panel = new Panel();
```

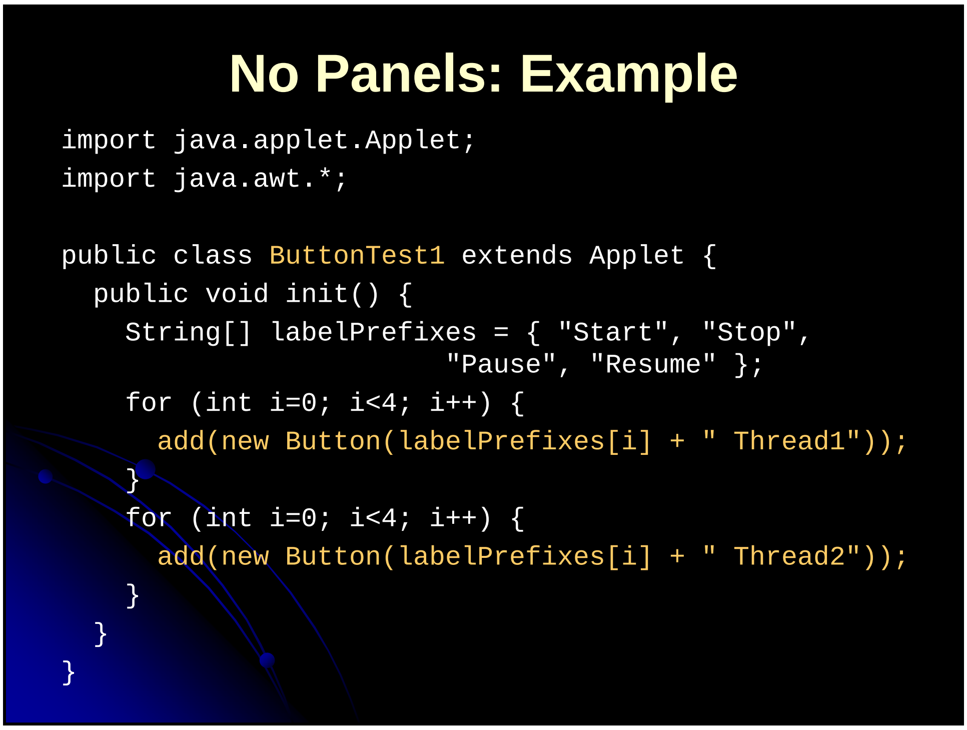
  - Add Components to Panel

```
panel.add(someComponent);
panel.add(someOtherComponent);
...
```



Panel (Continued)

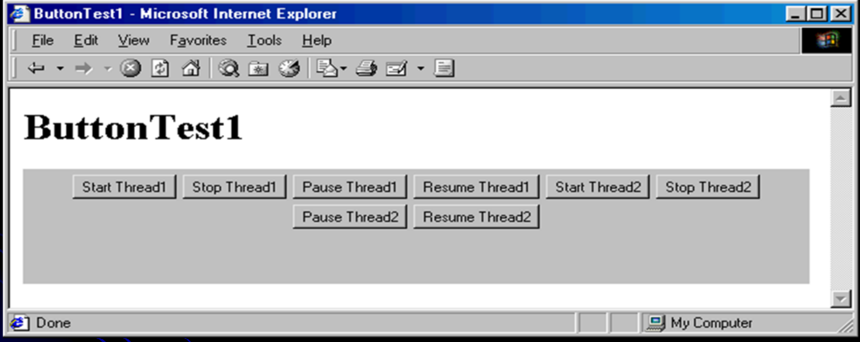
- Creating and Using, continued
  - Add Panel to Container
    - To an external container
      - container.add(panel);
    - From within a container
      - add(panel);
    - To an external container that is using BorderLayout
      - container.add(panel,region);
- Note the lack of an explicit setSize
  - The components inside determine the size of a panel; the panel is no larger then necessary to hold the components
  - A panel holding no components has a size of zero
- Note: Applet is a subclass of Panel



No Panels: Example

```
import java.applet.Applet;
import java.awt.*;

public class ButtonTest1 extends Applet {
  public void init() {
    String[] labelPrefixes = { "Start", "Stop",
                               "Pause", "Resume" };
    for (int i=0; i<4; i++) {
      add(new Button(labelPrefixes[i] + " Thread1"));
    }
    for (int i=0; i<4; i++) {
      add(new Button(labelPrefixes[i] + " Thread2"));
    }
  }
}
```

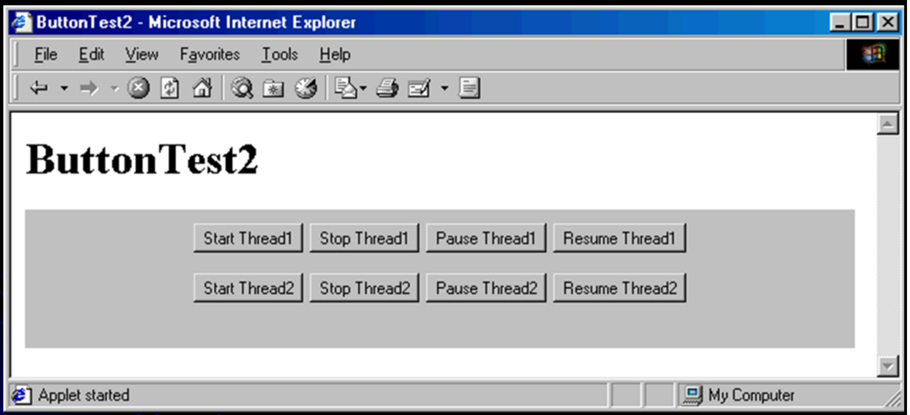
No Panels: Result



Panels: Example

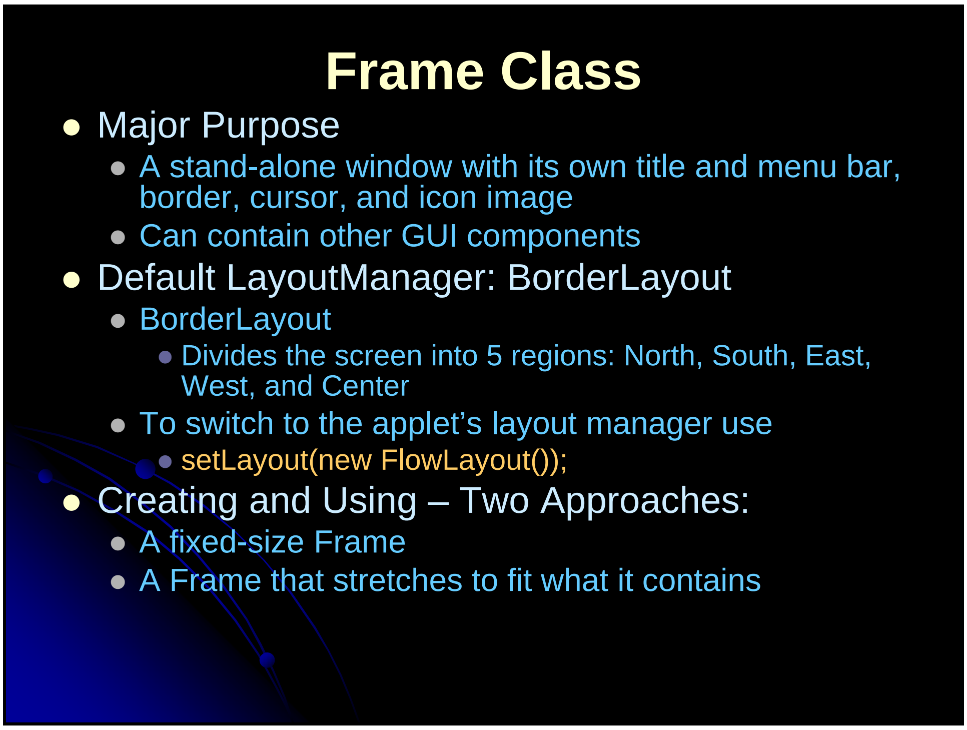
```
import java.applet.Applet;
import java.awt.*;

public class ButtonTest2 extends Applet {
  public void init() {
    String[] labelPrefixes = { "Start", "Stop",
                   "Pause", "Resume" };
    Panel p1 = new Panel();
    for (int i=0; i<4; i++) {
      p1.add(new Button(labelPrefixes[i] + " Thread1"));
    }
    Panel p2 = new Panel();
    for (int i=0; i<4; i++) {
      p2.add(new Button(labelPrefixes[i] + " Thread2"));
    }
    add(p1);
    add(p2);
  }
}
```

Panels: Result



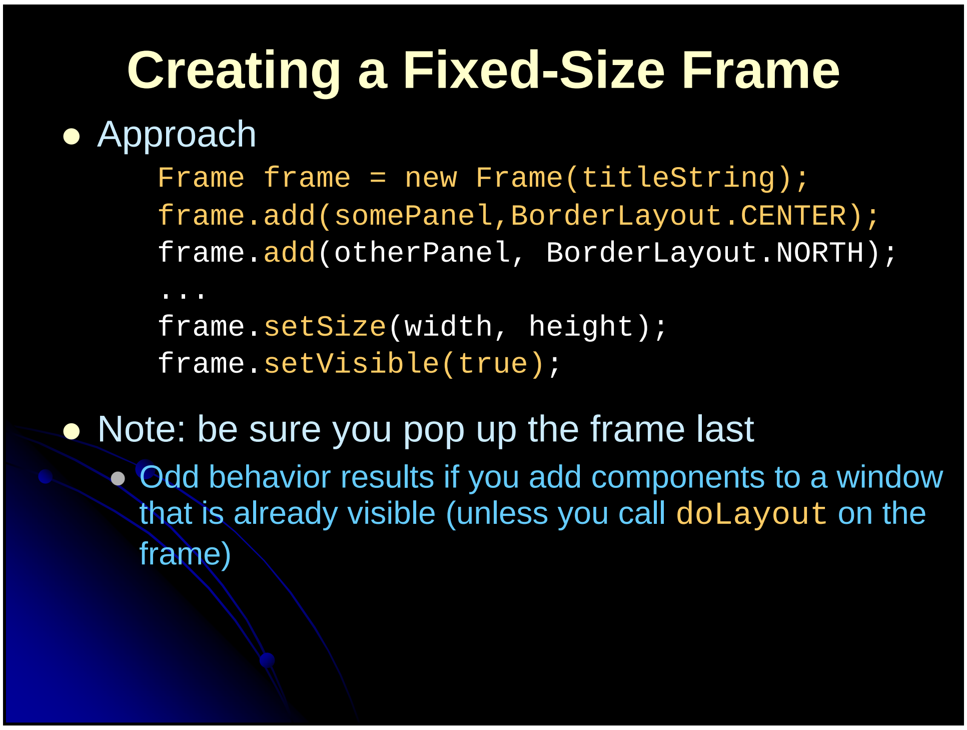
Container Class

- Ancestor of all Window Types Except `Canvas`
- Inherits all `Component` Methods
- Useful Container Methods
  - add
    - Add a component to the container (in the component array)
    - If using BorderLayout, you can also specify in which region to place the component
  - remove
    - Remove the component from the window (container)
  - getComponents
    - Returns an array of components in the window
    - Used by layout managers
  - setLayout
    - Changes the layout manager associated with the window



Frame Class

- Major Purpose
  - A stand-alone window with its own title and menu bar, border, cursor, and icon image
  - Can contain other GUI components
- Default LayoutManager: BorderLayout
  - BorderLayout
    - Divides the screen into 5 regions: North, South, East, West, and Center
  - To switch to the applet's layout manager use
    - setLayout(new FlowLayout());
- Creating and Using – Two Approaches:
  - A fixed-size Frame
  - A Frame that stretches to fit what it contains

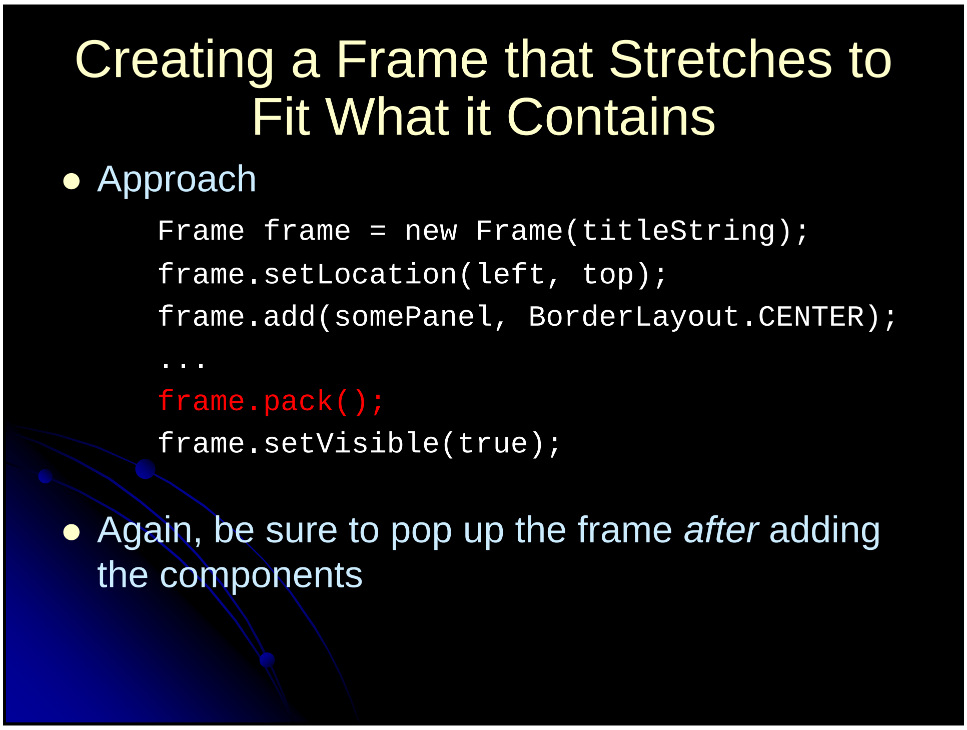


Creating a Fixed-Size Frame

- Approach

```
Frame frame = new Frame(titleString);
frame.add(somePanel,BorderLayout.CENTER);
frame.add(otherPanel, BorderLayout.NORTH);
...
frame.setSize(width, height);
frame.setVisible(true);
```

- Note: be sure you pop up the frame last
  - Odd behavior results if you add components to a window that is already visible (unless you call `doLayout` on the frame)

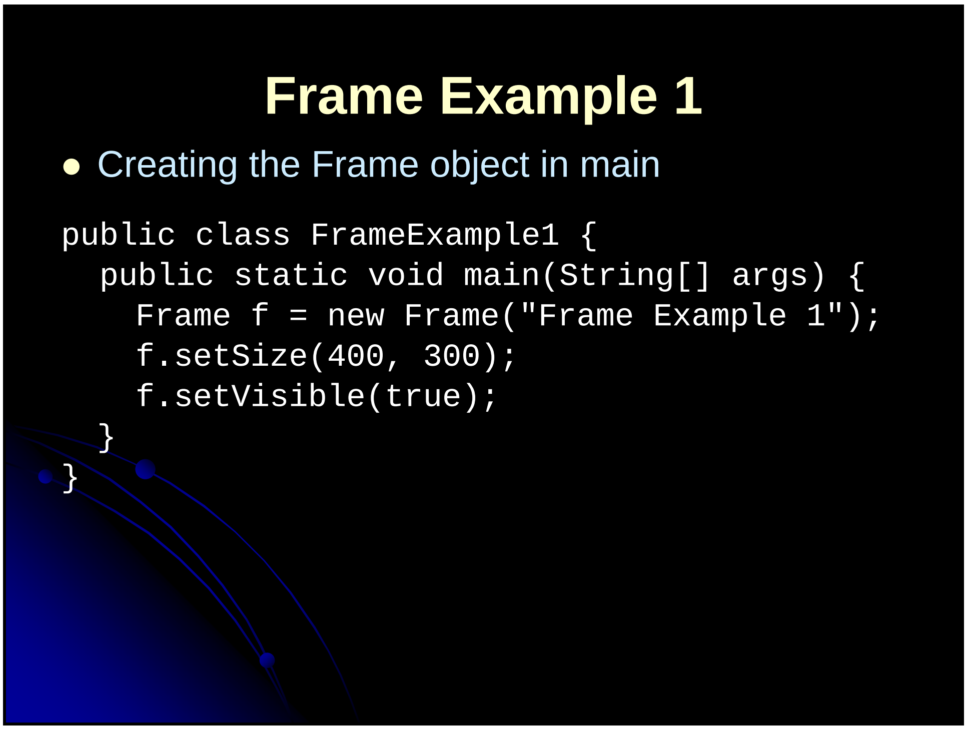


Creating a Frame that Stretches to Fit What it Contains

- Approach

```
Frame frame = new Frame(titleString);
frame.setLocation(left, top);
frame.add(somePanel, BorderLayout.CENTER);
...
frame.pack();
frame.setVisible(true);
```

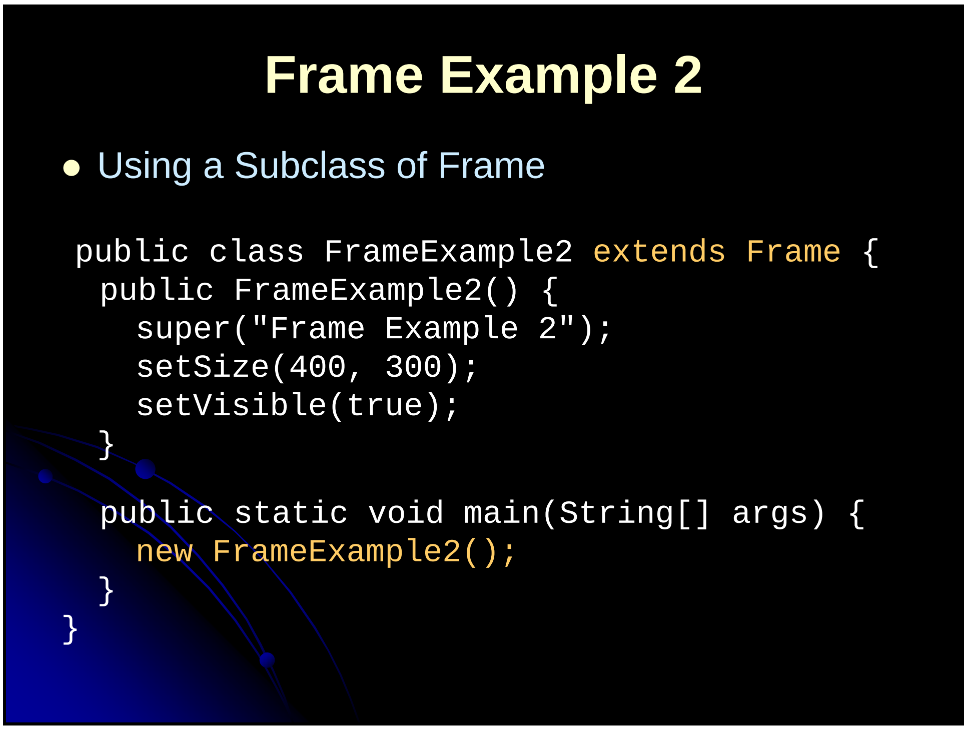
- Again, be sure to pop up the frame *after* adding the components



Frame Example 1

- Creating the Frame object in main

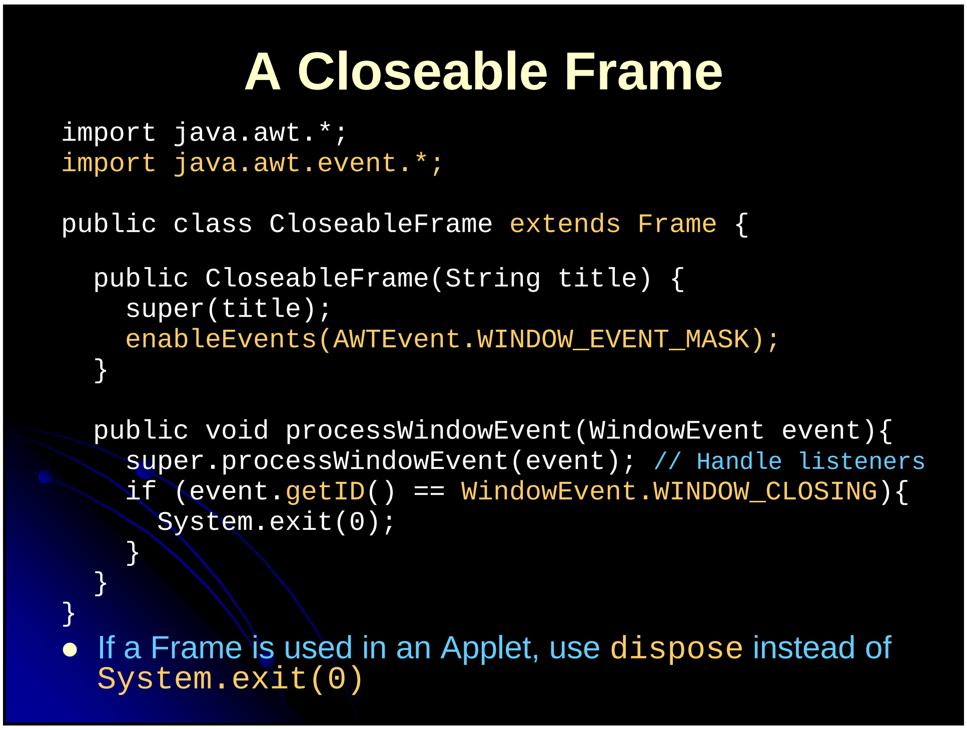
```
public class FrameExample1 {
  public static void main(String[] args) {
    Frame f = new Frame("Frame Example 1");
    f.setSize(400, 300);
    f.setVisible(true);
  }
}
```



Frame Example 2

- Using a Subclass of Frame

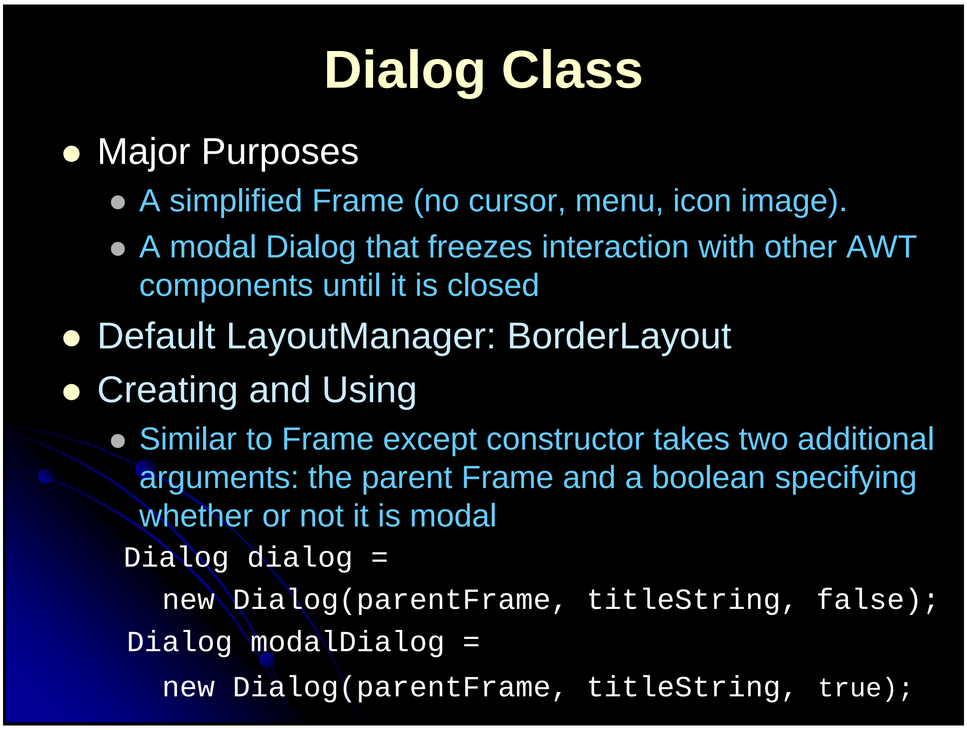
```
 public class FrameExample2 extends Frame {
   public FrameExample2() {
     super("Frame Example 2");
     setSize(400, 300);
     setVisible(true);
   }

   public static void main(String[] args) {
     new FrameExample2();
   }
}
```



A Closeable Frame

```
import java.awt.*;
import java.awt.event.*;

public class CloseableFrame extends Frame {

  public CloseableFrame(String title) {
    super(title);
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  }

  public void processWindowEvent(WindowEvent event){
    super.processWindowEvent(event); // Handle listeners
    if (event.getID() == WindowEvent.WINDOW_CLOSING){
      System.exit(0);
    }
  }
}
```

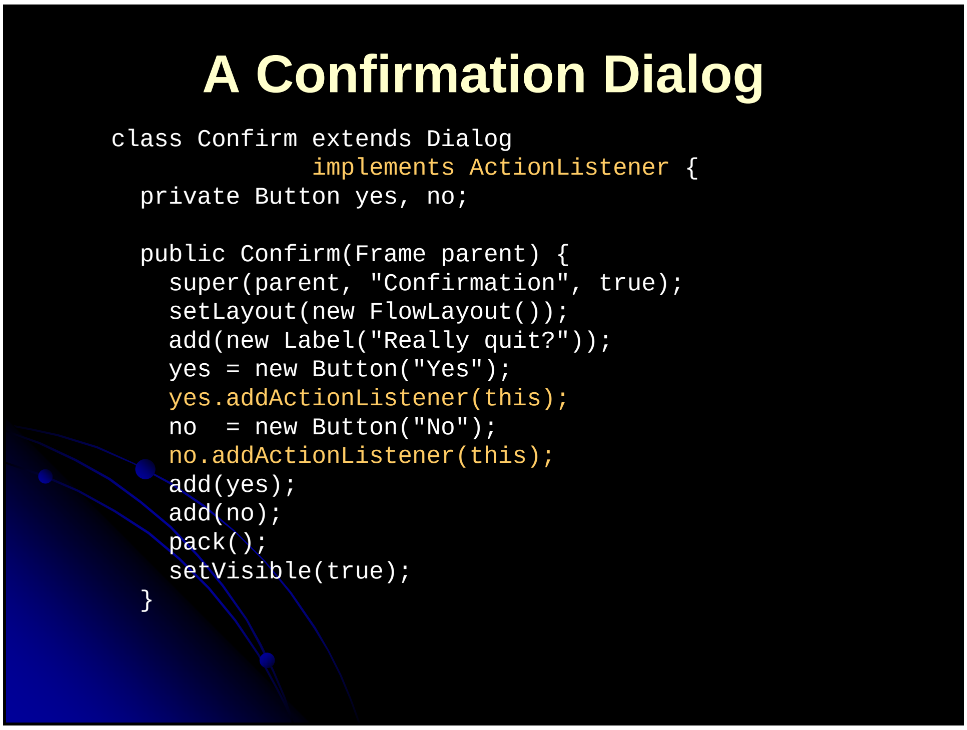
- If a Frame is used in an Applet, use `dispose` instead of `System.exit(0)`



Dialog Class

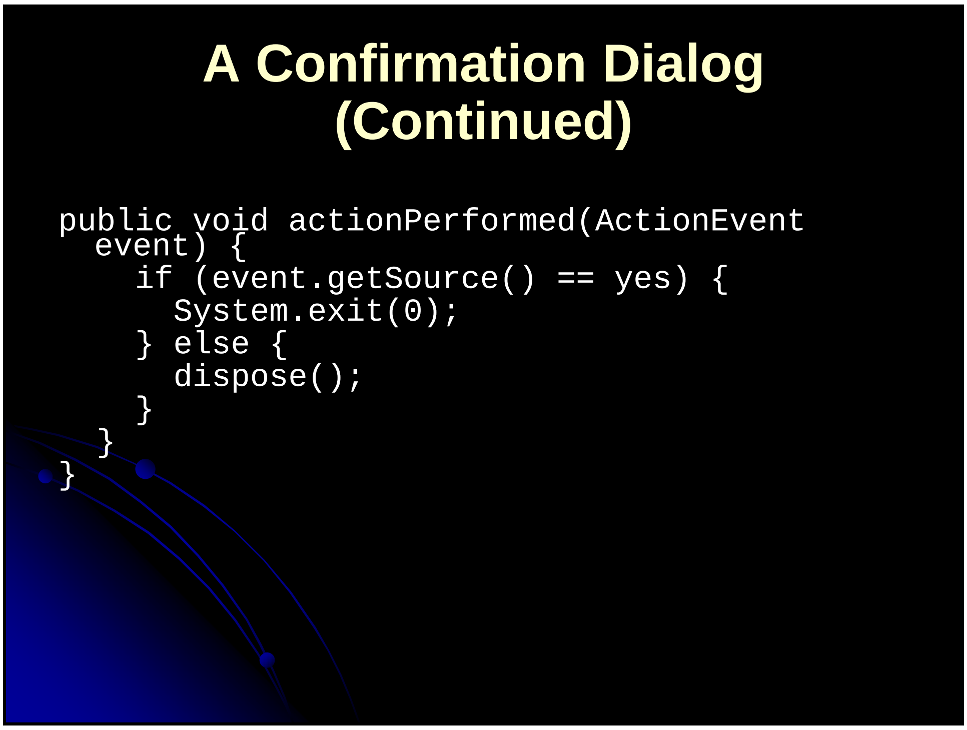
- Major Purposes
  - A simplified Frame (no cursor, menu, icon image).
  - A modal Dialog that freezes interaction with other AWT components until it is closed
- Default LayoutManager: BorderLayout
- Creating and Using
  - Similar to Frame except constructor takes two additional arguments: the parent Frame and a boolean specifying whether or not it is modal

```
Dialog dialog =
  new Dialog(parentFrame, titleString, false);
Dialog modalDialog =
  new Dialog(parentFrame, titleString, true);
```



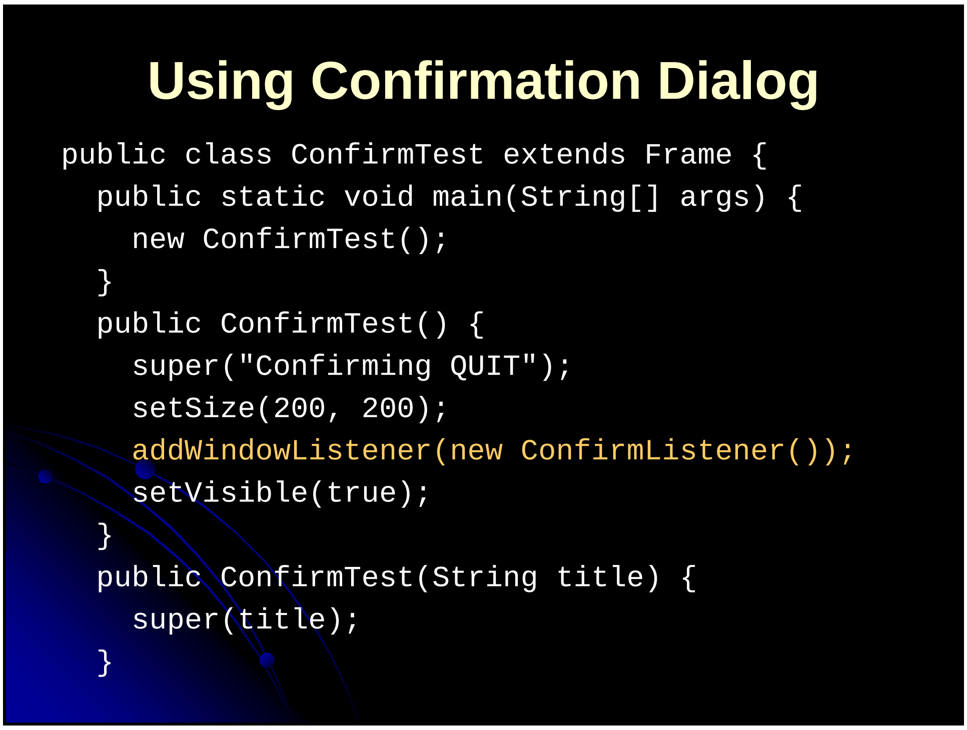
A Confirmation Dialog

```
class Confirm extends Dialog
              implements ActionListener {
  private Button yes, no;

  public Confirm(Frame parent) {
    super(parent, "Confirmation", true);
    setLayout(new FlowLayout());
    add(new Label("Really quit?"));
    yes = new Button("Yes");
    yes.addActionListener(this);
    no  = new Button("No");
    no.addActionListener(this);
    add(yes);
    add(no);
    pack();
    setVisible(true);
  }
```



A Confirmation Dialog (Continued)

```
public void actionPerformed(ActionEvent
  event) {
    if (event.getSource() == yes) {
      System.exit(0);
    } else {
      dispose();
    }
  }
}
```



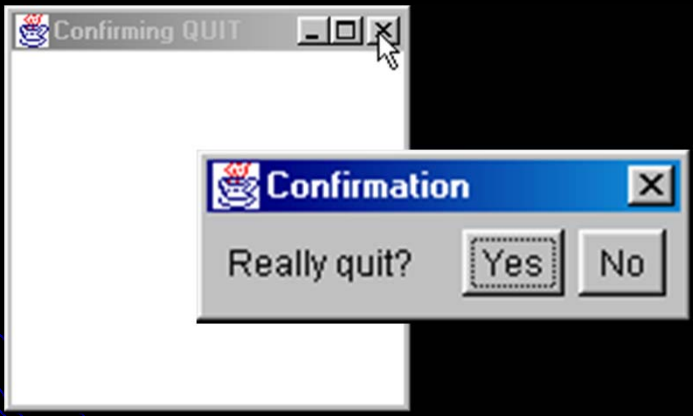
Using Confirmation Dialog

```
public class ConfirmTest extends Frame {
  public static void main(String[] args) {
    new ConfirmTest();
  }
  public ConfirmTest() {
    super("Confirming QUIT");
    setSize(200, 200);
    addWindowListener(new ConfirmListener());
    setVisible(true);
  }
  public ConfirmTest(String title) {
    super(title);
  }
```

Using Confirmation Dialog (Continued)

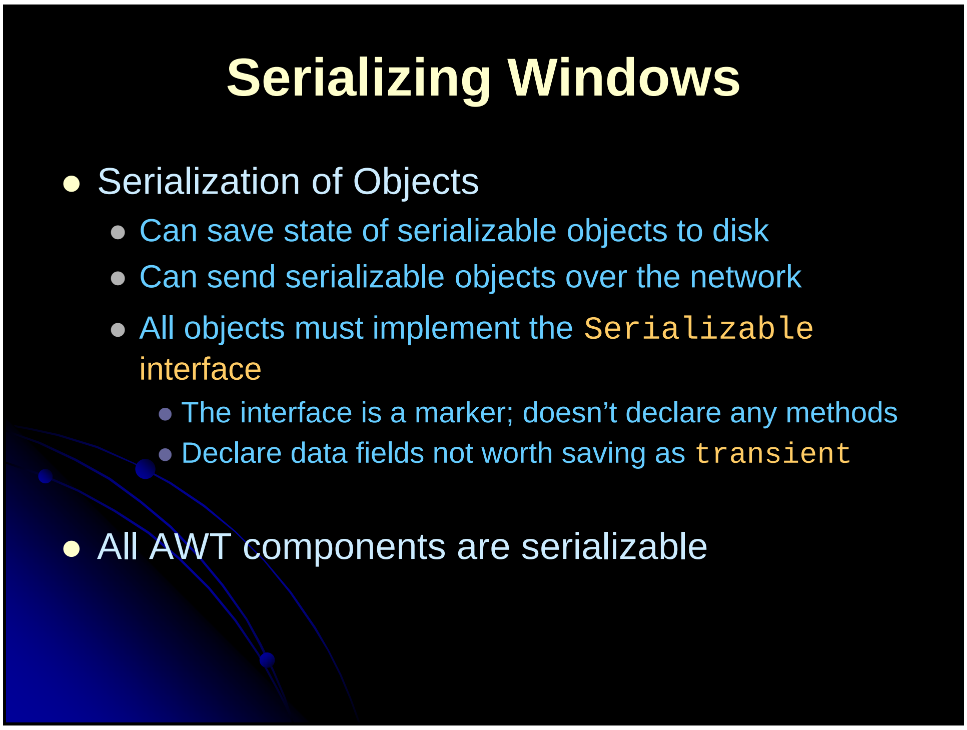
```
private class ConfirmListener extends
WindowAdapter {
  public void windowClosing(WindowEvent event) {
    new Confirm(ConfirmTest.this);
  }
}
}
```

A Confirmation Dialog: Result



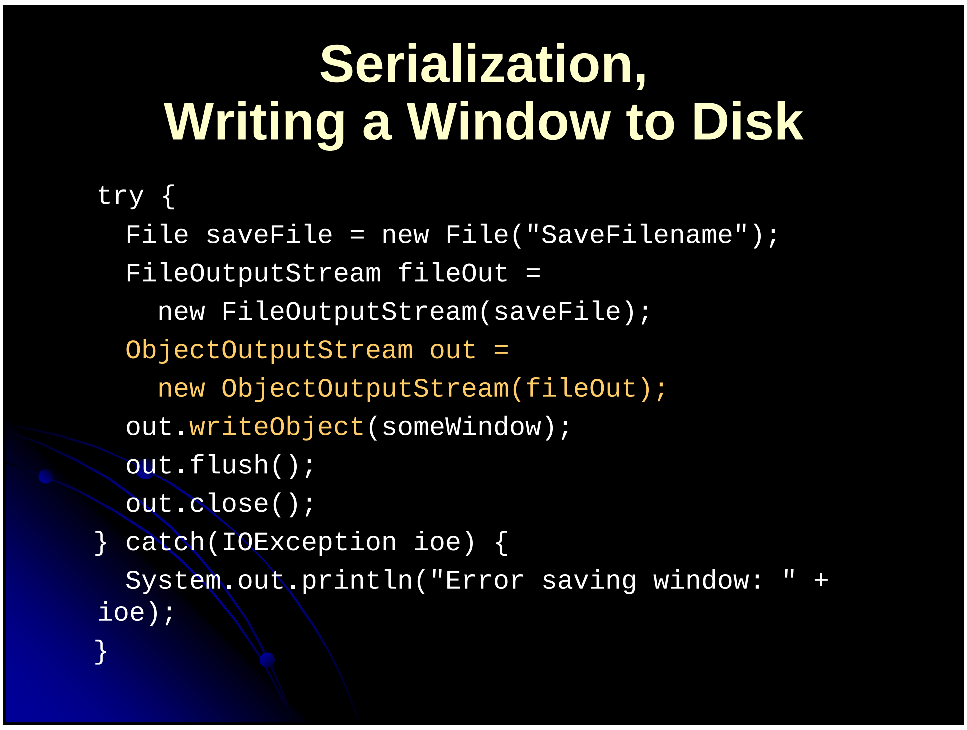
Really quit?

Modal dialogs freeze interaction with all other Java components



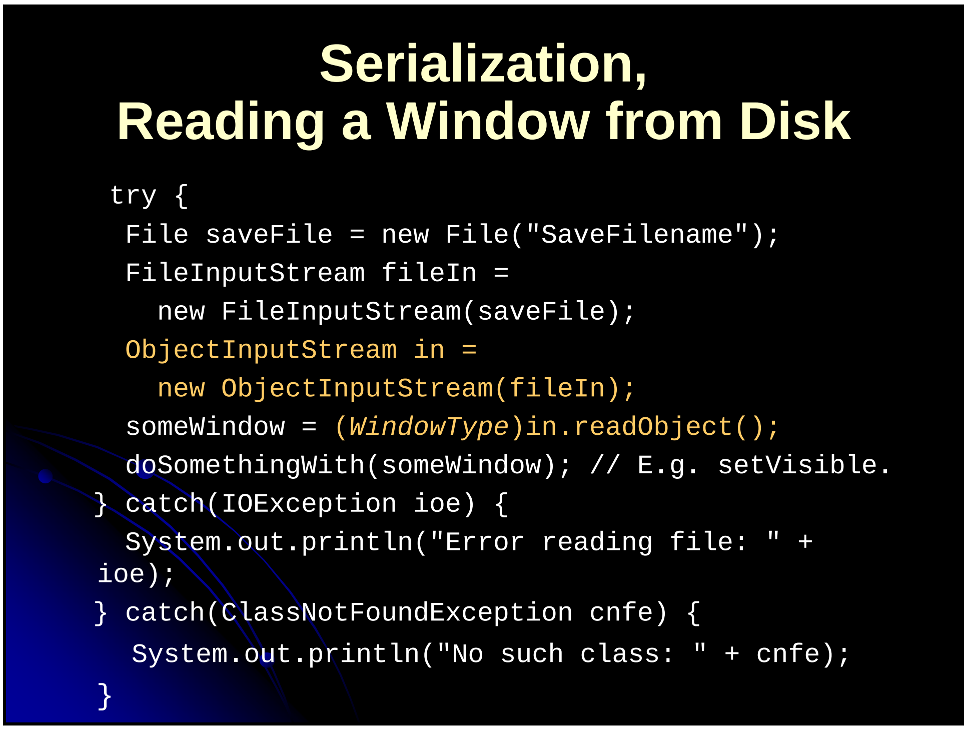
Serializing Windows

- Serialization of Objects
  - Can save state of serializable objects to disk
  - Can send serializable objects over the network
  - All objects must implement the `Serializable` interface
    - The interface is a marker; doesn't declare any methods
    - Declare data fields not worth saving as `transient`
- All AWT components are serializable



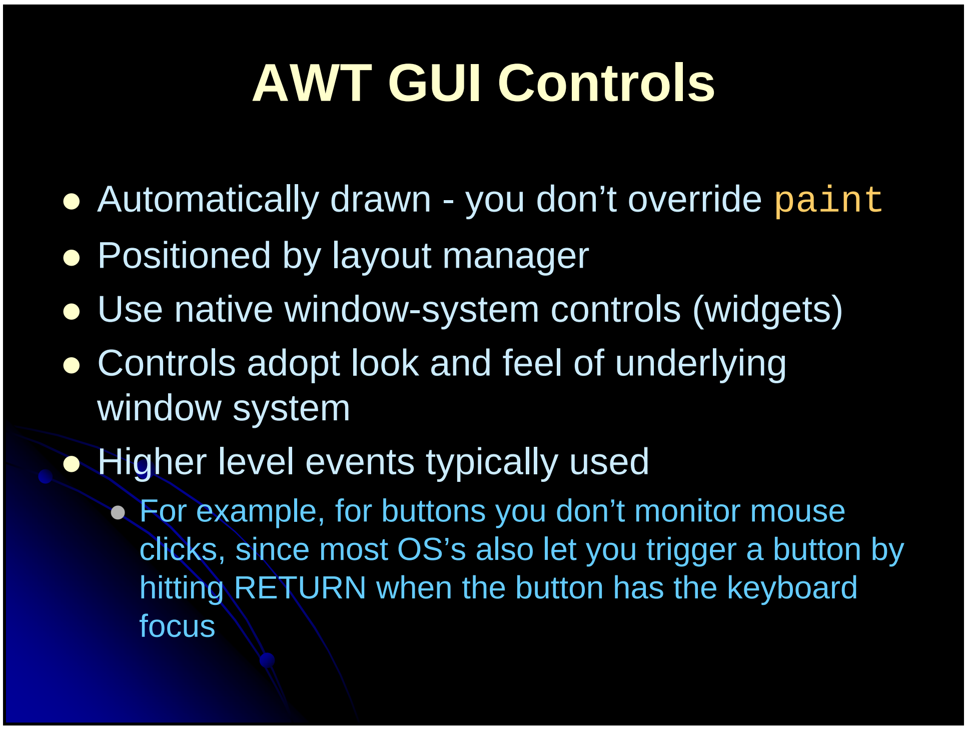
Serialization, Writing a Window to Disk

```
try {
  File saveFile = new File("SaveFilename");
  FileOutputStream fileOut =
    new FileOutputStream(saveFile);
  ObjectOutputStream out =
    new ObjectOutputStream(fileOut);
  out.writeObject(someWindow);
  out.flush();
  out.close();
} catch(IOException ioe) {
  System.out.println("Error saving window: " +
ioe);
}
```



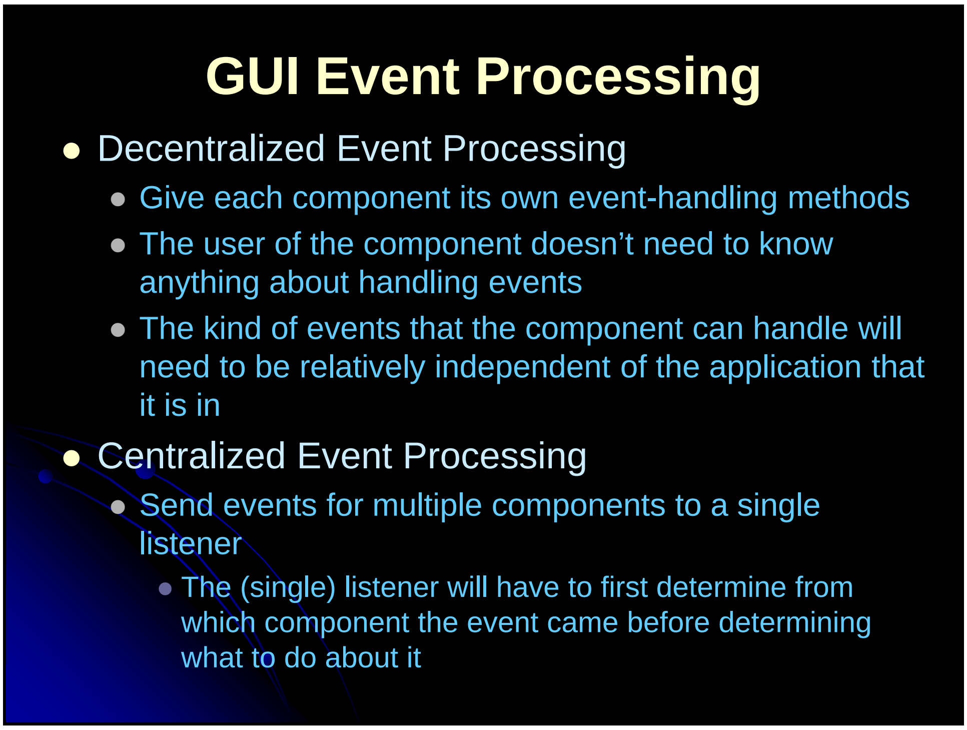
Serialization, Reading a Window from Disk

```
 try {
  File saveFile = new File("SaveFilename");
  FileInputStream fileIn =
    new FileInputStream(saveFile);
  ObjectInputStream in =
    new ObjectInputStream(fileIn);
  someWindow = (WindowType)in.readObject();
  doSomethingWith(someWindow); // E.g. setVisible.
} catch(IOException ioe) {
  System.out.println("Error reading file: " +
ioe);
} catch(ClassNotFoundException cnfe) {
   System.out.println("No such class: " + cnfe);
}
```



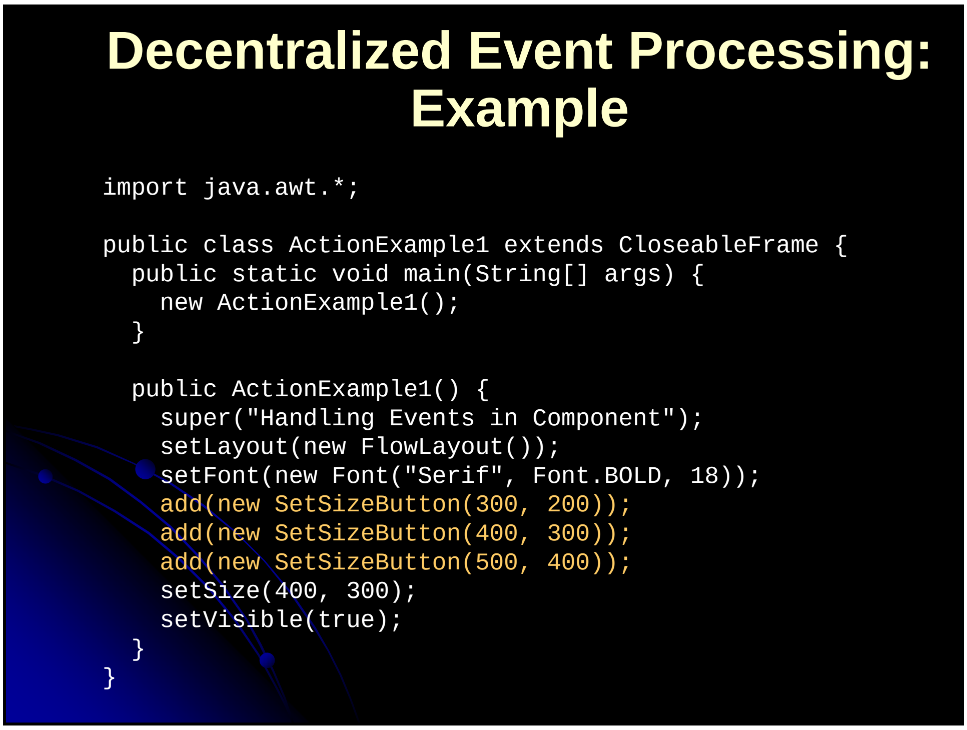
AWT GUI Controls

- Automatically drawn - you don't override `paint`
- Positioned by layout manager
- Use native window-system controls (widgets)
- Controls adopt look and feel of underlying window system
- Higher level events typically used
  - For example, for buttons you don't monitor mouse clicks, since most OS's also let you trigger a button by hitting RETURN when the button has the keyboard focus



GUI Event Processing

- Decentralized Event Processing
  - Give each component its own event-handling methods
  - The user of the component doesn’t need to know anything about handling events
  - The kind of events that the component can handle will need to be relatively independent of the application that it is in
- Centralized Event Processing
  - Send events for multiple components to a single listener
    - The (single) listener will have to first determine from which component the event came before determining what to do about it



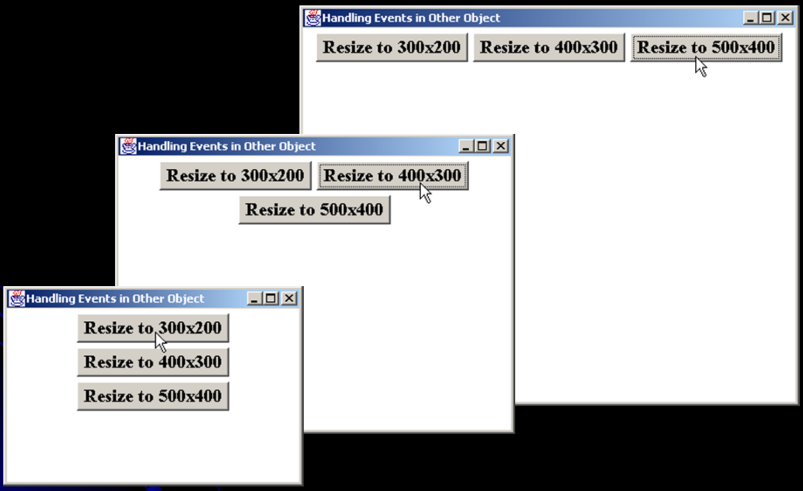
Decentralized Event Processing: Example

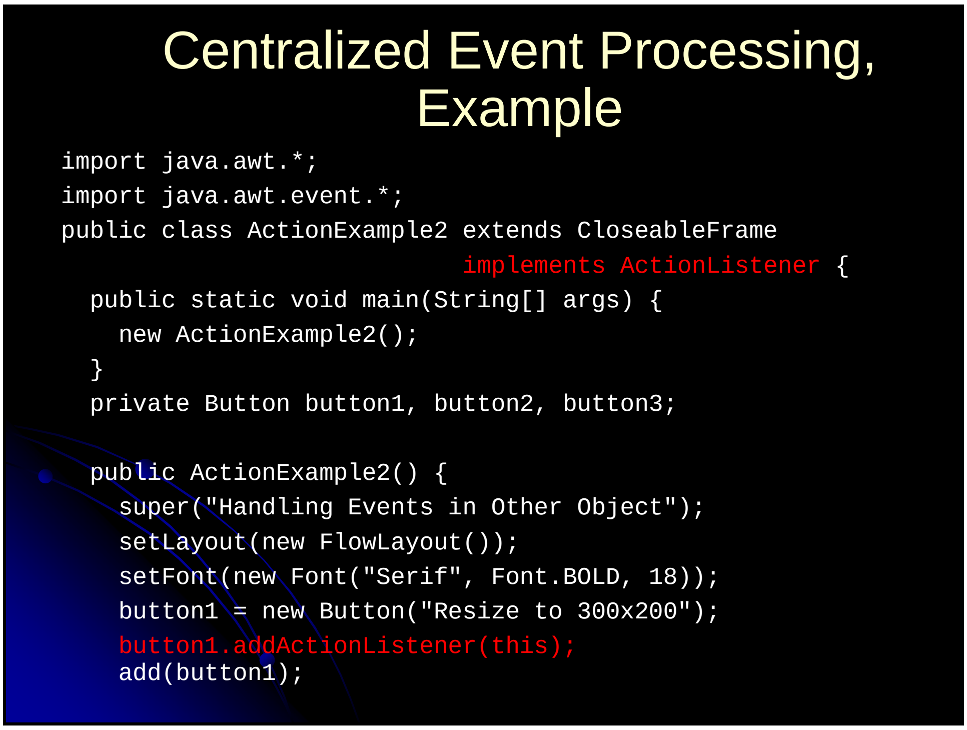
```
import java.awt.*;

public class ActionExample1 extends CloseableFrame {
  public static void main(String[] args) {
    new ActionExample1();
  }

  public ActionExample1() {
    super("Handling Events in Component");
    setLayout(new FlowLayout());
    setFont(new Font("Serif", Font.BOLD, 18));
    add(new SetSizeButton(300, 200));
    add(new SetSizeButton(400, 300));
    add(new SetSizeButton(500, 400));
    setSize(400, 300);
    setVisible(true);
  }
}
```

Decentralized Event Processing: Example (Continued)

```
import java.awt.*;
import java.awt.event.*;

public class SetSizeButton extends Button
                           implements ActionListener {
  private int width, height;

  public SetSizeButton(int width, int height) {
    super("Resize to " + width + "x" + height);
    this.width = width;
    this.height = height;
    addActionListener(this);
  }
  public void actionPerformed(ActionEvent event) {
    Container parent = getParent();
    parent.setSize(width, height);
    parent.invalidate();
    parent.validate();
  }
}
```

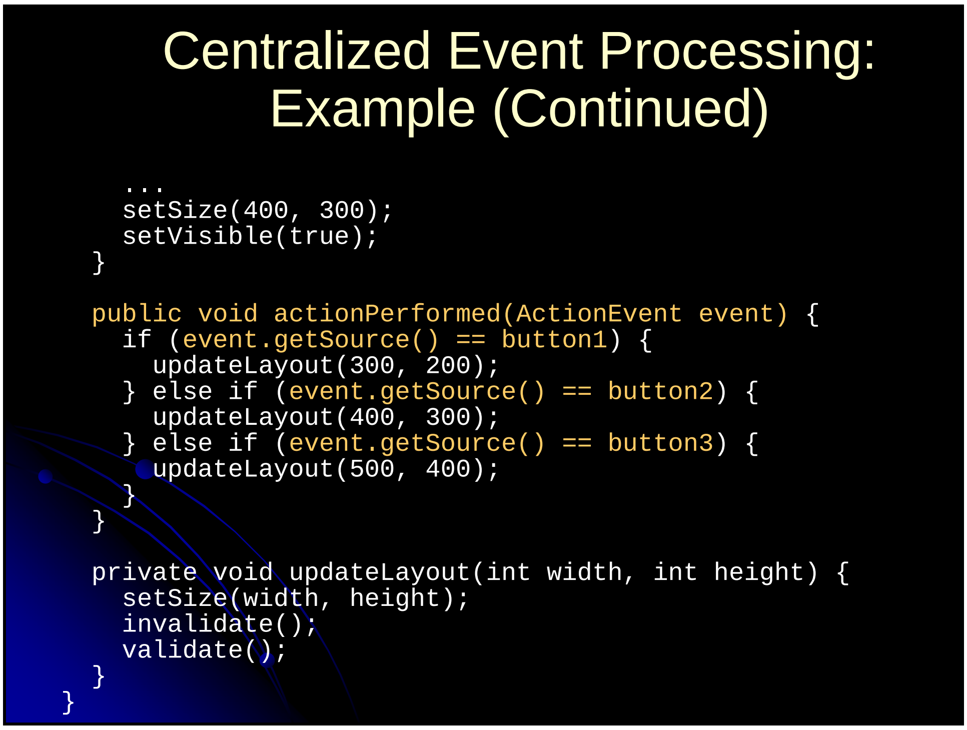
Decentralized Event Processing: Result





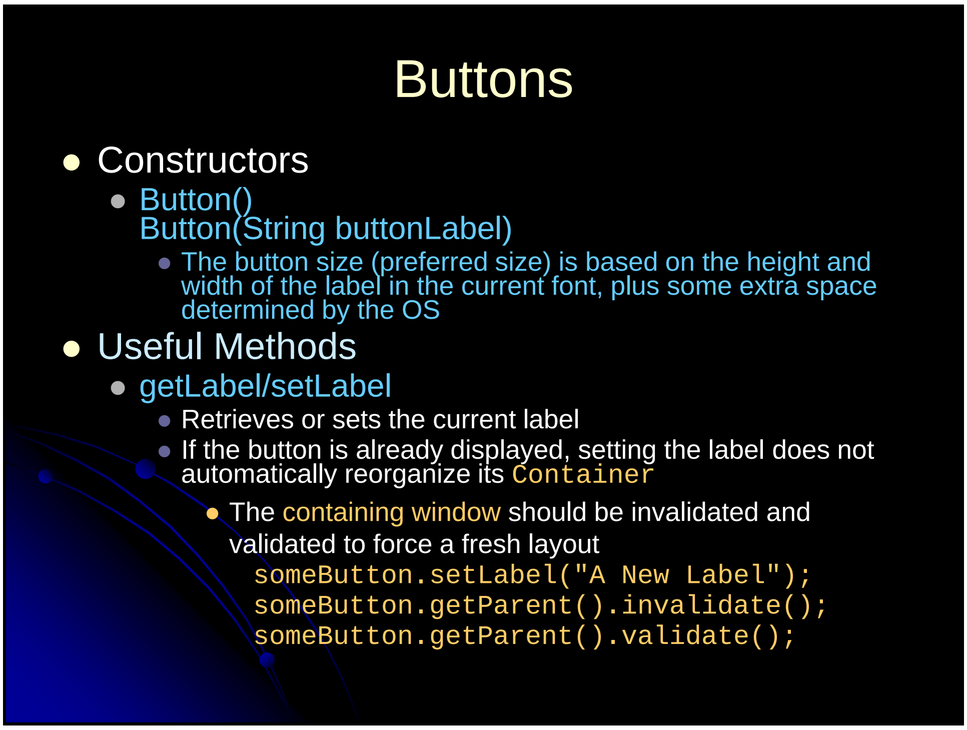
Centralized Event Processing, Example

```
import java.awt.*;
import java.awt.event.*;
public class ActionExample2 extends CloseableFrame
                            implements ActionListener {
  public static void main(String[] args) {
    new ActionExample2();
  }
  private Button button1, button2, button3;

  public ActionExample2() {
    super("Handling Events in Other Object");
    setLayout(new FlowLayout());
    setFont(new Font("Serif", Font.BOLD, 18));
    button1 = new Button("Resize to 300x200");
    button1.addActionListener(this);
    add(button1);
```



Centralized Event Processing: Example (Continued)

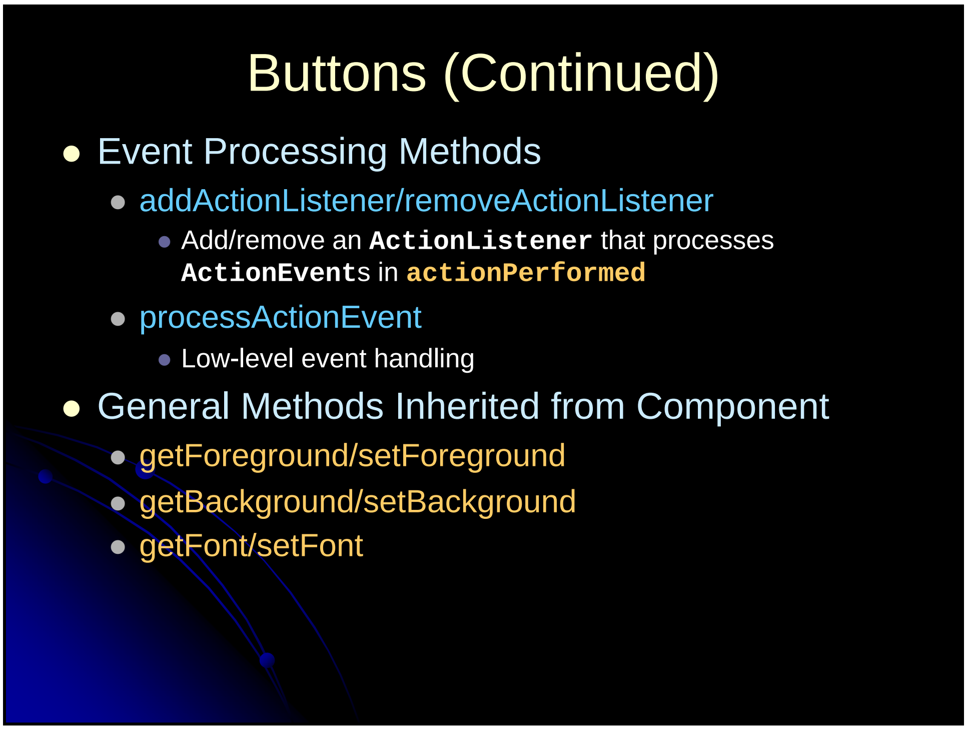
```
    ...
    setSize(400, 300);
    setVisible(true);
  }

  public void actionPerformed(ActionEvent event) {
    if (event.getSource() == button1) {
      updateLayout(300, 200);
    } else if (event.getSource() == button2) {
      updateLayout(400, 300);
    } else if (event.getSource() == button3) {
      updateLayout(500, 400);
    }
  }

  private void updateLayout(int width, int height) {
    setSize(width, height);
    invalidate();
    validate();
  }
}
```



Buttons

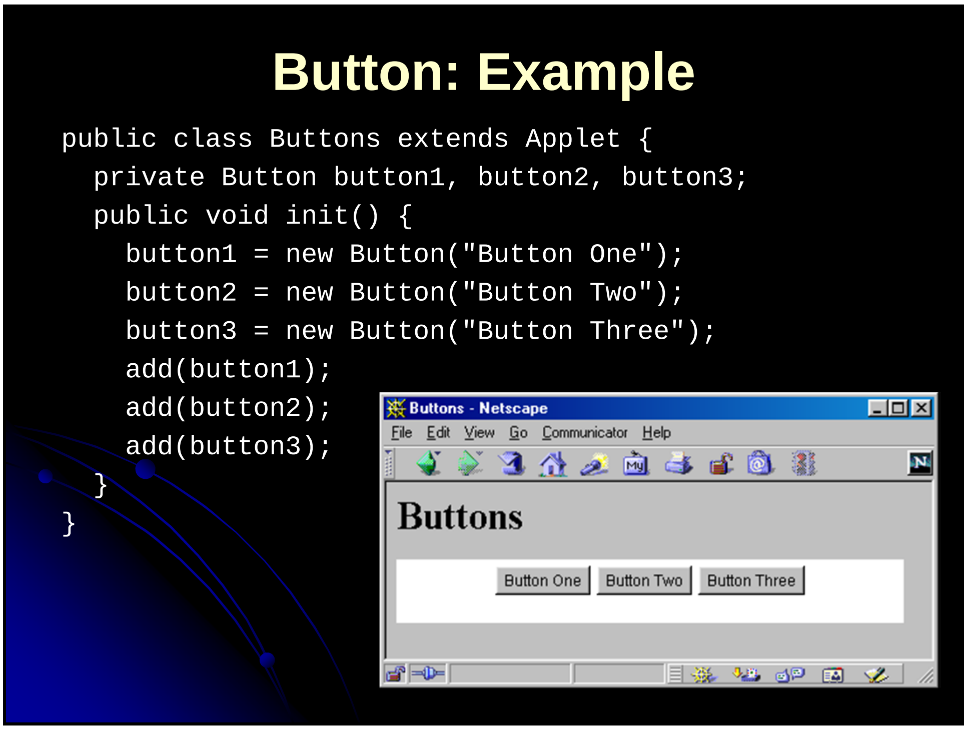
- Constructors
  - Button()
    Button(String buttonLabel)
    - The button size (preferred size) is based on the height and width of the label in the current font, plus some extra space determined by the OS
- Useful Methods
  - getLabel/setLabel
    - Retrieves or sets the current label
    - If the button is already displayed, setting the label does not automatically reorganize its `Container`
      - The containing window should be invalidated and validated to force a fresh layout

```
someButton.setLabel("A New Label");
someButton.getParent().invalidate();
someButton.getParent().validate();
```



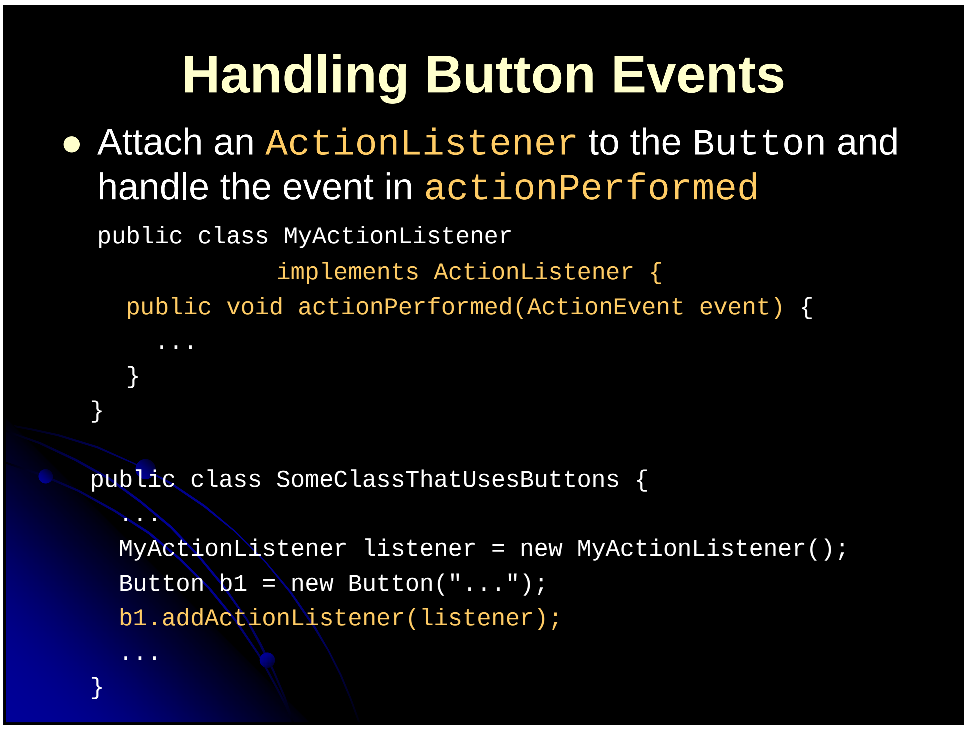
Buttons (Continued)

- Event Processing Methods
  - addActionListener/removeActionListener
    - Add/remove an **`ActionListener`** that processes **`ActionEvent`**s in **`actionPerformed`**
  - processActionEvent
    - Low-level event handling
- General Methods Inherited from Component
  - getForeground/setForeground
  - getBackground/setBackground
  - getFont/setFont



Button: Example

```
public class Buttons extends Applet {
  private Button button1, button2, button3;
  public void init() {
    button1 = new Button("Button One");
    button2 = new Button("Button Two");
    button3 = new Button("Button Three");
    add(button1);
    add(button2);
    add(button3);
  }
}
```



Handling Button Events

- Attach an `ActionListener` to the `Button` and handle the event in `actionPerformed`

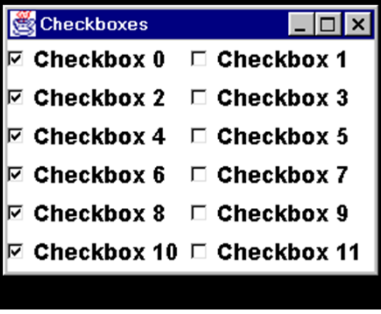
```
public class MyActionListener
             implements ActionListener {
  public void actionPerformed(ActionEvent event) {
    ...
  }
}

public class SomeClassThatUsesButtons {
  ...
  MyActionListener listener = new MyActionListener();
  Button b1 = new Button("...");
  b1.addActionListener(listener);
  ...
}
```

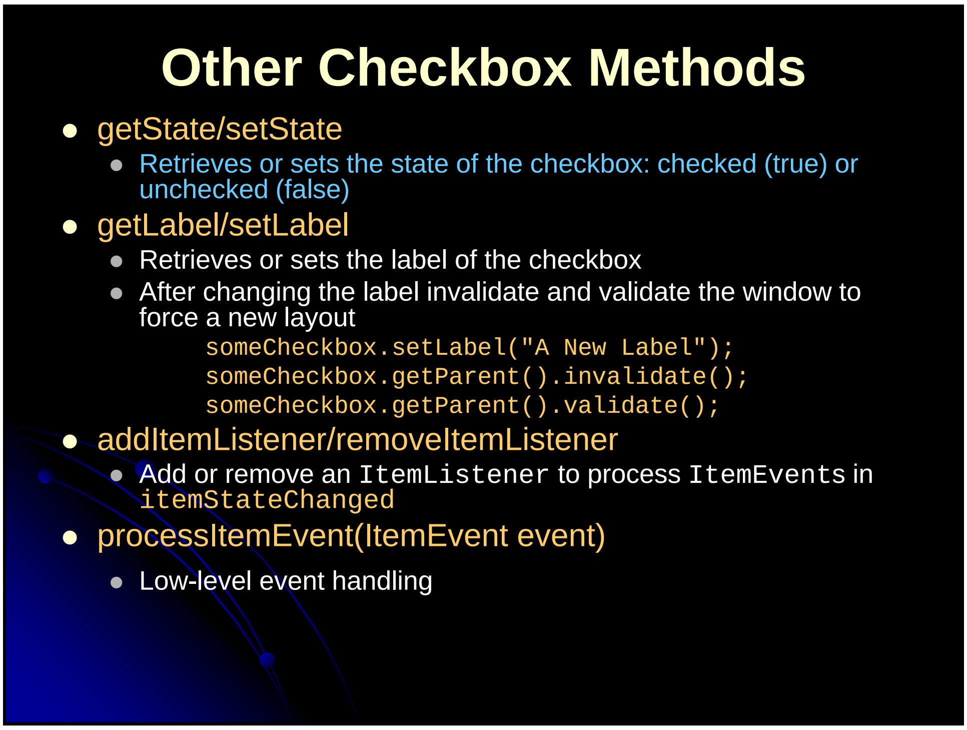
Checkboxes

- Constructors
  - These three constructors apply to checkboxes that operate independently of each other (i.e., not radio buttons)
  - Checkbox()
    - Creates an initially unchecked checkbox with no label
  - Checkbox(String checkboxLabel)
    - Creates a checkbox (initially unchecked) with the specified label; see `setState` for changing it
  - Checkbox(String checkboxLabel, boolean state)
    - Creates a checkbox with the specified label
      - The initial state is determined by the boolean value provided
      - A value of true means it is checked

Checkbox, Example

```
public class Checkboxes extends CloseableFrame {
  public Checkboxes() {
    super("Checkboxes");
    setFont(new Font("SansSerif", Font.BOLD, 18));
    setLayout(new GridLayout(0, 2));
    Checkbox box;
    for(int i=0; i<12; i++) {
      box = new Checkbox("Checkbox " + i);
      if (i%2 == 0) {
        box.setState(true);
      }
      add(box);
    }
    pack();
    setVisible(true);
  }
}
```



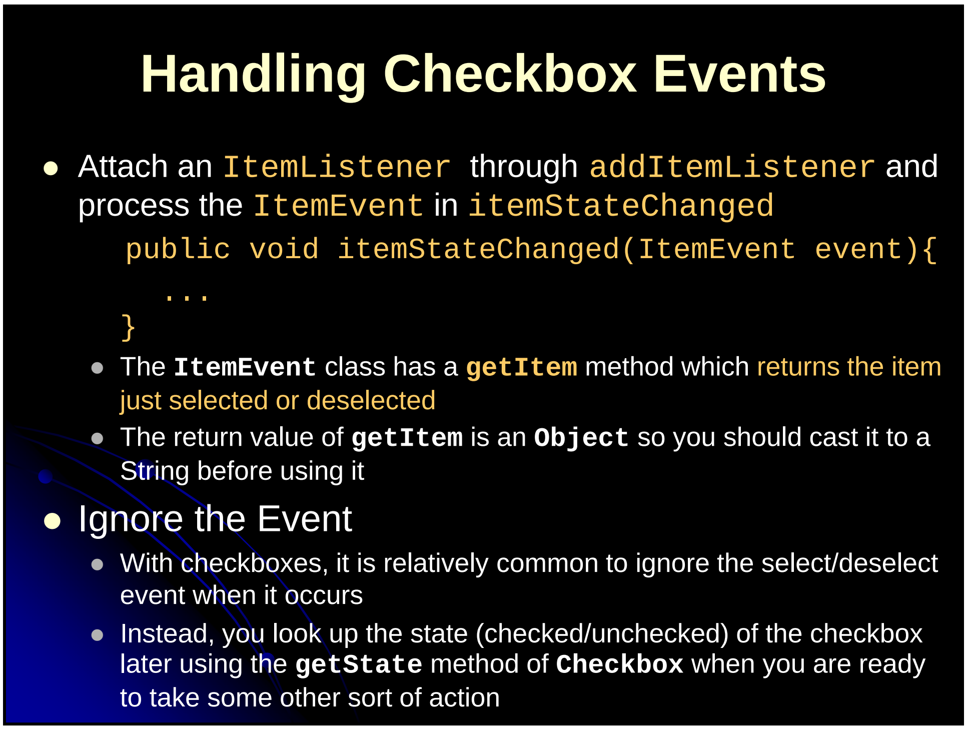


Other Checkbox Methods

- getState/setState
  - Retrieves or sets the state of the checkbox: checked (true) or unchecked (false)
- getLabel/setLabel
  - Retrieves or sets the label of the checkbox
  - After changing the label invalidate and validate the window to force a new layout

```
someCheckbox.setLabel("A New Label");
someCheckbox.getParent().invalidate();
someCheckbox.getParent().validate();
```

- addItemListener/removeItemListener
  - Add or remove an `ItemListener` to process `ItemEvents` in `itemStateChanged`
- processItemEvent(ItemEvent event)
  - Low-level event handling

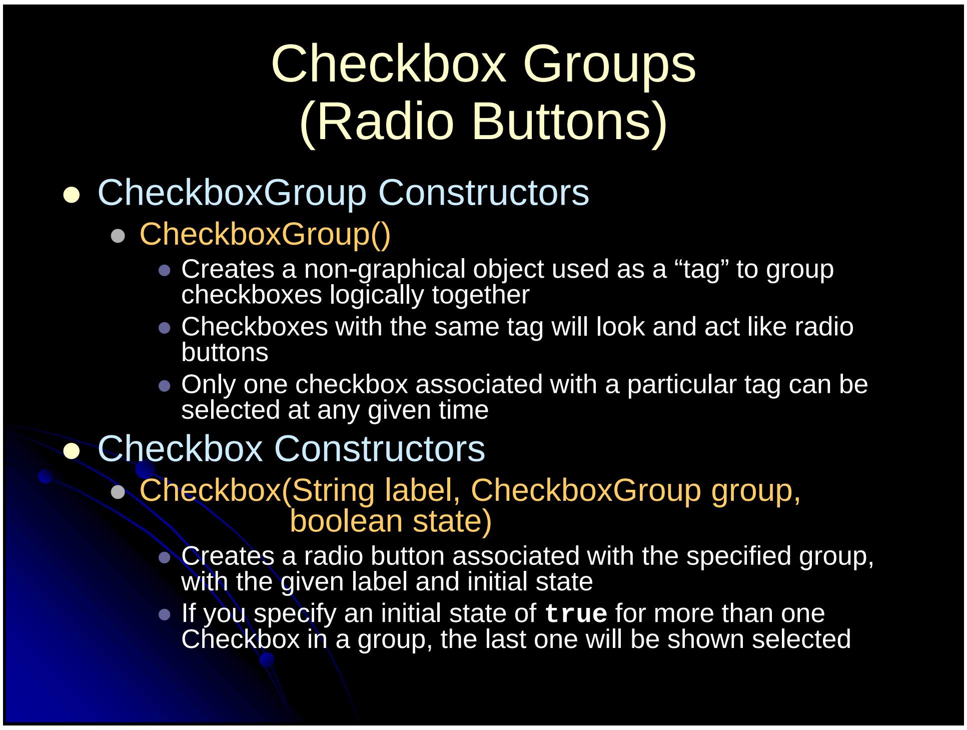


Handling Checkbox Events

- Attach an `ItemListener` through `addItemListener` and process the `ItemEvent` in `itemStateChanged`

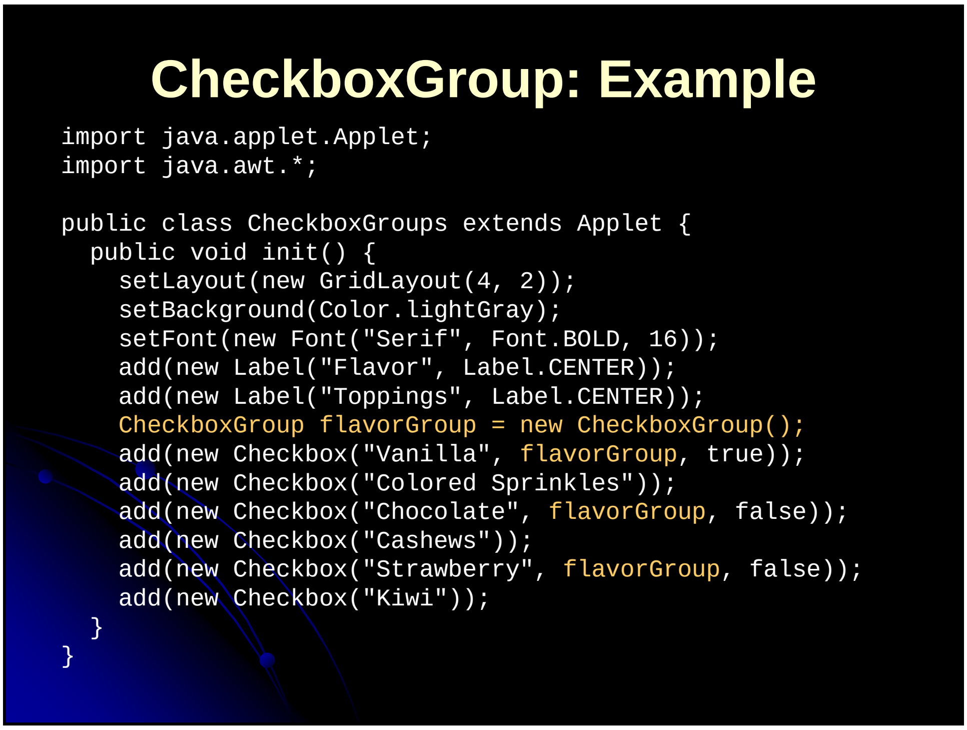
```
public void itemStateChanged(ItemEvent event){
  ...
}
```

  - The **`ItemEvent`** class has a **`getItem`** method which returns the item just selected or deselected
  - The return value of **`getItem`** is an **`Object`** so you should cast it to a String before using it
- Ignore the Event
  - With checkboxes, it is relatively common to ignore the select/deselect event when it occurs
  - Instead, you look up the state (checked/unchecked) of the checkbox later using the **`getState`** method of **`Checkbox`** when you are ready to take some other sort of action



Checkbox Groups (Radio Buttons)

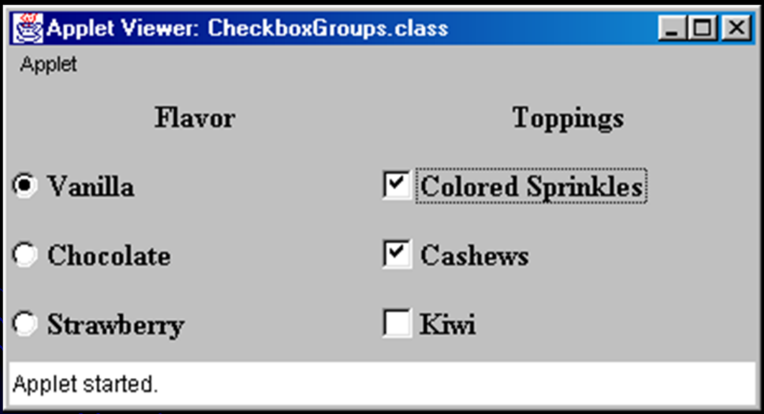
- CheckboxGroup Constructors
  - CheckboxGroup()
    - Creates a non-graphical object used as a “tag” to group checkboxes logically together
    - Checkboxes with the same tag will look and act like radio buttons
    - Only one checkbox associated with a particular tag can be selected at any given time
- Checkbox Constructors
  - Checkbox(String label, CheckboxGroup group, boolean state)
    - Creates a radio button associated with the specified group, with the given label and initial state
    - If you specify an initial state of **`true`** for more than one Checkbox in a group, the last one will be shown selected



CheckboxGroup: Example

```
import java.applet.Applet;
import java.awt.*;

public class CheckboxGroups extends Applet {
  public void init() {
    setLayout(new GridLayout(4, 2));
    setBackground(Color.lightGray);
    setFont(new Font("Serif", Font.BOLD, 16));
    add(new Label("Flavor", Label.CENTER));
    add(new Label("Toppings", Label.CENTER));
    CheckboxGroup flavorGroup = new CheckboxGroup();
    add(new Checkbox("Vanilla", flavorGroup, true));
    add(new Checkbox("Colored Sprinkles"));
    add(new Checkbox("Chocolate", flavorGroup, false));
    add(new Checkbox("Cashews"));
    add(new Checkbox("Strawberry", flavorGroup, false));
    add(new Checkbox("Kiwi"));
  }
}
```

CheckboxGroup, Result



Flavor

● Vanilla

○ Chocolate

○ Strawberry

Toppings

☑ Colored Sprinkles

☑ Cashews

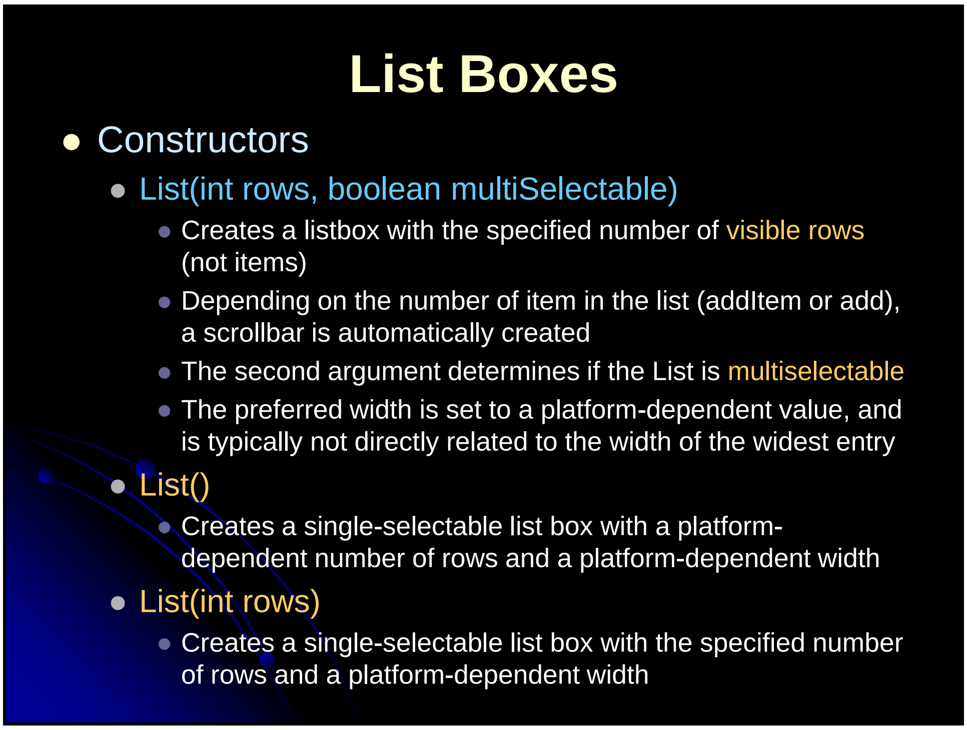
□ Kiwi

Applet started.

By tagging Checkboxes with a CheckboxGroup, the Checkboxes in the group function as radio buttons

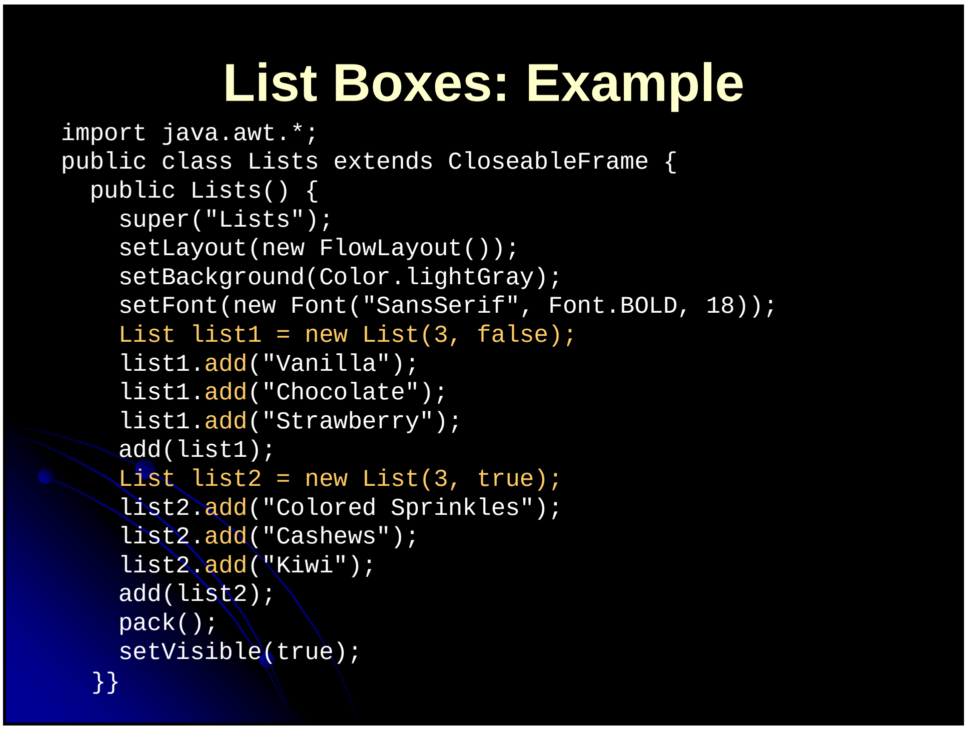
Other Methods for Radio Buttons

- CheckboxGroup
  - getSelectedCheckbox
    - Returns the radio button (**`Checkbox`**) that is currently selected or **`null`** if none is selected
- Checkbox
  - In addition to the general methods described in Checkboxes, `Checkbox` has the following two methods specific to CheckboxGroup's:
  - getCheckboxGroup/setCheckboxGroup
    - Determines or registers the group associated with the radio button
- Note: Event-handling is the same as with Checkboxes



List Boxes

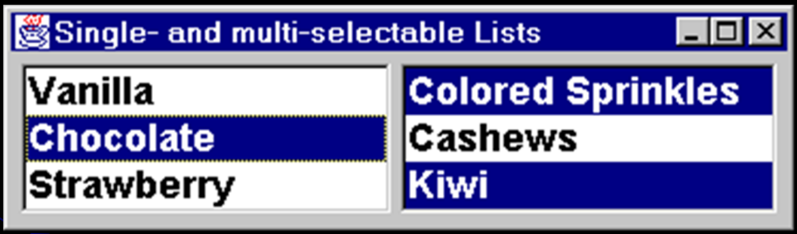
- Constructors
  - List(int rows, boolean multiSelectable)
    - Creates a listbox with the specified number of visible rows (not items)
    - Depending on the number of item in the list (addItem or add), a scrollbar is automatically created
    - The second argument determines if the List is multiselectable
    - The preferred width is set to a platform-dependent value, and is typically not directly related to the width of the widest entry
  - List()
    - Creates a single-selectable list box with a platform-dependent number of rows and a platform-dependent width
  - List(int rows)
    - Creates a single-selectable list box with the specified number of rows and a platform-dependent width



List Boxes: Example

```
import java.awt.*;
public class Lists extends CloseableFrame {
  public Lists() {
    super("Lists");
    setLayout(new FlowLayout());
    setBackground(Color.lightGray);
    setFont(new Font("SansSerif", Font.BOLD, 18));
    List list1 = new List(3, false);
    list1.add("Vanilla");
    list1.add("Chocolate");
    list1.add("Strawberry");
    add(list1);
    List list2 = new List(3, true);
    list2.add("Colored Sprinkles");
    list2.add("Cashews");
    list2.add("Kiwi");
    add(list2);
    pack();
    setVisible(true);
  }}
```

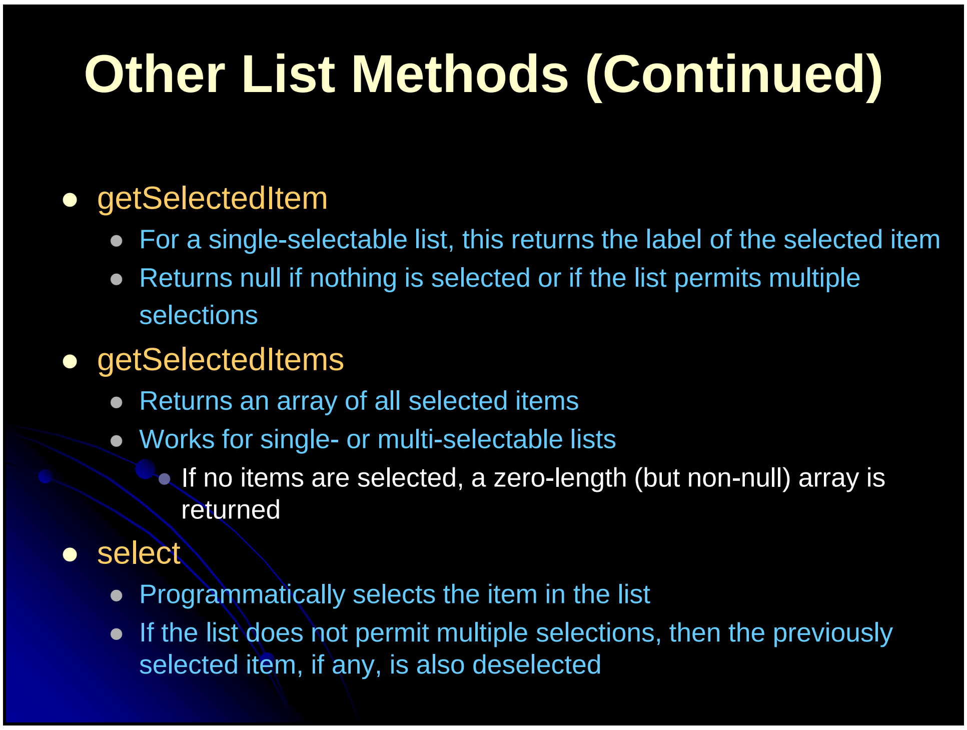
List Boxes: Result



A list can be *single*-selectable or *multi*-selectable

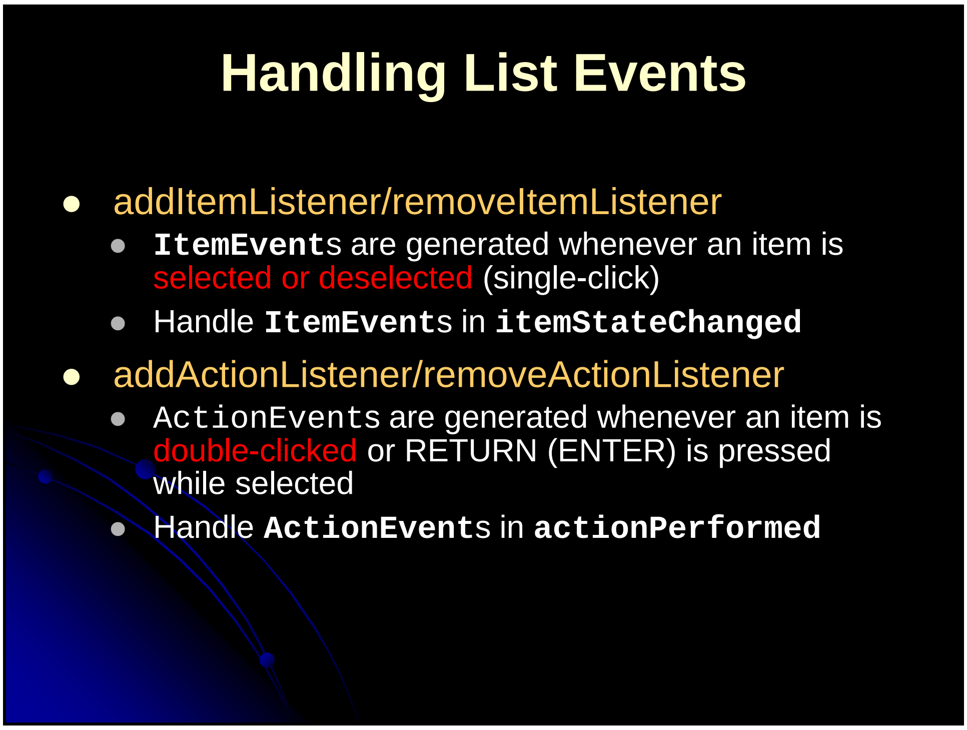
Other List Methods

- add
  - Add an item at the end or specified position in the list box
  - All items at that index or later get moved down
- isMultipleMode
  - Determines if the list is multiple selectable (`true`) or single selectable (`false`)
- remove/removeAll
  - Remove an item or all items from the list
- getSelectedIndex
  - For a single-selectable list, this returns the index of the selected item
  - Returns –1 if nothing is selected or if the list permits multiple selections
- getSelectedIndexes
  - Returns an array of the indexes of all selected items
    - Works for single- or multi-selectable lists
    - If no items are selected, a zero-length (but non-null) array is returned



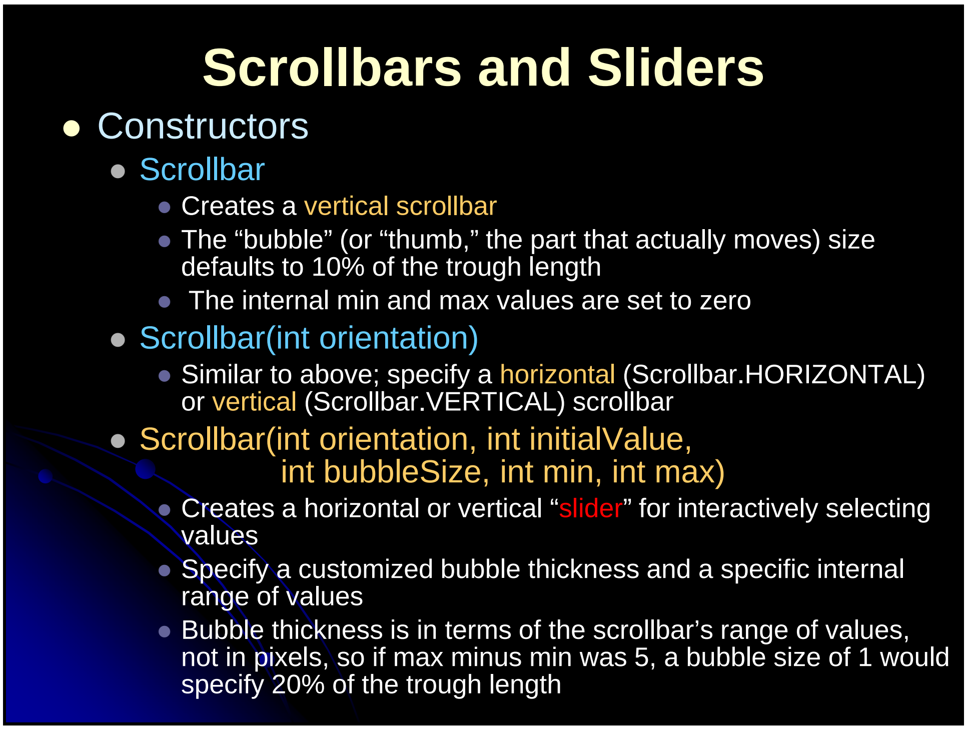
Other List Methods (Continued)

- getSelectedItem
  - For a single-selectable list, this returns the label of the selected item
  - Returns null if nothing is selected or if the list permits multiple selections
- getSelectedItems
  - Returns an array of all selected items
  - Works for single- or multi-selectable lists
    - If no items are selected, a zero-length (but non-null) array is returned
- select
  - Programmatically selects the item in the list
  - If the list does not permit multiple selections, then the previously selected item, if any, is also deselected



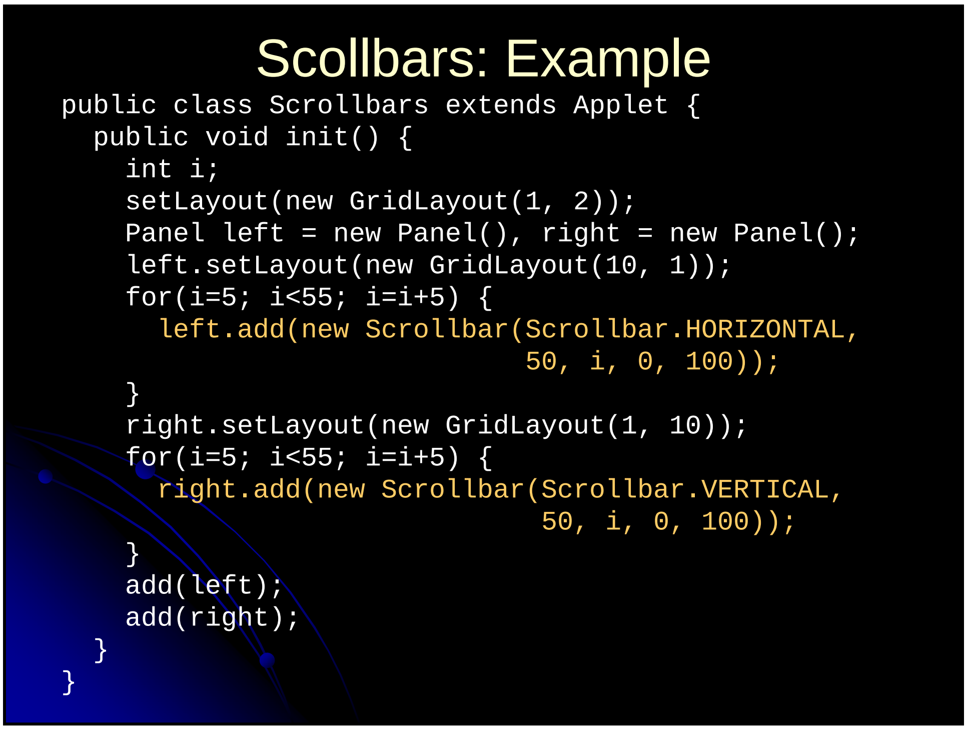
Handling List Events

- addItemListener/removeItemListener
  - **`ItemEvent`**s are generated whenever an item is selected or deselected (single-click)
  - Handle **`ItemEvent`**s in **`itemStateChanged`**
- addActionListener/removeActionListener
  - `ActionEvent`s are generated whenever an item is double-clicked or RETURN (ENTER) is pressed while selected
  - Handle **`ActionEvent`**s in **`actionPerformed`**



Scrollbars and Sliders

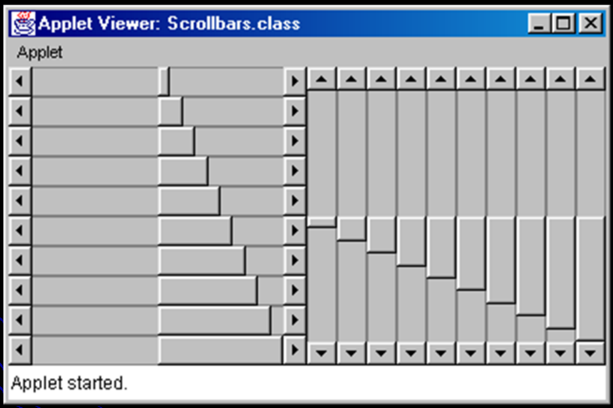
- Constructors
  - Scrollbar
    - Creates a vertical scrollbar
    - The “bubble” (or “thumb,” the part that actually moves) size defaults to 10% of the trough length
    - The internal min and max values are set to zero
  - Scrollbar(int orientation)
    - Similar to above; specify a horizontal (Scrollbar.HORIZONTAL) or vertical (Scrollbar.VERTICAL) scrollbar
  - Scrollbar(int orientation, int initialValue,
    int bubbleSize, int min, int max)
    - Creates a horizontal or vertical “slider” for interactively selecting values
    - Specify a customized bubble thickness and a specific internal range of values
    - Bubble thickness is in terms of the scrollbar’s range of values, not in pixels, so if max minus min was 5, a bubble size of 1 would specify 20% of the trough length



Scollbars: Example

```
public class Scrollbars extends Applet {
  public void init() {
    int i;
    setLayout(new GridLayout(1, 2));
    Panel left = new Panel(), right = new Panel();
    left.setLayout(new GridLayout(10, 1));
    for(i=5; i<55; i=i+5) {
      left.add(new Scrollbar(Scrollbar.HORIZONTAL,
                             50, i, 0, 100));
    }
    right.setLayout(new GridLayout(1, 10));
    for(i=5; i<55; i=i+5) {
      right.add(new Scrollbar(Scrollbar.VERTICAL,
                              50, i, 0, 100));
    }
    add(left);
    add(right);
  }
}
```

Scrollbars: Result



Scrollbars with varying bubble sizes, but constant ranges and initial values, shown on Windows 98

Handling Scrollbar Events

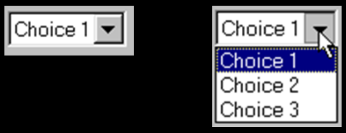
- AdjustmentListener
  - Attach an `AdjustmentListener` through `addAdjustmentListener` and process the `AdjustmentEvent` in `adjustmentValueChanged`

```
public void adjustmentValueChanged
                       (AdjustmentEvent event) {
    ...
}
```

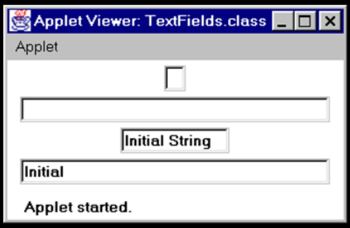
- Use ScrollPane
  - If you are using a Scrollbar only to implement scrolling, a ScrollPane is much simpler
- JSlider (Swing) is much better

Other GUI Controls

- Choice Lists (Combo Boxes)



- Textfields

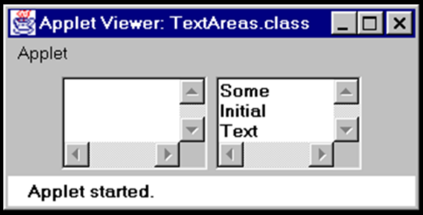

Applet Viewer: TextFields.class

Applet

Applet started.

Other GUI Controls (Continued)

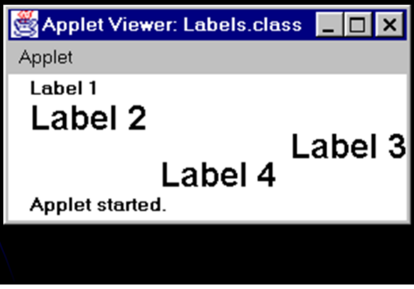
- Text Areas


Applet Viewer: TextAreas.class

Applet

Applet started.

- Labels


Applet Viewer: Labels.class

Applet

Label 1

Label 2

Label 3

Label 4

Applet started.

Summary

- In the AWT, all windows and graphical components are rectangular and opaque
- Canvas: drawing area or custom component
- Panel: grouping other components
- Frame: popup window
- Button: handle events with ActionListener
- Checkbox, radio button: handle events with ItemListener
- List box: handle single click with ItemListener, double click with ActionListener
- To quickly determine the event handlers for a component, simply look at the online API
  - add*Xxx*Listener methods are at the top

Thank you for your attention!

Layout Managers

Arranging Elements in Windows

Agenda

- How layout managers simplify interface design
- Standard layout managers
  - FlowLayout, BorderLayout, CardLayout, GridLayout, GridBagLayout, BoxLayout
- Positioning components manually
- Strategies for using layout managers effectively
- Using invisible components

Layout Managers

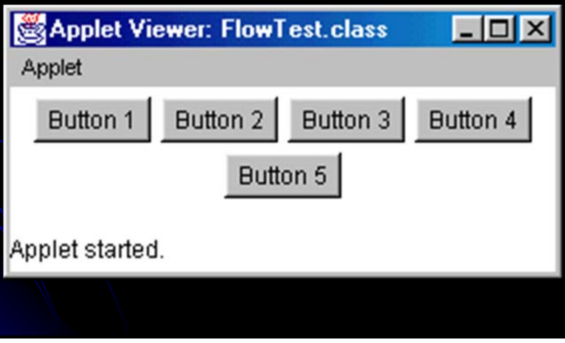
- Assigned to each Container
  - Give *sizes* and *positions* to components in the window
  - Helpful for windows whose size changes or that display on multiple operating systems
- Relatively easy for simple layouts
  - But, it is surprisingly hard to get complex layouts with a single layout manager
- Controlling complex layouts
  - Use nested containers (each with its own layout manager)
  - Use invisible components and layout manager options
  - Write your own layout manager
  - Turn some layout managers off and arrange some things manually

FlowLayout

- Default layout for `Panel` and `Applet`
- Behavior
  - Resizes components to their *preferred* size
  - Places components in rows *left to right, top to bottom*
  - Rows are *centered* by default
- Constructors
  - `FlowLayout()`
    - Centers each row and keeps 5 pixels between entries in a row and between rows
  - `FlowLayout(int alignment)`
    - Same 5 pixels spacing, but changes the alignment of the rows
    - `FlowLayout.LEFT, FlowLayout.RIGHT, FlowLayout.CENTER`
  - `FlowLayout(int alignment, int hGap, int vGap)`
    - Specify the alignment as well as the horizontal and vertical spacing between components (in pixels)

FlowLayout: Example

```
public class FlowTest extends Applet {
  public void init() {
    // setLayout(new FlowLayout()); [Default]
    for(int i=1; i<6; i++) {
      add(new Button("Button " + i));
    }
  }
}
```



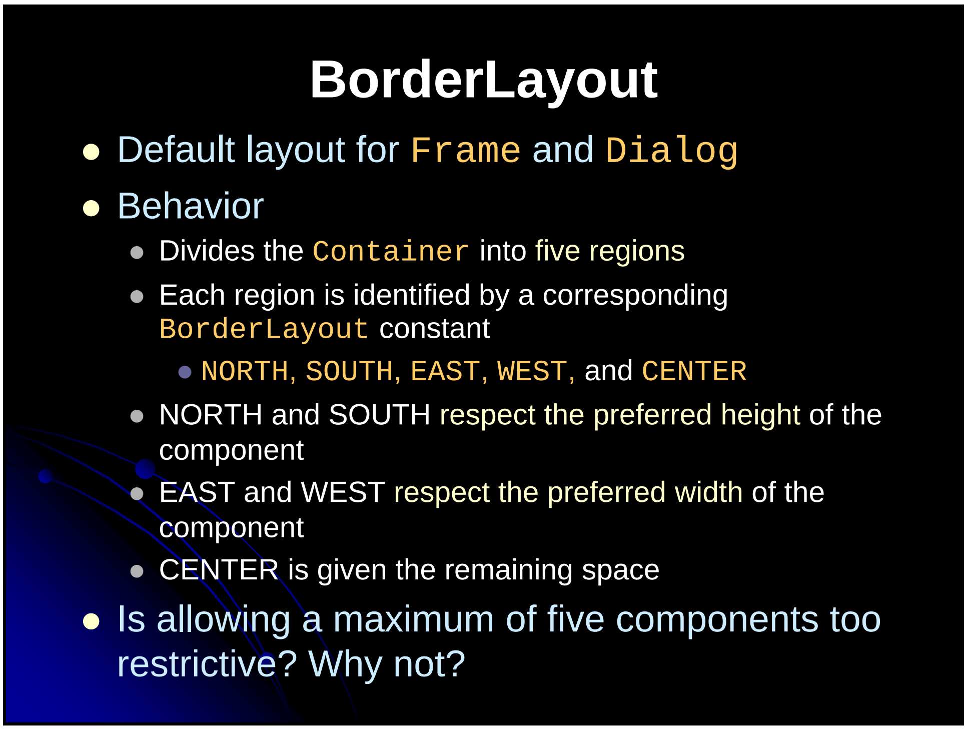
Applet started.

Testing the `FlowLayout` Manager

The components are arranged in the container from left to right in the order in which they were added. When one row becomes filled, a new row is started.

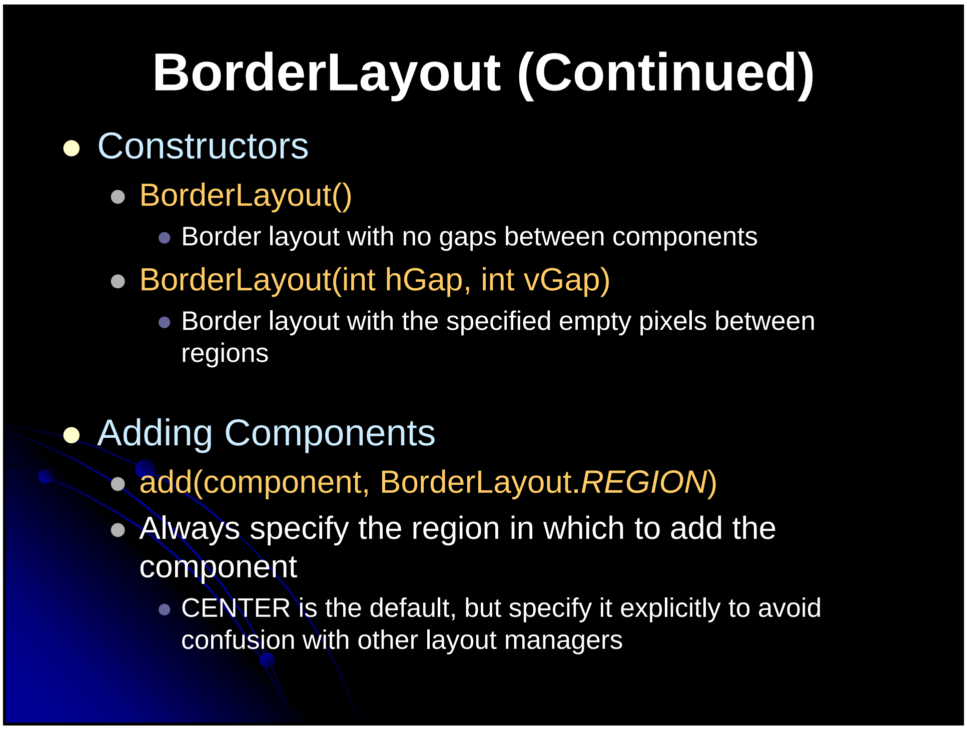
ShowFlowLayout

Run



BorderLayout

- Default layout for `Frame` and `Dialog`
- Behavior
  - Divides the `Container` into five regions
  - Each region is identified by a corresponding `BorderLayout` constant
    - `NORTH`, `SOUTH`, `EAST`, `WEST`, and `CENTER`
  - NORTH and SOUTH respect the preferred height of the component
  - EAST and WEST respect the preferred width of the component
  - CENTER is given the remaining space
- Is allowing a maximum of five components too restrictive? Why not?

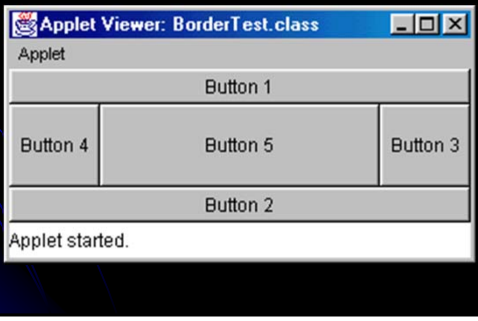


BorderLayout (Continued)

- Constructors
  - BorderLayout()
    - Border layout with no gaps between components
  - BorderLayout(int hGap, int vGap)
    - Border layout with the specified empty pixels between regions
- Adding Components
  - add(component, BorderLayout.*REGION*)
  - Always specify the region in which to add the component
    - CENTER is the default, but specify it explicitly to avoid confusion with other layout managers

BorderLayout: Example

```
public class BorderTest extends Applet {
  public void init() {
    setLayout(new BorderLayout());
    add(new Button("Button 1"), BorderLayout.NORTH);
    add(new Button("Button 2"), BorderLayout.SOUTH);
    add(new Button("Button 3"), BorderLayout.EAST);
    add(new Button("Button 4"), BorderLayout.WEST);
    add(new Button("Button 5"), BorderLayout.CENTER);
  }
}
```



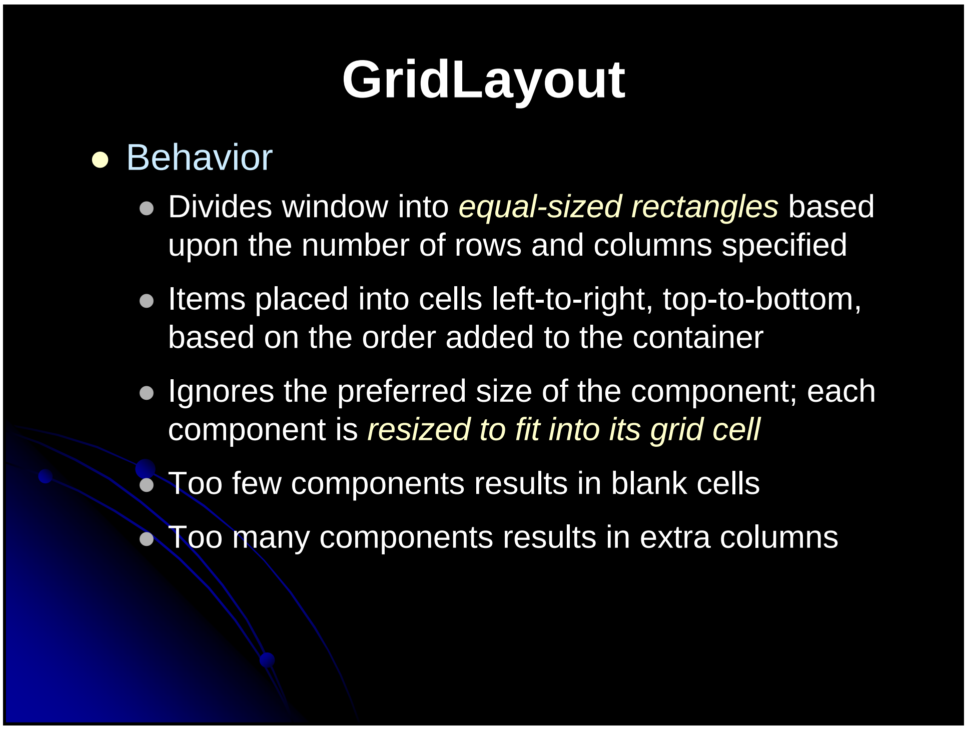
Applet started.

Testing the `BorderLayout` Manager

The `BorderLayout` manager divides the window into five areas: East, South, West, North, and Center. Components are added to a `BorderLayout` by using `add(Component, constraint)`, where `constraint` is `BorderLayout.East`, `BorderLayout.South`, `BorderLayout.West"`, `BorderLayout.North"`, or `BorderLayout.Center`.

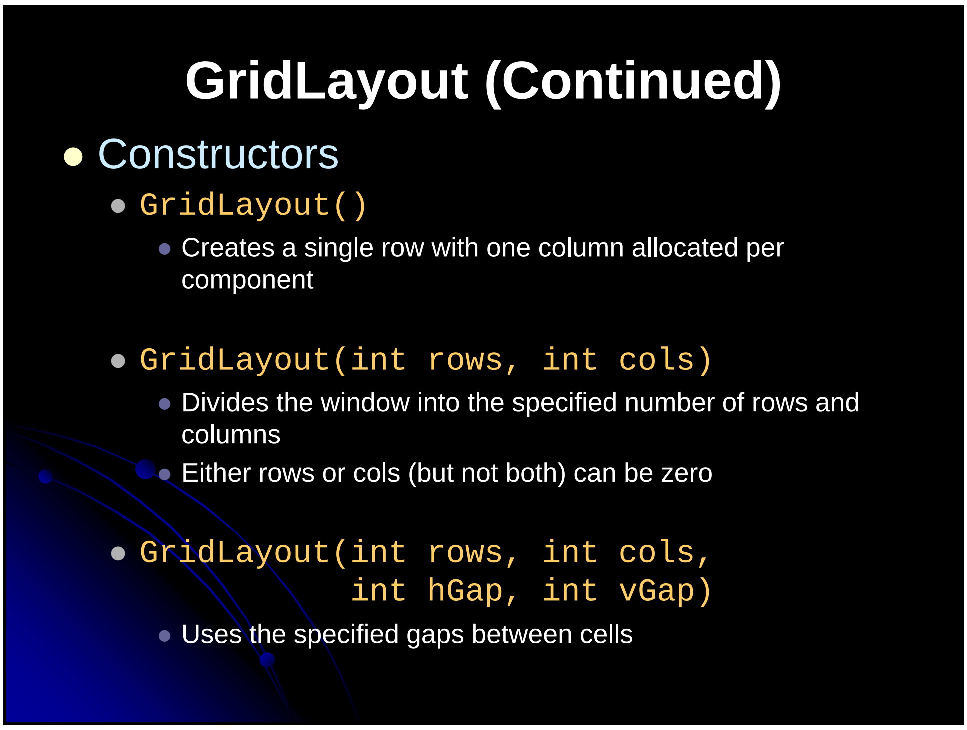
ShowBorderLayout

Run



GridLayout

- Behavior
  - Divides window into *equal-sized rectangles* based upon the number of rows and columns specified
  - Items placed into cells left-to-right, top-to-bottom, based on the order added to the container
  - Ignores the preferred size of the component; each component is *resized to fit into its grid cell*
  - Too few components results in blank cells
  - Too many components results in extra columns



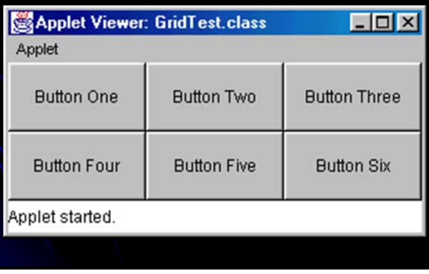
GridLayout (Continued)

- Constructors
  - `GridLayout()`
    - Creates a single row with one column allocated per component
  - `GridLayout(int rows, int cols)`
    - Divides the window into the specified number of rows and columns
    - Either rows or cols (but not both) can be zero

  -
    ```
    GridLayout(int rows, int cols,
               int hGap, int vGap)
    ```
    - Uses the specified gaps between cells

GridLayout, Example

```
public class GridTest extends Applet {
  public void init() {
    setLayout(new GridLayout(2,3)); // 2 rows, 3
  cols
    add(new Button("Button One"));
    add(new Button("Button Two"));
    add(new Button("Button Three"));
    add(new Button("Button Four"));
    add(new Button("Button Five"));
    add(new Button("Button Six"));
  }
}
```



Applet started.

Testing the `GridLayout` Manager

The `GridLayout` manager arranges components in a grid (matrix) formation with the number of rows and columns defined by the constructor. The components are placed in the grid from left to right starting with the first row, then the second, and so on.

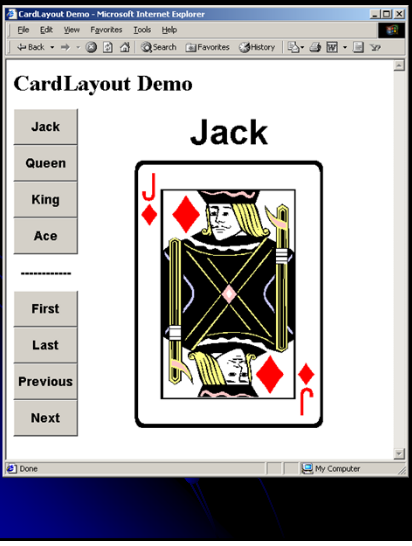
ShowGridLayout   Run

CardLayout

- Behavior
  - Stacks components on top of each other, displaying the top one
  - Associates a name with each component in window

```
Panel cardPanel;
CardLayout layout new CardLayout();
cardPanel.setLayout(layout);
...
cardPanel.add("Card 1", component1);
cardPanel.add("Card 2", component2);
...
layout.show(cardPanel, "Card 1");
layout.first(cardPanel);
layout.next(cardPanel);
```

CardLayout, Example


CardLayout Demo - Microsoft Internet Explorer

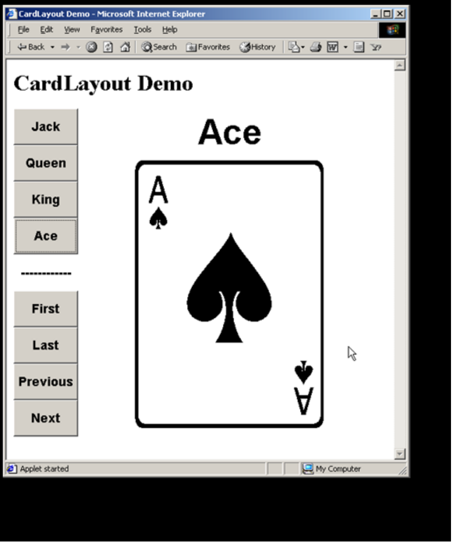
CardLayout Demo

Jack

Done

File Edit View Favorites Tools Help
Back Search Favorites History
Jack
Queen
King
Ace

First
Last
Previous
Next
My Computer


CardLayout Demo - Microsoft Internet Explorer

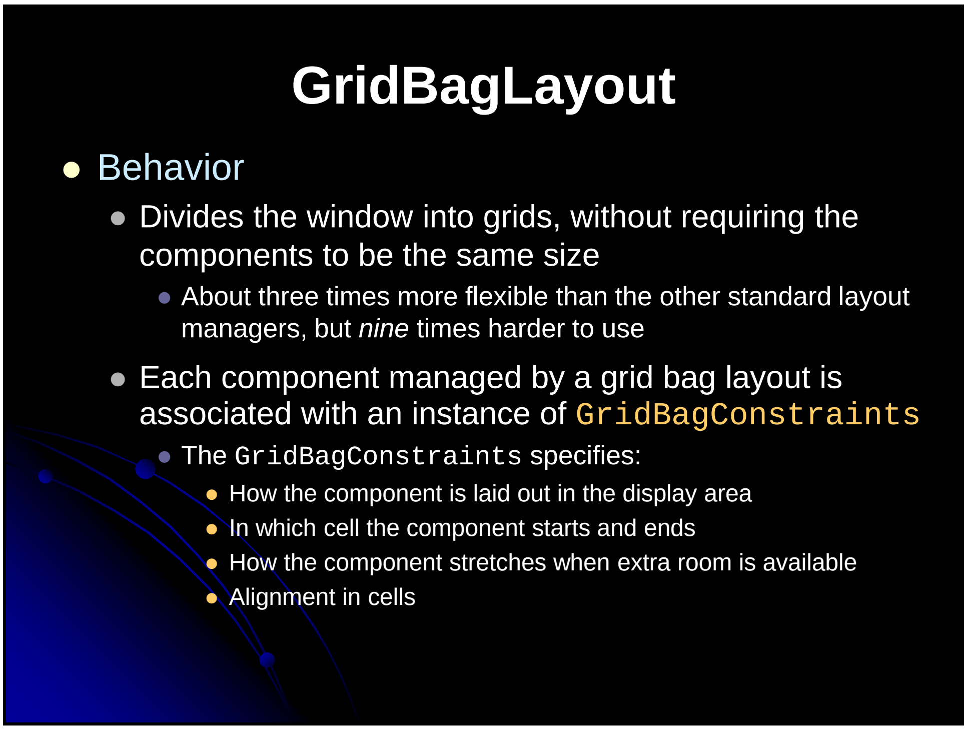
CardLayout Demo

Ace

Applet started

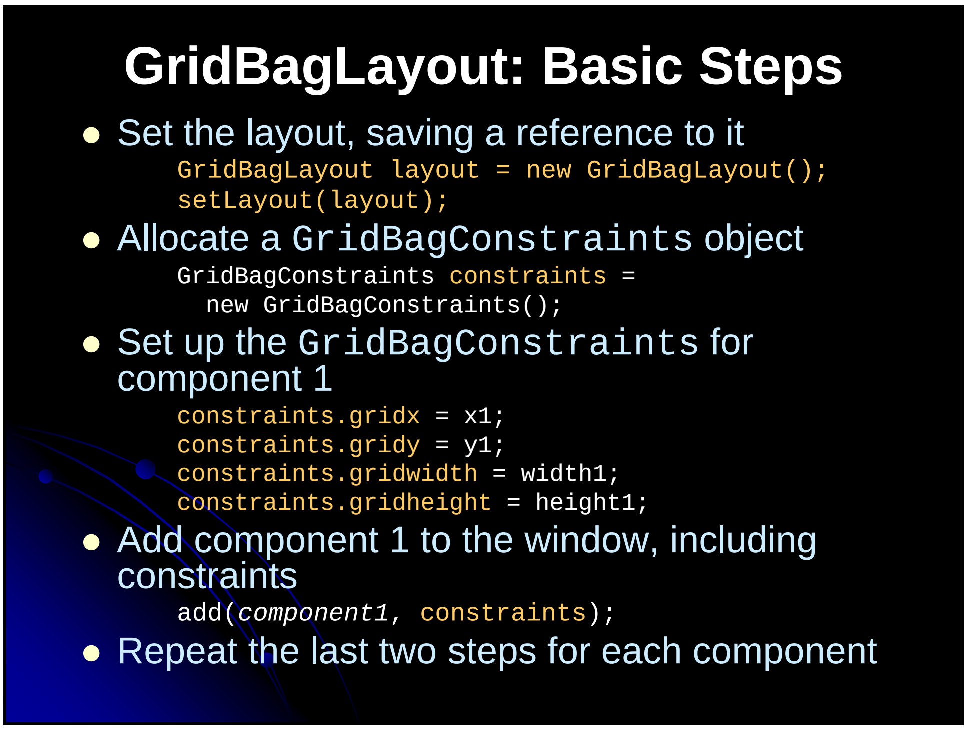
File Edit View Favorites Tools Help
Back Search Favorites History
Jack
Queen
King
Ace

First
Last
Previous
Next
My Computer



GridBagLayout

- Behavior
  - Divides the window into grids, without requiring the components to be the same size
    - About three times more flexible than the other standard layout managers, but *nine* times harder to use
  - Each component managed by a grid bag layout is associated with an instance of `GridBagConstraints`
    - The `GridBagConstraints` specifies:
      - How the component is laid out in the display area
      - In which cell the component starts and ends
      - How the component stretches when extra room is available
      - Alignment in cells



GridBagLayout: Basic Steps

- Set the layout, saving a reference to it

```
GridBagLayout layout = new GridBagLayout();
setLayout(layout);
```

- Allocate a `GridBagConstraints` object

```
GridBagConstraints constraints =
  new GridBagConstraints();
```

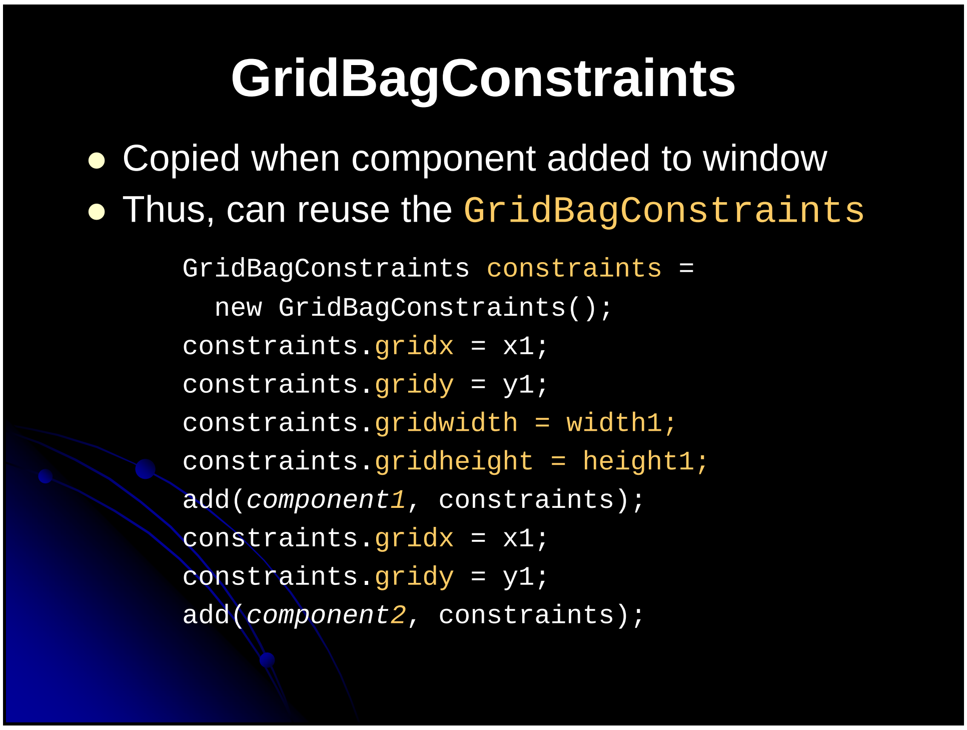
- Set up the `GridBagConstraints` for component 1

```
constraints.gridx = x1;
constraints.gridy = y1;
constraints.gridwidth = width1;
constraints.gridheight = height1;
```

- Add component 1 to the window, including constraints

```
add(component1, constraints);
```

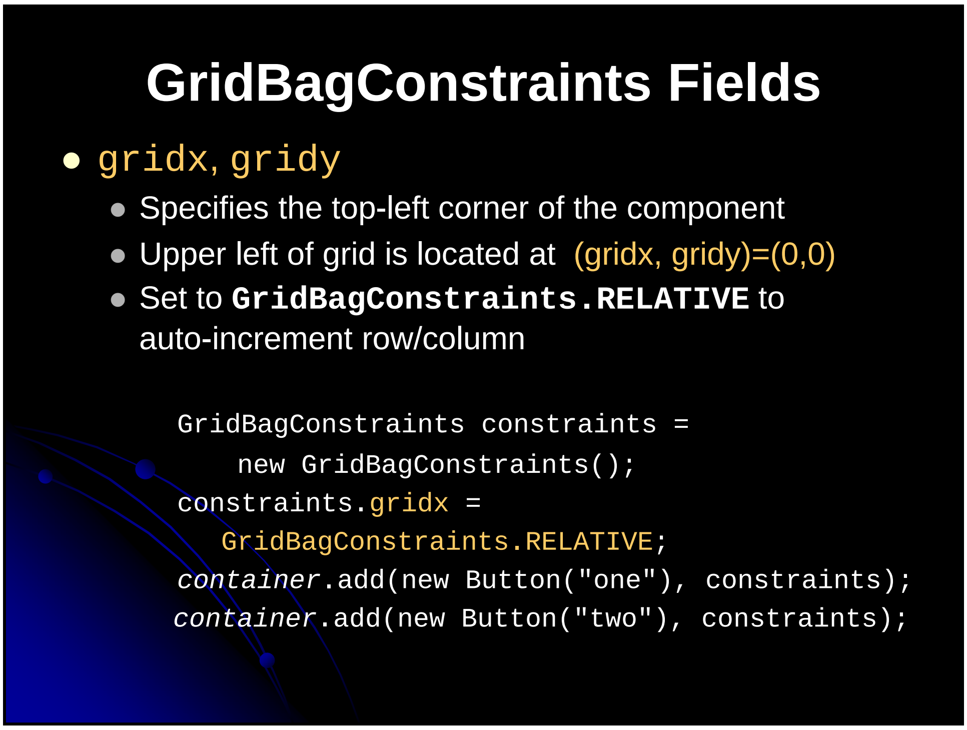
- Repeat the last two steps for each component



GridBagConstraints

- Copied when component added to window
- Thus, can reuse the `GridBagConstraints`

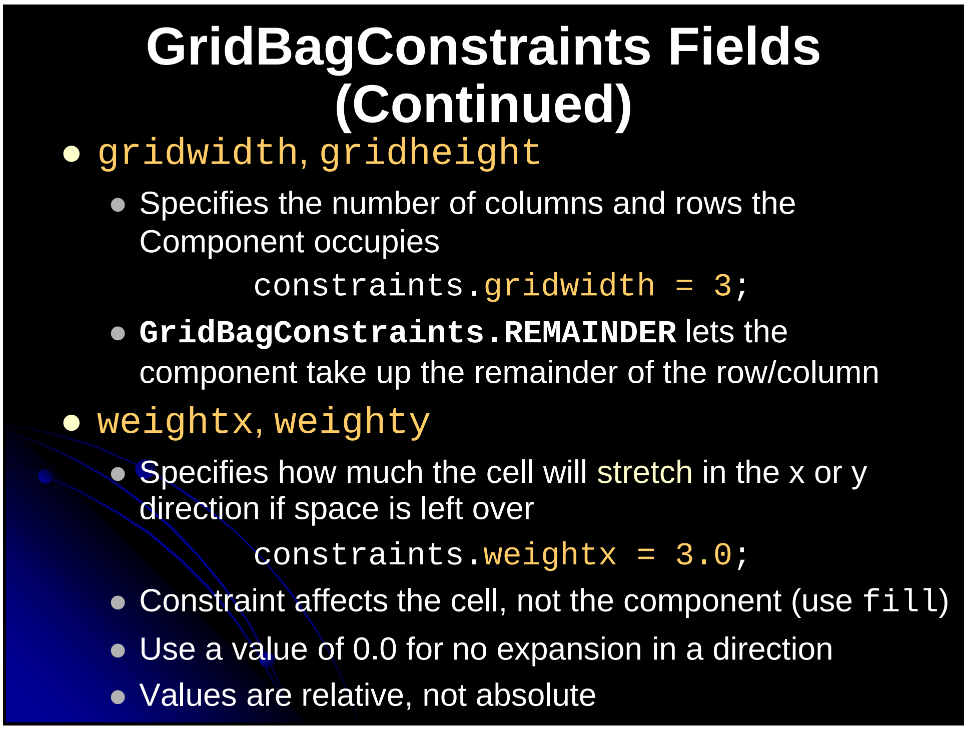
```
GridBagConstraints constraints =
  new GridBagConstraints();
constraints.gridx = x1;
constraints.gridy = y1;
constraints.gridwidth = width1;
constraints.gridheight = height1;
add(component1, constraints);
constraints.gridx = x1;
constraints.gridy = y1;
add(component2, constraints);
```



GridBagConstraints Fields

- `gridx`, `gridy`
  - Specifies the top-left corner of the component
  - Upper left of grid is located at (gridx, gridy)=(0,0)
  - Set to **`GridBagConstraints.RELATIVE`** to auto-increment row/column

```
GridBagConstraints constraints =
    new GridBagConstraints();
constraints.gridx =
   GridBagConstraints.RELATIVE;
container.add(new Button("one"), constraints);
container.add(new Button("two"), constraints);
```

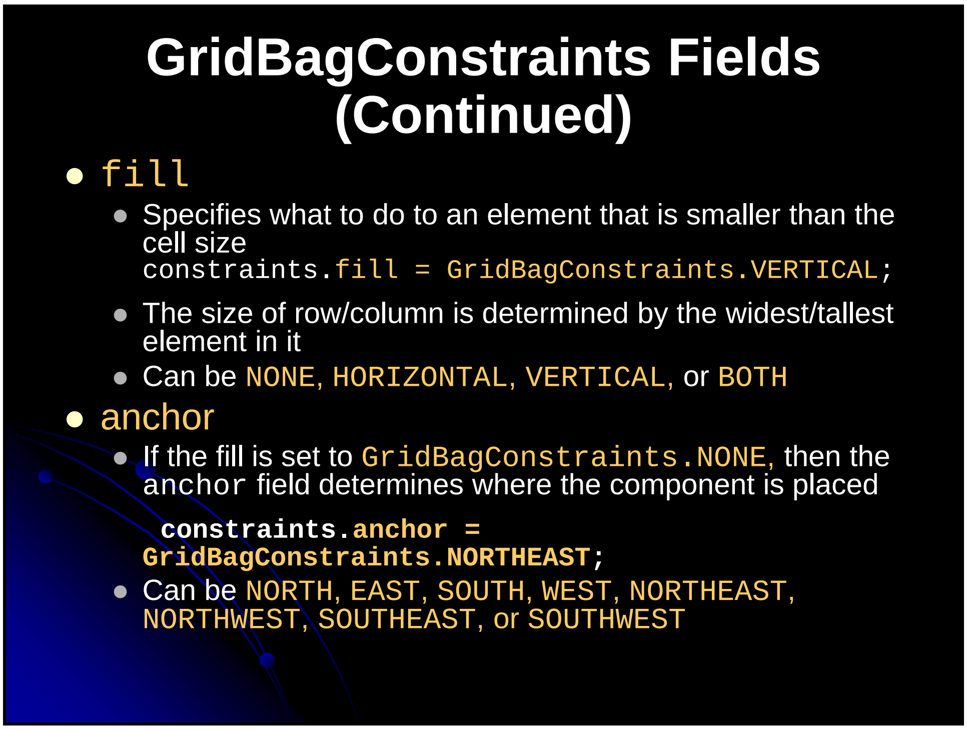


GridBagConstraints Fields (Continued)

- `gridwidth`, `gridheight`
  - Specifies the number of columns and rows the Component occupies

    ```
    constraints.gridwidth = 3;
    ```
  - **`GridBagConstraints.REMAINDER`** lets the component take up the remainder of the row/column
- `weightx`, `weighty`
  - Specifies how much the cell will stretch in the x or y direction if space is left over

    ```
    constraints.weightx = 3.0;
    ```
  - Constraint affects the cell, not the component (use `fill`)
  - Use a value of 0.0 for no expansion in a direction
  - Values are relative, not absolute



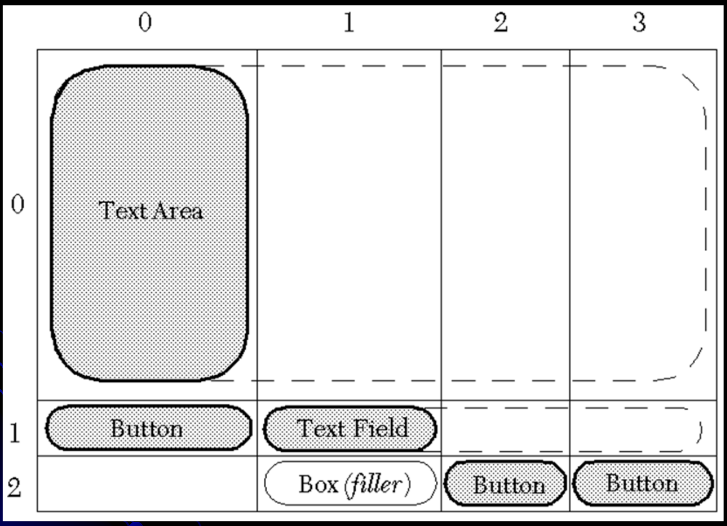
GridBagConstraints Fields (Continued)

- `fill`
  - Specifies what to do to an element that is smaller than the cell size

    ```
    constraints.fill = GridBagConstraints.VERTICAL;
    ```
  - The size of row/column is determined by the widest/tallest element in it
  - Can be `NONE`, `HORIZONTAL`, `VERTICAL`, or `BOTH`
- anchor
  - If the fill is set to `GridBagConstraints.NONE`, then the `anchor` field determines where the component is placed

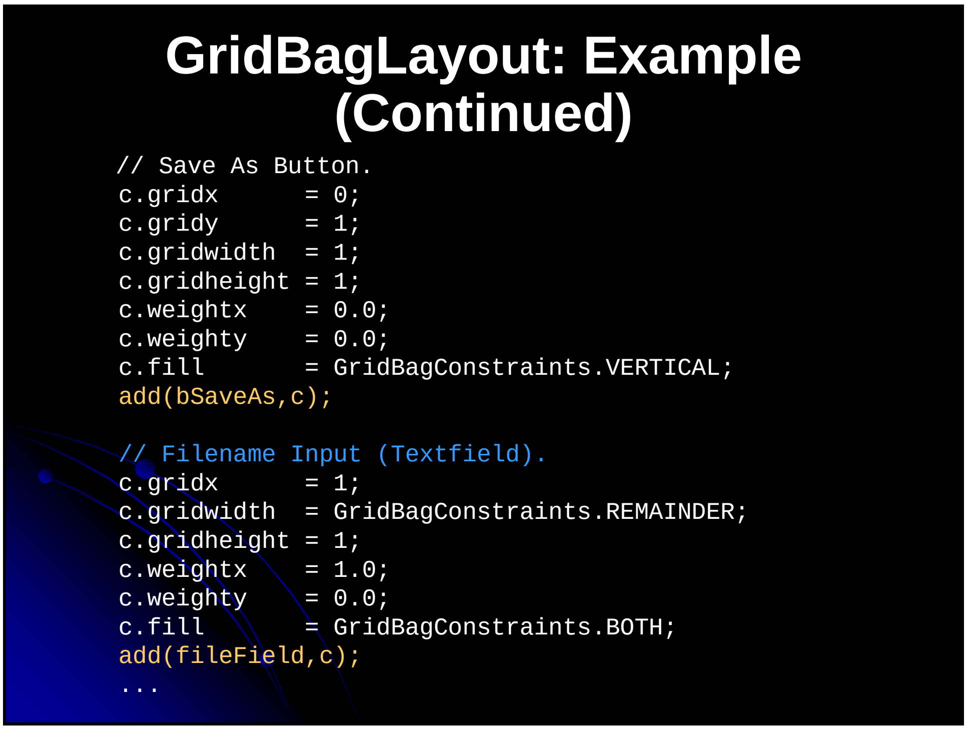
    ```
     constraints.anchor =
    GridBagConstraints.NORTHEAST;
    ```
  - Can be `NORTH`, `EAST`, `SOUTH`, `WEST`, `NORTHEAST`, `NORTHWEST`, `SOUTHEAST`, or `SOUTHWEST`

GridBagLayout: Example



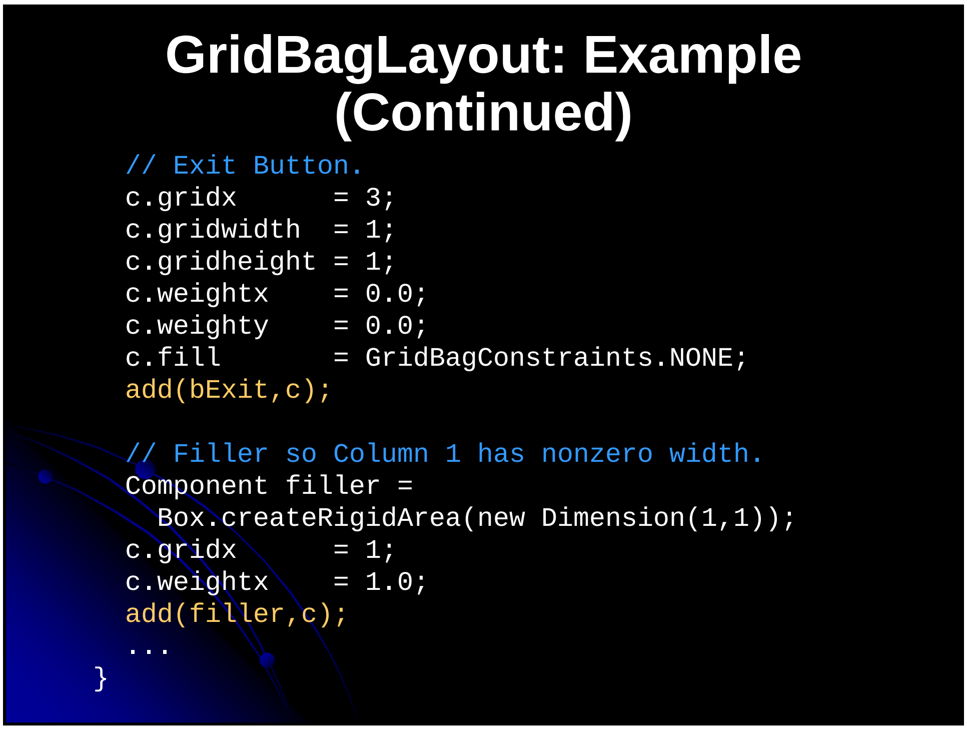
GridBagLayout: Example

```
public GridBagTest() {
   setLayout(new GridBagLayout());
   textArea = new JTextArea(12, 40);  // 12 rows, 40 cols
   bSaveAs = new JButton("Save As");
   fileField = new JTextField("C:\\Document.txt");
   bOk = new JButton("OK");
   bExit = new JButton("Exit");
   GridBagConstraints c = new GridBagConstraints();
   // Text Area.
   c.gridx      = 0;
   c.gridy      = 0;
   c.gridwidth  = GridBagConstraints.REMAINDER;
   c.gridheight = 1;
   c.weightx    = 1.0;
   c.weighty    = 1.0;
   c.fill       = GridBagConstraints.BOTH;
   c.insets     = new Insets(2,2,2,2); //t,l,b,r
   add(textArea, c);
   ...
```



GridBagLayout: Example (Continued)

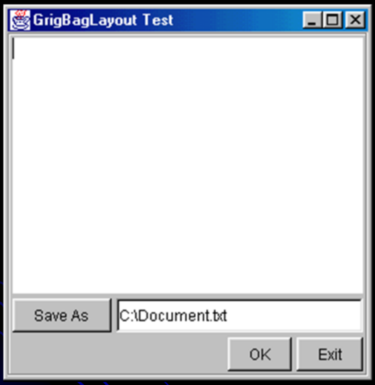
```
  // Save As Button.
  c.gridx      = 0;
  c.gridy      = 1;
  c.gridwidth  = 1;
  c.gridheight = 1;
  c.weightx    = 0.0;
  c.weighty    = 0.0;
  c.fill       = GridBagConstraints.VERTICAL;
  add(bSaveAs,c);

  // Filename Input (Textfield).
  c.gridx      = 1;
  c.gridwidth  = GridBagConstraints.REMAINDER;
  c.gridheight = 1;
  c.weightx    = 1.0;
  c.weighty    = 0.0;
  c.fill       = GridBagConstraints.BOTH;
  add(fileField,c);
  ...
```



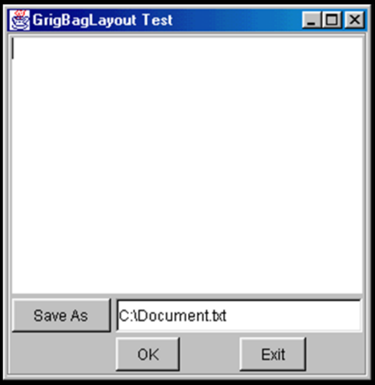
GridBagLayout: Example (Continued)

```
  // Exit Button.
  c.gridx      = 3;
  c.gridwidth  = 1;
  c.gridheight = 1;
  c.weightx    = 0.0;
  c.weighty    = 0.0;
  c.fill       = GridBagConstraints.NONE;
  add(bExit,c);

  // Filler so Column 1 has nonzero width.
  Component filler =
    Box.createRigidArea(new Dimension(1,1));
  c.gridx      = 1;
  c.weightx    = 1.0;
  add(filler,c);
  ...
}
```

GridBagLayout: Result



With Box filler at (2,1)



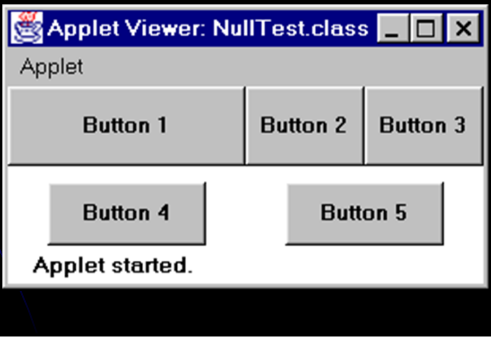
Without Box filler at (2,1)

Disabling the Layout Manager

- Behavior
  - If the layout is set to **`null`**, then components must be *sized* and *positioned* by hand
- Positioning components
    - *`component`*`.setSize(width, height)`
    - *`component`*`.setLocation(left, top)`
  - or
    - *`component`*`.setBounds(left, top, width, height)`

No Layout Manager: Example

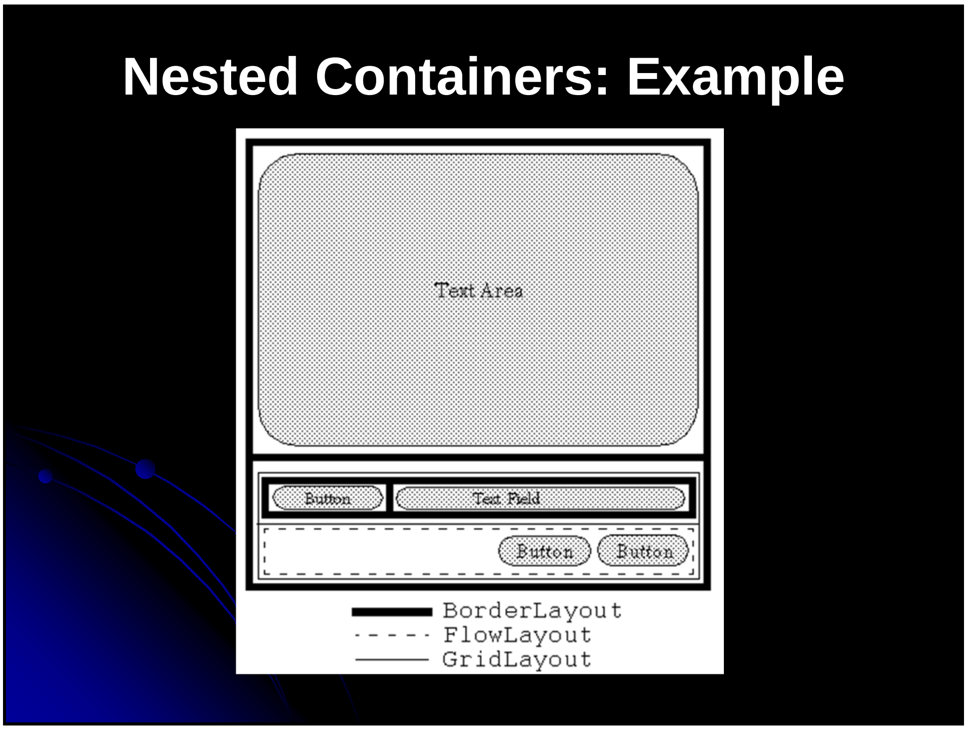
```
setLayout(null);
Button b1 = new Button("Button 1");
Button b2 = new Button("Button 2");
...
b1.setBounds(0, 0, 150, 50);
b2.setBounds(150, 0, 75, 50);
...
add(b1);
add(b2);
...
```



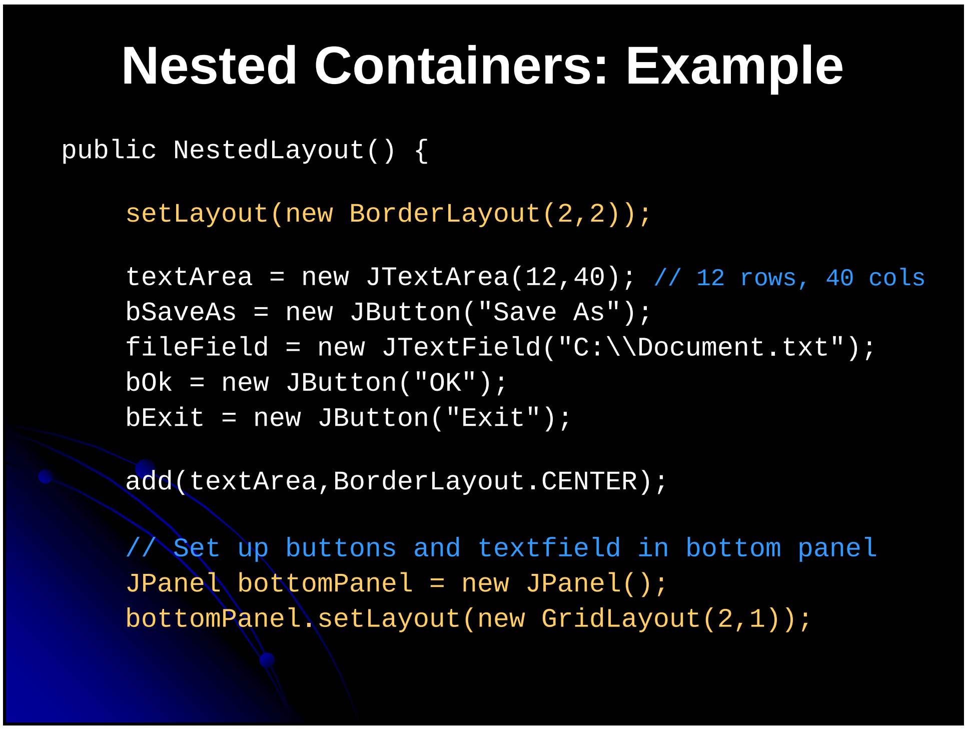
Applet started.

Using Layout Managers Effectively

- Use nested containers
  - Rather than struggling to fit your design in a single layout, try dividing the design into sections
  - Let each section be a panel with its own layout manager
- Turn off the layout manager for *some* containers
- Adjust the empty space around components
  - Change the space allocated by the layout manager
  - Override `insets` in the `Container`
  - Use a `Canvas` or a `Box` as an invisible spacer



Nested Containers: Example



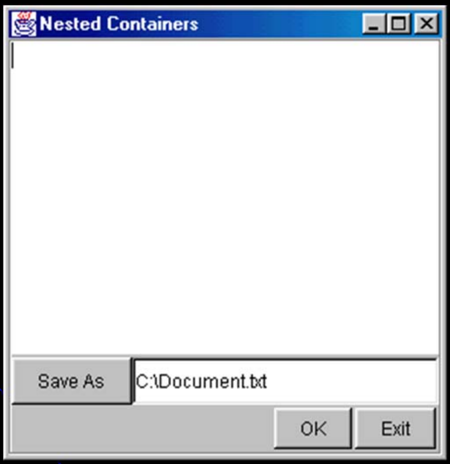
Nested Containers: Example

```
public NestedLayout() {

    setLayout(new BorderLayout(2,2));

    textArea = new JTextArea(12,40); // 12 rows, 40 cols
    bSaveAs = new JButton("Save As");
    fileField = new JTextField("C:\\Document.txt");
    bOk = new JButton("OK");
    bExit = new JButton("Exit");

    add(textArea,BorderLayout.CENTER);

    // Set up buttons and textfield in bottom panel
    JPanel bottomPanel = new JPanel();
    bottomPanel.setLayout(new GridLayout(2,1));
```

Nested Containers, Example

```
    JPanel subPanel1 = new JPanel();
    JPanel subPanel2 = new JPanel();
    subPanel1.setLayout(new BorderLayout());
    subPanel2.setLayout
              (new FlowLayout(FlowLayout.RIGHT,2,2));

    subPanel1.add(bSaveAs,BorderLayout.WEST);
    subPanel1.add(fileField,BorderLayout.CENTER);
    subPanel2.add(bOk);
    subPanel2.add(bExit);

    bottomPanel.add(subPanel1);
    bottomPanel.add(subPanel2);

    add(bottomPanel,BorderLayout.SOUTH);
  }
```

Nested Containers: Result



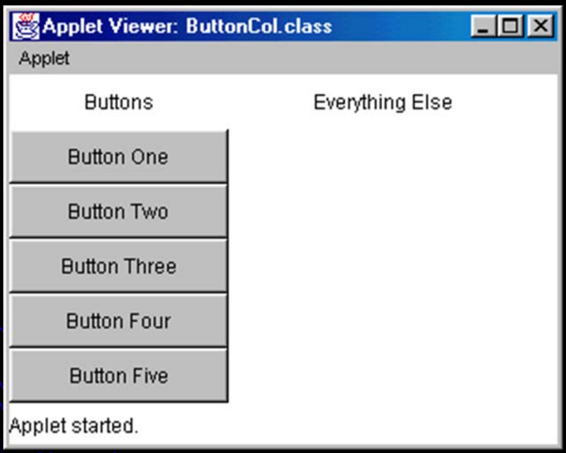
C:\Document.txt

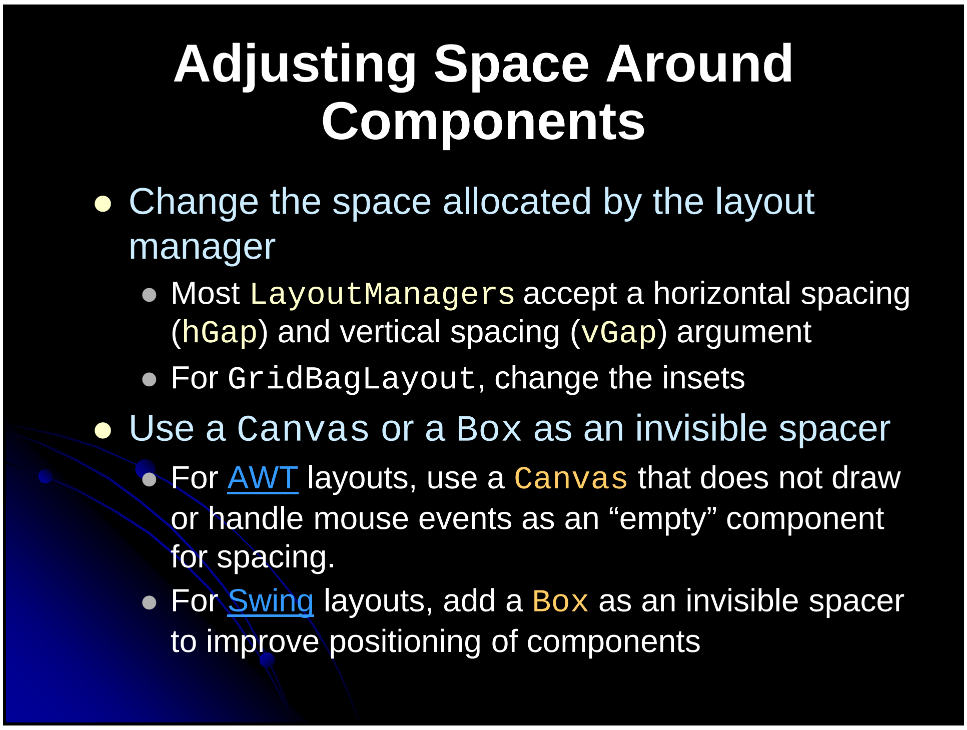
Turning Off Layout Manager for Some Containers: Example

- Suppose that you wanted to arrange a column of buttons (on the left) that take exactly 40% of the width of the container

```
setLayout(null);
int width1 = getSize().width*4/10;,
int height = getSize().height;
Panel buttonPanel = new Panel();
buttonPanel.setBounds(0, 0, width1, height);
buttonPanel.setLayout(new GridLayout(6, 1));
buttonPanel.add(new Label("Buttons", Label.CENTER));
buttonPanel.add(new Button("Button One"));
...
buttonPanel.add(new Button("Button Five"));
add(buttonPanel);
Panel everythingElse = new Panel();
int width2 = getSize().width - width1,
everythingElse.setBounds(width1+1, 0, width2, height);
```

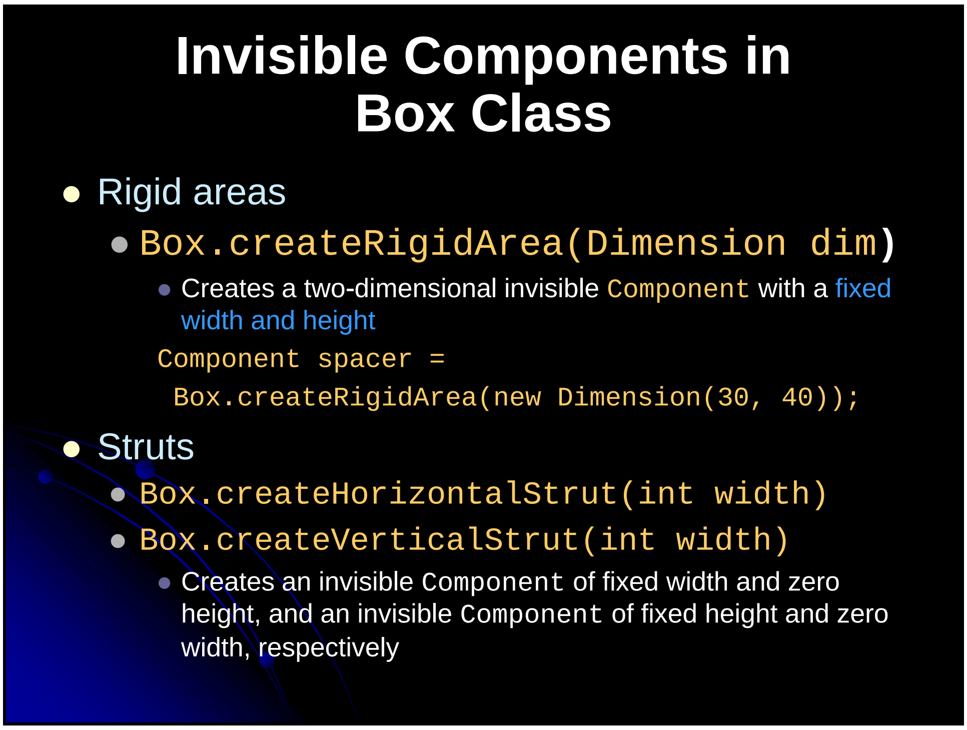
Turning Off Layout Manager for Some Containers: Result





Adjusting Space Around Components

- Change the space allocated by the layout manager
  - Most `LayoutManagers` accept a horizontal spacing (`hGap`) and vertical spacing (`vGap`) argument
  - For `GridBagLayout`, change the insets
- Use a `Canvas` or a `Box` as an invisible spacer
  - For AWT layouts, use a `Canvas` that does not draw or handle mouse events as an "empty" component for spacing.
  - For Swing layouts, add a `Box` as an invisible spacer to improve positioning of components



Invisible Components in Box Class

- Rigid areas
  - `Box.createRigidArea(Dimension dim)`
    - Creates a two-dimensional invisible `Component` with a fixed width and height

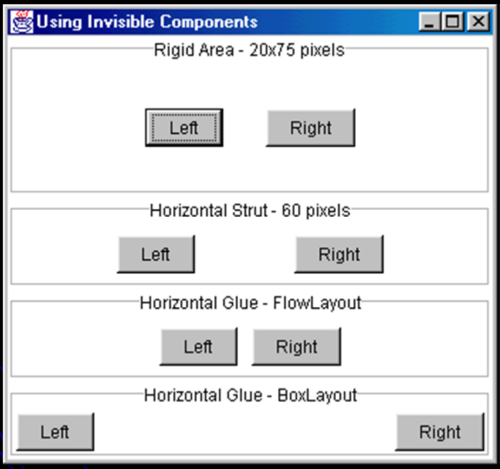
```
Component spacer =
 Box.createRigidArea(new Dimension(30, 40));
```

- Struts
  - `Box.createHorizontalStrut(int width)`
  - `Box.createVerticalStrut(int width)`
    - Creates an invisible `Component` of fixed width and zero height, and an invisible `Component` of fixed height and zero width, respectively

Invisible Components in Box Class (Continued)

- Glue
  - `Box.createHorizontalGlue()`
  - `Box.createVerticalGlue()`
    - Create an invisible `Component` that can expand horizontally or vertically, respectively, to fill all remaining space
  - `Box.createGlue()`
    - Creates a `Component` that can expand in both directions
    - A `Box` object achieves the glue effect by expressing a maximum size of `Short.MAX_VALUE`
    - Only apply `glue` to layout managers that respect the maximum size of a `Component`

Invisible Components: Example


Using Invisible Components
Rigid Area - 20x75 pixels
Left
Right
Horizontal Strut - 60 pixels
Left
Right
Horizontal Glue - FlowLayout
Left
Right
Horizontal Glue - BoxLayout
Left
Right

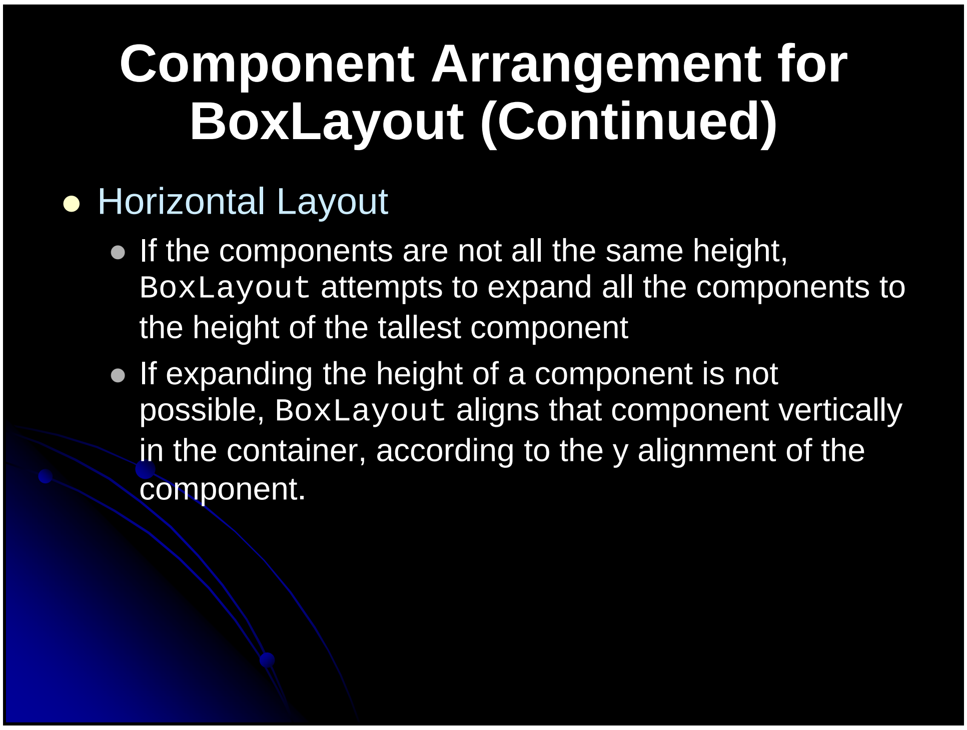
BoxLayout

- Behavior
  - Manager from Swing; available only in Java 2
  - Arranges `Components` either in a horizontal row, `BoxLayout.X_AXIS`, or in a vertical column, `BoxLayout.Y_AXIS`
  - Lays out the components in the order in which they were added to the `Container`
  - Resizing the container does not cause the components to relocate
  - Unlike the other standard layout managers, the `BoxLayout` manager cannot be shared with more than one `Container`

```
BoxLayout layout =
    new BoxLayout(container,BoxLayout.X_AXIS);
```

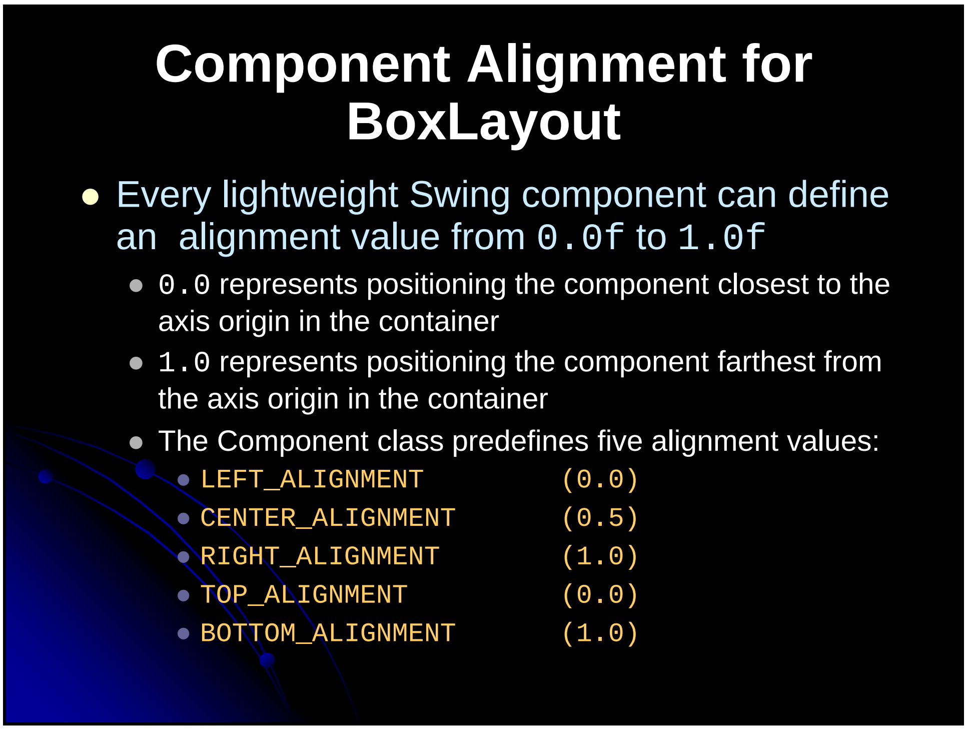
Component Arrangement for BoxLayout

- Attempts to arrange the components with:
  - Their preferred widths (vertical layout), or
  - Their preferred heights (horizontal layout)
- Vertical Layout
  - If the components are not all the same width, `BoxLayout` attempts to expand all the components to the width of the component with the largest preferred width
  - If expanding a component is not possible (restricted maximum size), `BoxLayout` aligns that component horizontally in the container, according to the x alignment of the component



Component Arrangement for BoxLayout (Continued)

- Horizontal Layout
  - If the components are not all the same height, `BoxLayout` attempts to expand all the components to the height of the tallest component
  - If expanding the height of a component is not possible, `BoxLayout` aligns that component vertically in the container, according to the y alignment of the component.



Component Alignment for BoxLayout

- Every lightweight Swing component can define an alignment value from `0.0f` to `1.0f`
  - `0.0` represents positioning the component closest to the axis origin in the container
  - `1.0` represents positioning the component farthest from the axis origin in the container
  - The Component class predefines five alignment values:

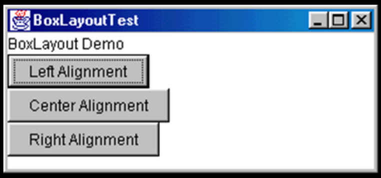
`LEFT_ALIGNMENT`	`(0.0)`
`CENTER_ALIGNMENT`	`(0.5)`
`RIGHT_ALIGNMENT`	`(1.0)`
`TOP_ALIGNMENT`	`(0.0)`
`BOTTOM_ALIGNMENT`	`(1.0)`

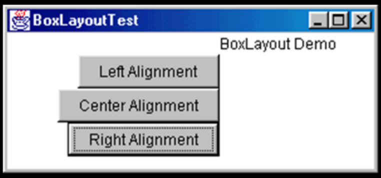
Component Alignment for BoxLayout (Continued)

- Most Swing components have a default x-axis alignment of center
  - Exceptions include `JButton`, `JComboBox`, `JLabel`, and `JMenu`, which have x-axis alignment of **`left`**
- Set the `Component` alignment

```
component.setAlignmentX(Component.Xxx_ALIGNMENT)
component.setAlignmentY(Component.Xxx_ALIGNMENT)
```

BoxLayout: Example





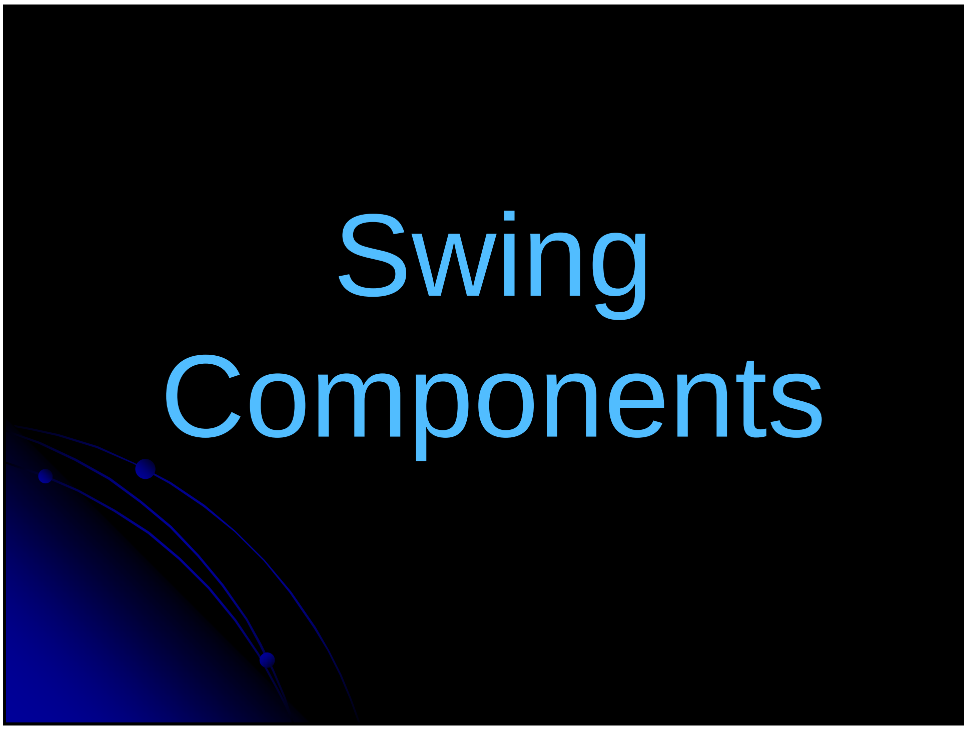
- All components have a 0.0 (left) alignment

- The label has a 0.0 alignment
- The buttons have a 1.0 (right) alignment

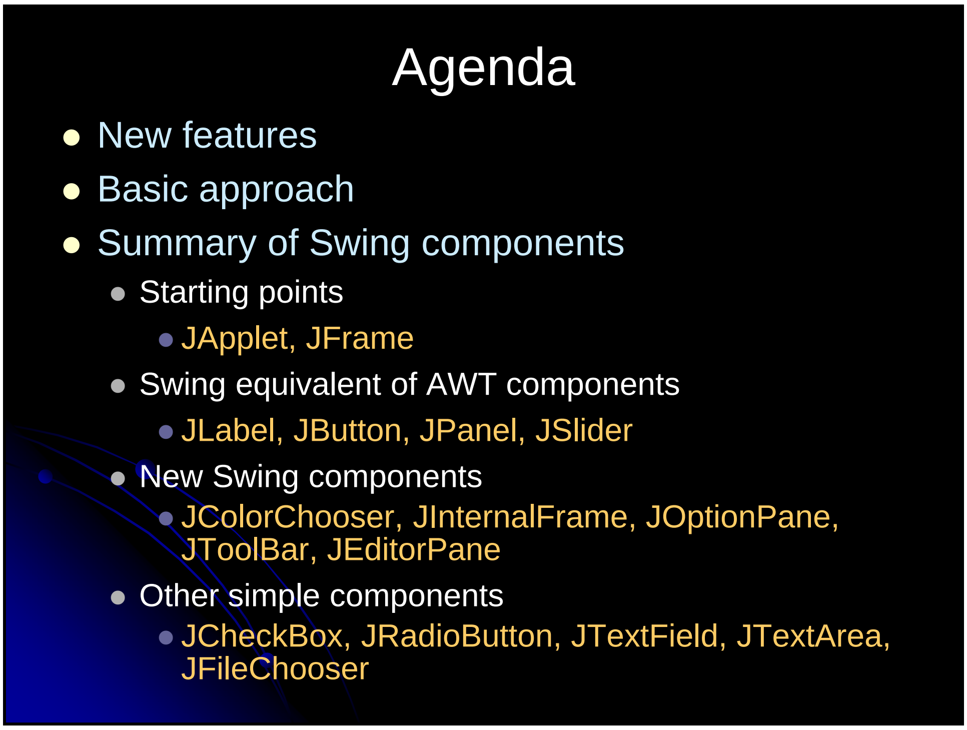
Summary

- Default layout managers
  - Applet and Panel: FlowLayout
  - Frame and Dialog: BorderLayout
- Layout managers respect the preferred size of the component differently
- GridBagLayout is the most complicated but most flexible manager
  - Use `GridBagConstraints` to specify the layout of each component
- Complex layouts can often be simplified through nested containers
- In AWT use a Canvas as a spacer; in Swing use a Box as a spacer

Thank you for your attention!



Swing Components



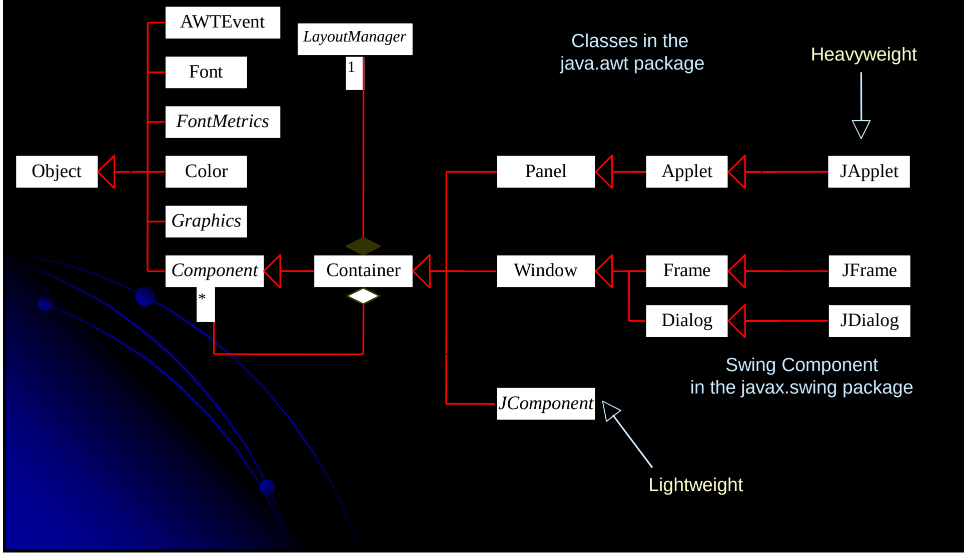
Agenda

- New features
- Basic approach
- Summary of Swing components
  - Starting points
    - JApplet, JFrame
  - Swing equivalent of AWT components
    - JLabel, JButton, JPanel, JSlider
  - New Swing components
    - JColorChooser, JInternalFrame, JOptionPane, JToolBar, JEditorPane
  - Other simple components
    - JCheckBox, JRadioButton, JTextField, JTextArea, JFileChooser

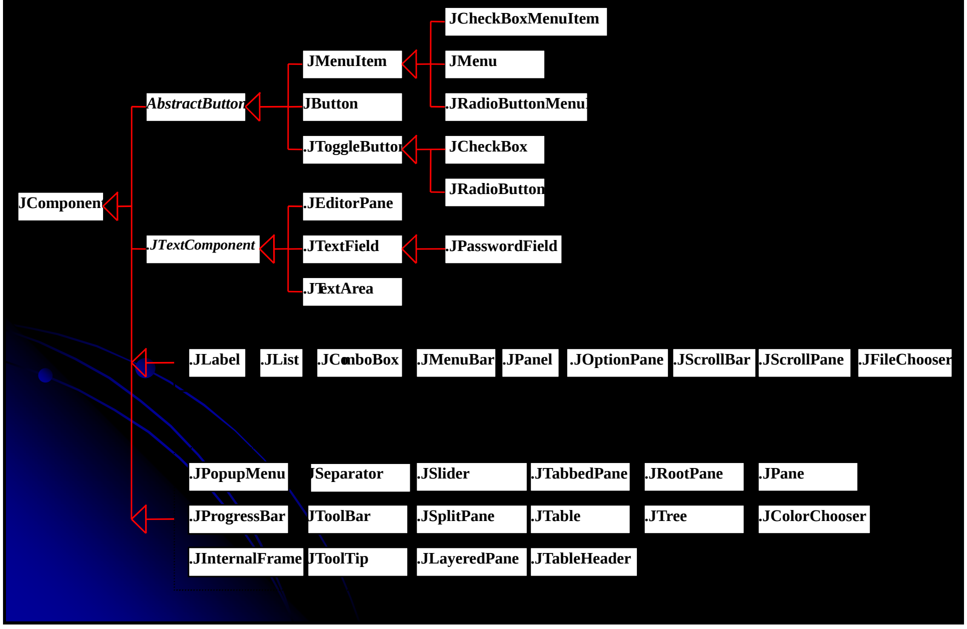
New Features

- Many more built-in controls
  - Image buttons, tabbed panes, sliders, toolbars, color choosers, HTML text areas, lists, trees, and tables.
- Increased customization of components
  - Border styles, text alignments, and basic drawing features. Images can be added to almost any control.
- A pluggable “look and feel”
  - Not limited to “native” look.
- Many miscellaneous small features
  - Built-in double buffering, tool-tips, dockable toolbars, keyboard accelerators, custom cursors, etc.
- Model-view-controller architecture
  - Can change internal representation of trees, lists, tables.

Graphics Class Hierarchy (Swing)



JComponent


JCheckBoxMenuItem

JMenuItem

JMenu

AbstractButton

JButton

.JRadioButtonMenu

.JToggleButton

JCheckBox

JRadioButton

JComponen

.JEditorPane

.*JTextComponent*

.JTextField

.JPasswordField

.JTextArea

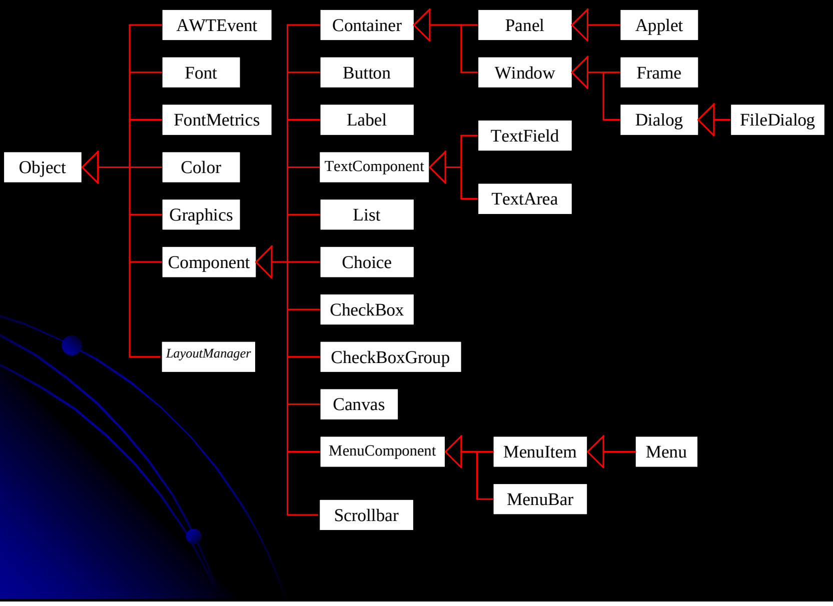
.JLabel .JList .JComboBox .JMenuBar .JPanel .JOptionPane .JScrollBar .JScrollPane .JFileChooser

.JPopupMenu JSeparator .JSlider .JTabbedPane .JRootPane .JPane

.JProgressBar JToolBar .JSplitPane .JTable .JTree .JColorChooser

.JInternalFrame JToolTip .JLayeredPane .JTableHeader

AWT (Abstract Window Toolkit)


Object

AWTEvent

Font

FontMetrics

Color

Graphics

Component

LayoutManager

Container

Button

Label

TextComponent

List

Choice

CheckBox

CheckBoxGroup

Canvas

MenuComponent

Scrollbar

Panel

Window

TextField

TextArea

MenuItem

MenuBar

Applet

Frame

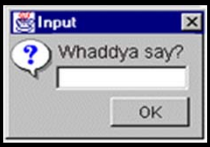
Dialog

FileDialog

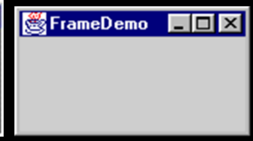
Menu

Basic Components

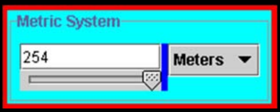
Applet



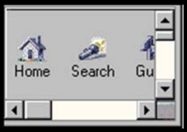
Dialog



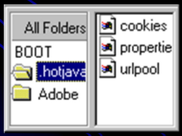
Frame



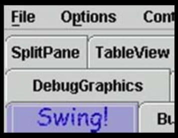
Panel



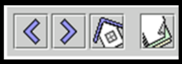
Scroll panel



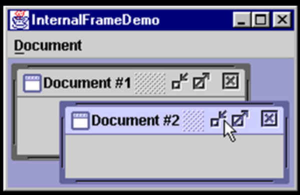
Split pane



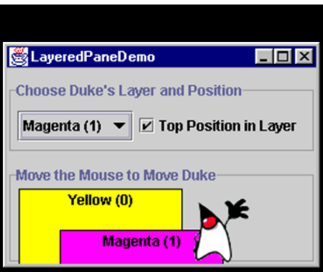
Tabbed pane



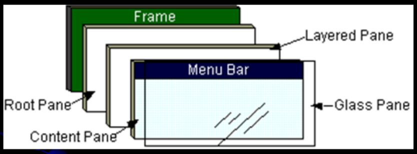
Tool bar



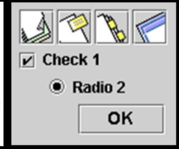
Internal frame



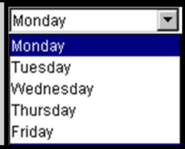
Layered pane



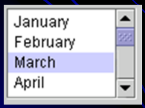
Root pane



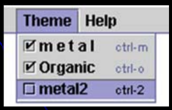
Buttons



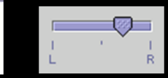
Combo box



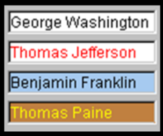
List



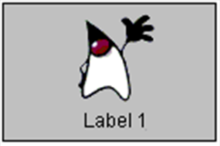
Menu



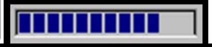
Slider



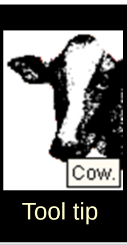
Text fields



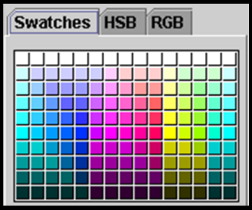
Label



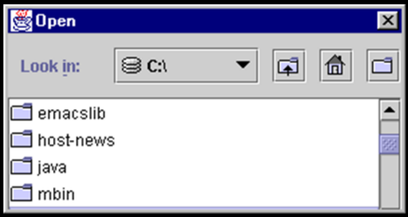
Progress bar



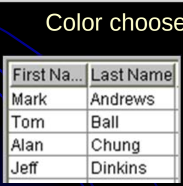
Tool tip



Color chooser

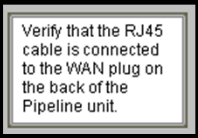


File chooser



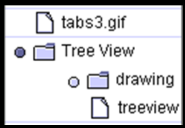
First Na...	Last Name
Mark	Andrews
Tom	Ball
Alan	Chung
Jeff	Dinkins

Table



Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

Text

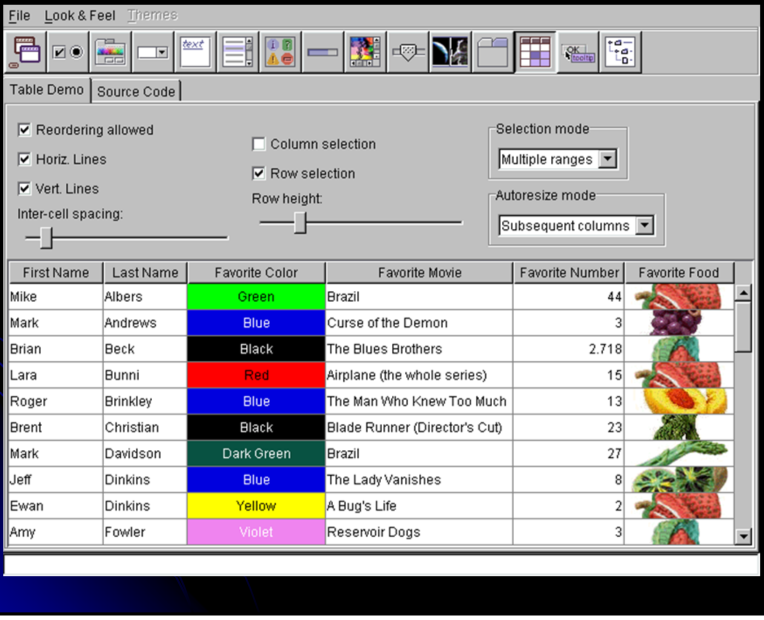


Tree

Swing vs. AWT Programming

- Naming convention
  - All Swing component names begin with a capital J and follow the format J*Xxx*. E.g., JFrame, JPanel, JApplet, JDialog, JButton. Many are just AWT names with a J.
- Lightweight components
  - Most Swing components are *lightweight*: formed by drawing in the underlying window.
- Use of paintComponent for drawing
  - Custom drawing code is in paintComponent, not paint. Double buffering turned on by default.
- New Look and Feel as default
  - With Swing, you have to explicitly set the native look.
- Don't mix Swing and AWT in same window

Windows Look and Feel



☑ Reordering allowed

☑ Horiz. Lines

☑ Vert. Lines

Inter-cell spacing:

□ Column selection

☑ Row selection

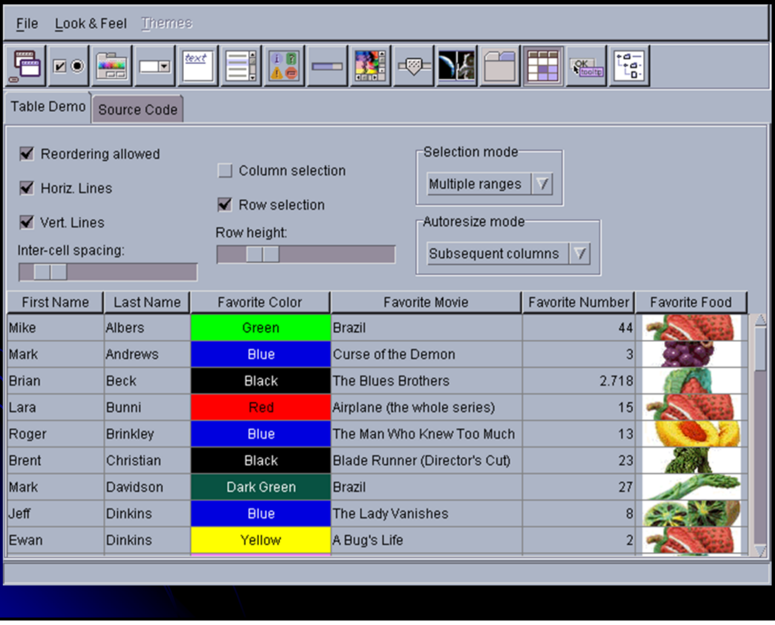
Row height:

Selection mode: Multiple ranges

Autoresize mode: Subsequent columns

First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the Demon	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the whole series)	15	
Roger	Brinkley	Blue	The Man Who Knew Too Much	13	
Brent	Christian	Black	Blade Runner (Director's Cut)	23	
Mark	Davidson	Dark Green	Brazil	27	
Jeff	Dinkins	Blue	The Lady Vanishes	8	
Ewan	Dinkins	Yellow	A Bug's Life	2	
Amy	Fowler	Violet	Reservoir Dogs	3	

Motif Look and Feel



☑ Reordering allowed

☑ Horiz. Lines

☑ Vert. Lines

Inter-cell spacing:

□ Column selection

☑ Row selection

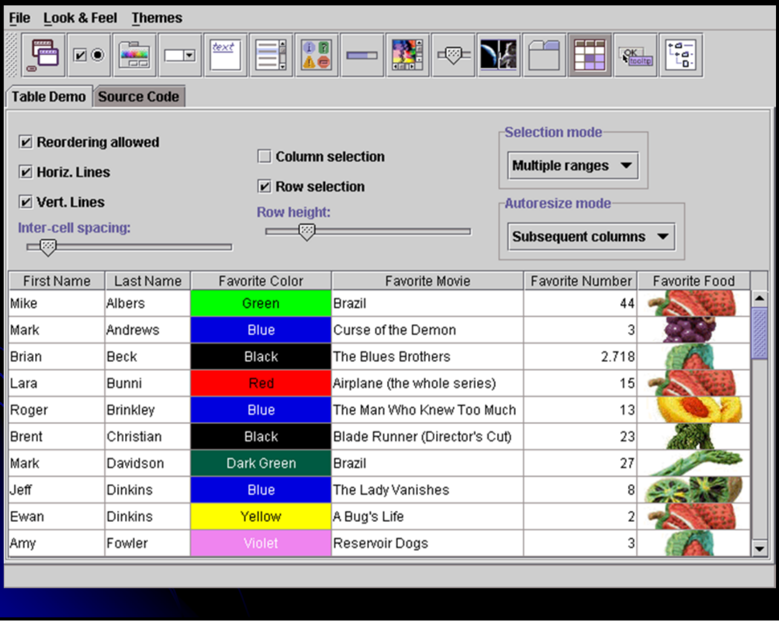
Row height:

Selection mode: Multiple ranges

Autoresize mode: Subsequent columns

First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the Demon	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the whole series)	15	
Roger	Brinkley	Blue	The Man Who Knew Too Much	13	
Brent	Christian	Black	Blade Runner (Director's Cut)	23	
Mark	Davidson	Dark Green	Brazil	27	
Jeff	Dinkins	Blue	The Lady Vanishes	8	
Ewan	Dinkins	Yellow	A Bug's Life	2	

Java Look and Feel



☑ Reordering allowed

☑ Horiz. Lines

☑ Vert. Lines

Inter-cell spacing:

□ Column selection

☑ Row selection

Row height:

Selection mode: Multiple ranges

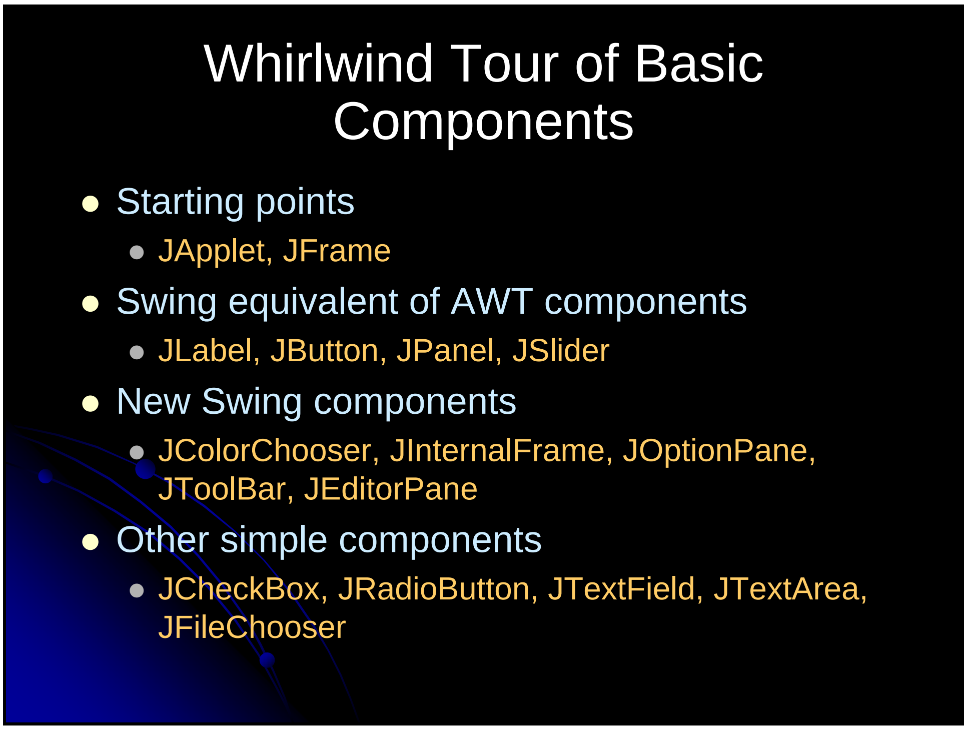
Autoresize mode: Subsequent columns

First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the Demon	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the whole series)	15	
Roger	Brinkley	Blue	The Man Who Knew Too Much	13	
Brent	Christian	Black	Blade Runner (Director's Cut)	23	
Mark	Davidson	Dark Green	Brazil	27	
Jeff	Dinkins	Blue	The Lady Vanishes	8	
Ewan	Dinkins	Yellow	A Bug's Life	2	
Amy	Fowler	Violet	Reservoir Dogs	3	

Setting Native Look and Feel

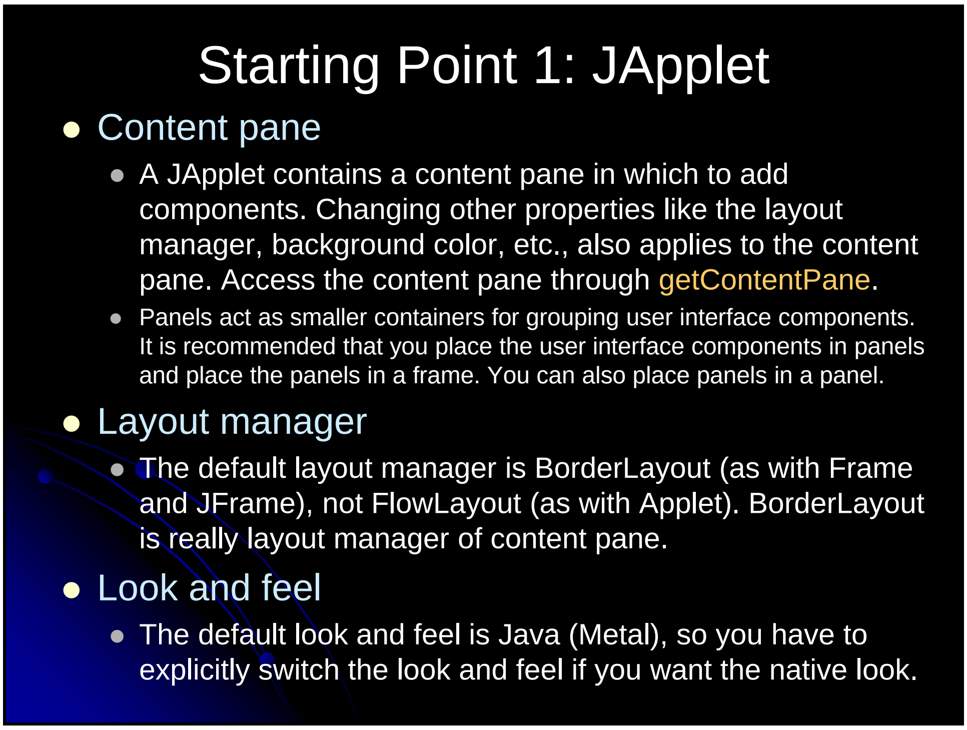
- Most applications should use native look, not default “Java” look
- Changing is tedious, so use static method

```
public class WindowUtilities {
  public static void setNativeLookAndFeel() {
    try {
     UIManager.setLookAndFeel(
       UIManager.getSystemLookAndFeelClassName());
    } catch(Exception e) {
      System.out.println("Error setting native LAF: " + e);
    }
  }
  ...
```



Whirlwind Tour of Basic Components

- Starting points
  - JApplet, JFrame
- Swing equivalent of AWT components
  - JLabel, JButton, JPanel, JSlider
- New Swing components
  - JColorChooser, JInternalFrame, JOptionPane, JToolBar, JEditorPane
- Other simple components
  - JCheckBox, JRadioButton, JTextField, JTextArea, JFileChooser



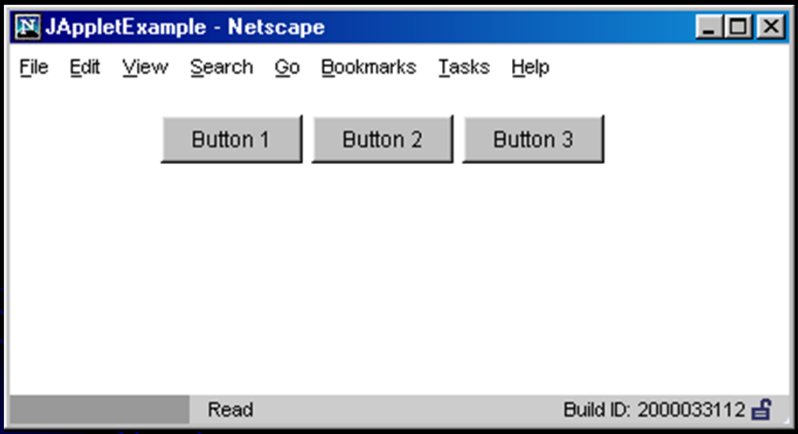
Starting Point 1: JApplet

- Content pane
  - A JApplet contains a content pane in which to add components. Changing other properties like the layout manager, background color, etc., also applies to the content pane. Access the content pane through getContentPane.
  - Panels act as smaller containers for grouping user interface components. It is recommended that you place the user interface components in panels and place the panels in a frame. You can also place panels in a panel.
- Layout manager
  - The default layout manager is BorderLayout (as with Frame and JFrame), not FlowLayout (as with Applet). BorderLayout is really layout manager of content pane.
- Look and feel
  - The default look and feel is Java (Metal), so you have to explicitly switch the look and feel if you want the native look.

JApplet: Example Code

```
import java.awt.*;
import javax.swing.*;

public class JAppletExample extends JApplet {
  public void init() {
    WindowUtilities.setNativeLookAndFeel();
    Container content = getContentPane();
    content.setBackground(Color.white);
    content.setLayout(new FlowLayout());
    content.add(new JButton("Button 1"));
    content.add(new JButton("Button 2"));
    content.add(new JButton("Button 3"));
  }
}
```

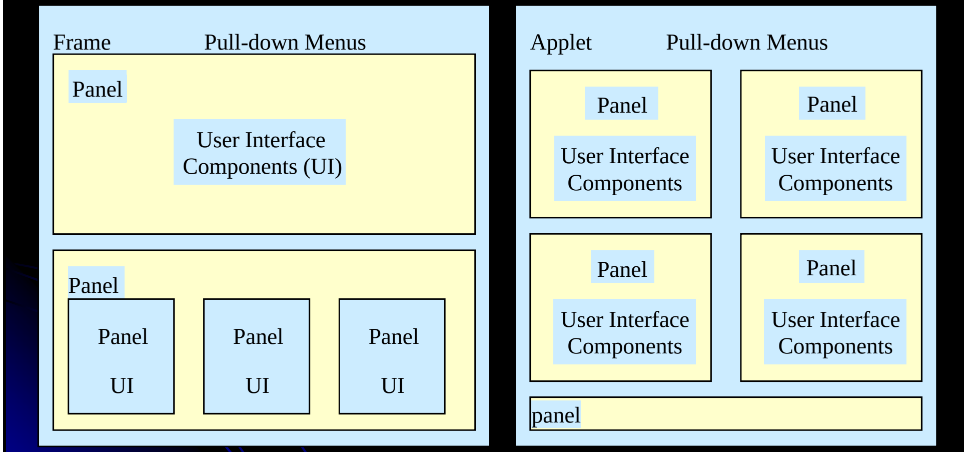
JApplet: Example Output

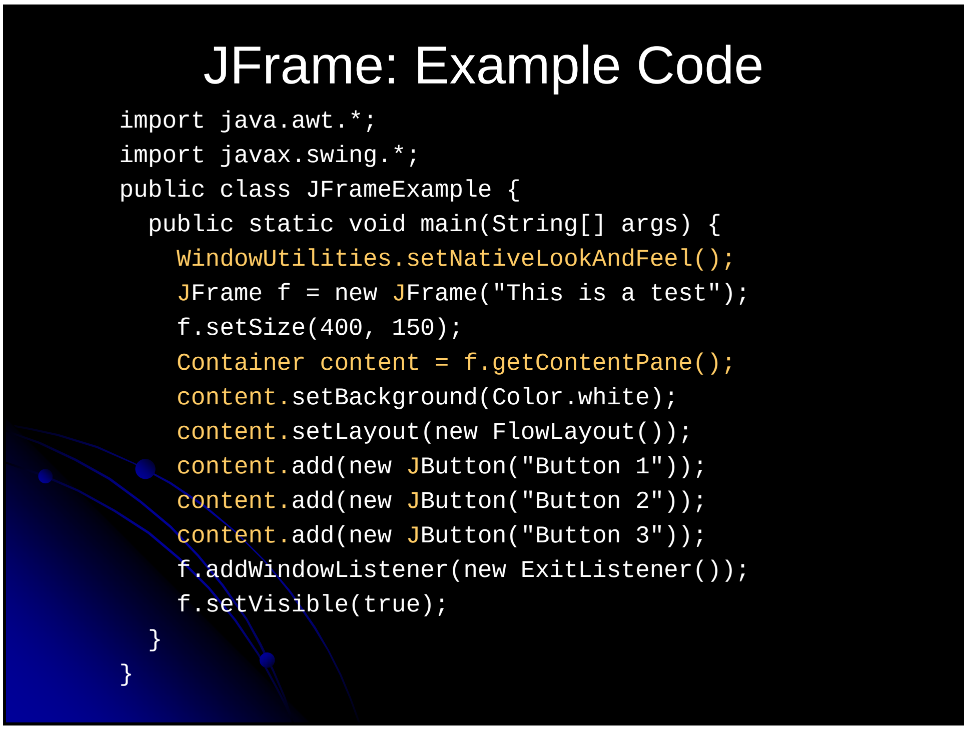


Starting Point 2: JFrame

- Content pane
  - JFrame uses content pane in same way as does JApplet.
  - Frame is a window that is not contained inside another window. Frame is the basis to contain other user interface components in Java graphical applications. The Frame class can be used to create windows.
- Auto-close behavior
  - JFrames close automatically when you click on the Close button (unlike AWT Frames). However, closing the last JFrame does not result in your program exiting the Java application. So, your "main" JFrame still needs a WindowListener to call System.exit. Or, alternatively, if using JDK 1.3 or later, you can call setDefault-CloseOperation(EXIT_ON_CLOSE). This permits the JFrame to close; however, you won't be able to complete any house cleaning as you might in the WindowListener.
- Look and feel
  - The default look and feel is Java (Metal)

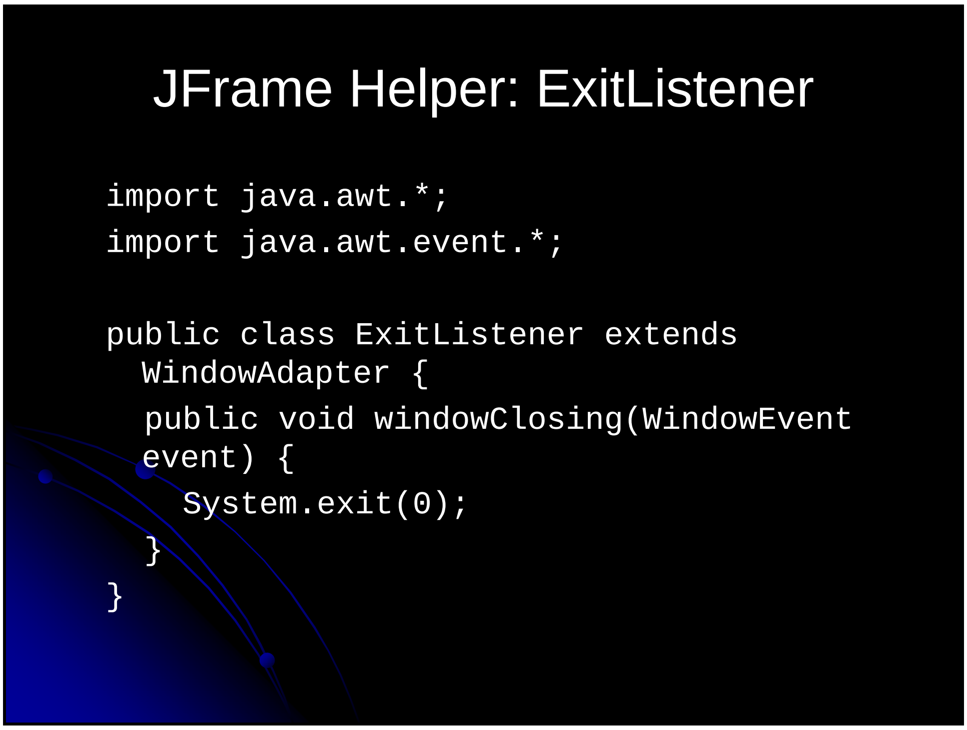
JFrames


Frame
Pull-down Menus
Panel
User Interface Components (UI)
Panel
Panel
UI
Panel
UI
Panel
UI
Applet
Pull-down Menus
Panel
User Interface Components
Panel
User Interface Components
Panel
User Interface Components
Panel
User Interface Components
panel



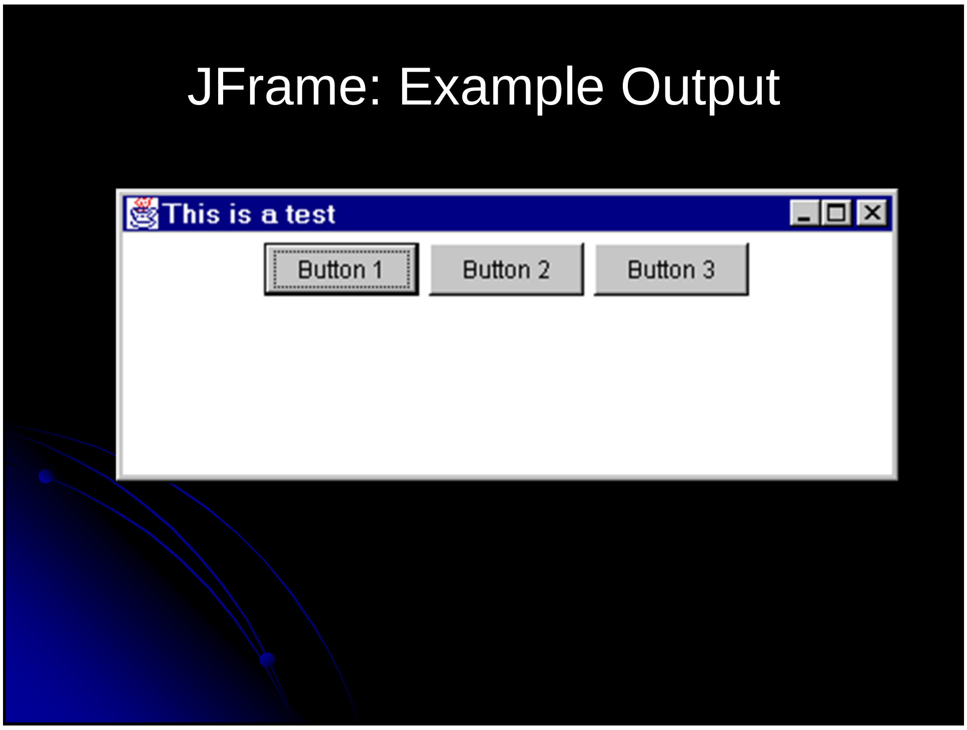
JFrame: Example Code

```
import java.awt.*;
import javax.swing.*;
public class JFrameExample {
  public static void main(String[] args) {
    WindowUtilities.setNativeLookAndFeel();
    JFrame f = new JFrame("This is a test");
    f.setSize(400, 150);
    Container content = f.getContentPane();
    content.setBackground(Color.white);
    content.setLayout(new FlowLayout());
    content.add(new JButton("Button 1"));
    content.add(new JButton("Button 2"));
    content.add(new JButton("Button 3"));
    f.addWindowListener(new ExitListener());
    f.setVisible(true);
  }
}
```

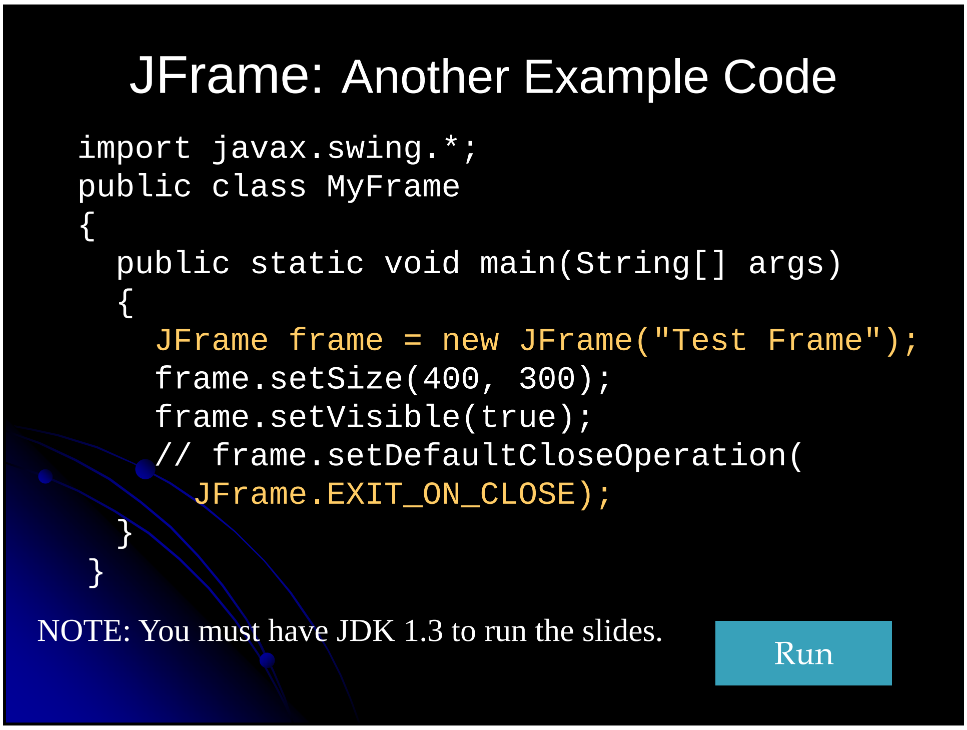


JFrame Helper: ExitListener

```
import java.awt.*;
import java.awt.event.*;

public class ExitListener extends
  WindowAdapter {
  public void windowClosing(WindowEvent
  event) {
    System.exit(0);
  }
}
```



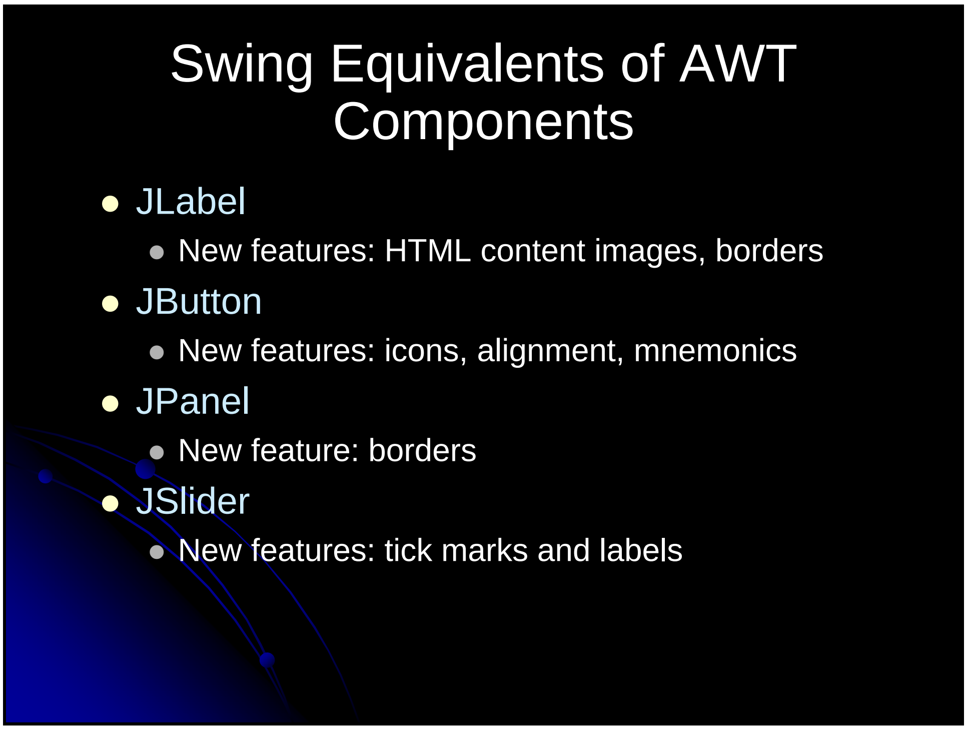
JFrame: Example Output



JFrame: Another Example Code

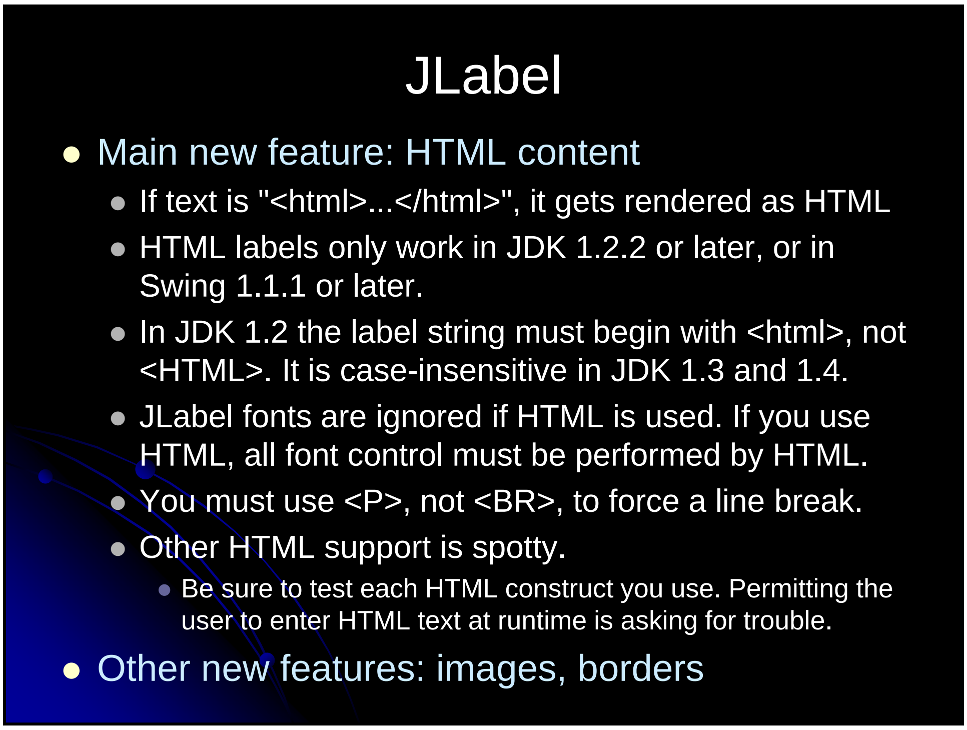
```
import javax.swing.*;
public class MyFrame
{
  public static void main(String[] args)
  {
    JFrame frame = new JFrame("Test Frame");
    frame.setSize(400, 300);
    frame.setVisible(true);
    // frame.setDefaultCloseOperation(
      JFrame.EXIT_ON_CLOSE);
  }
 }
```

NOTE: You must have JDK 1.3 to run the slides.



Swing Equivalents of AWT Components

- JLabel
  - New features: HTML content images, borders
- JButton
  - New features: icons, alignment, mnemonics
- JPanel
  - New feature: borders
- JSlider
  - New features: tick marks and labels



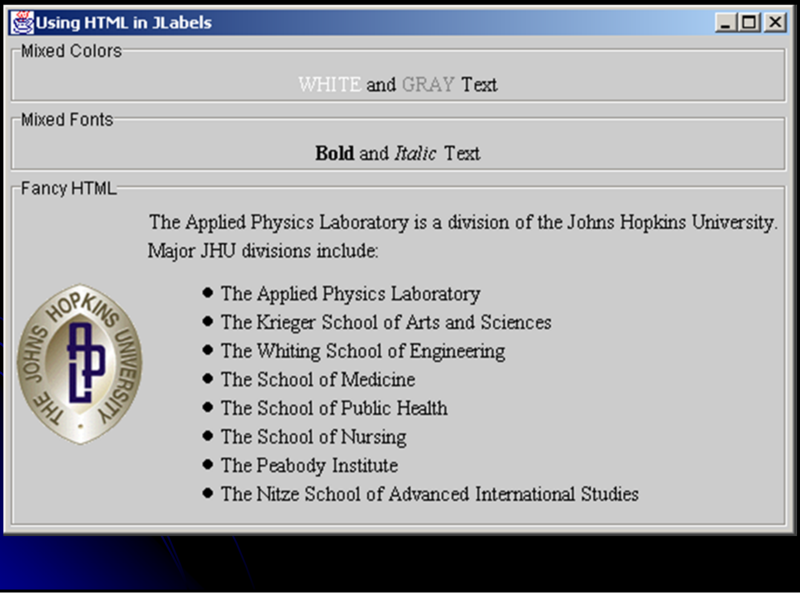
JLabel

- Main new feature: HTML content
  - If text is "<html>...</html>", it gets rendered as HTML
  - HTML labels only work in JDK 1.2.2 or later, or in Swing 1.1.1 or later.
  - In JDK 1.2 the label string must begin with <html>, not <HTML>. It is case-insensitive in JDK 1.3 and 1.4.
  - JLabel fonts are ignored if HTML is used. If you use HTML, all font control must be performed by HTML.
  - You must use <P>, not <BR>, to force a line break.
  - Other HTML support is spotty.
    - Be sure to test each HTML construct you use. Permitting the user to enter HTML text at runtime is asking for trouble.
- Other new features: images, borders

JLabel: Example Code

```
String labelText =
  “<html><FONT COLOR=WHITE>WHITE</FONT> and " +
  "<FONT COLOR=GRAY>GRAY</FONT> Text</html>";
JLabel coloredLabel =
  new JLabel(labelText, JLabel.CENTER);
...
labelText =
  “<html><B>Bold</B> and <I>Italic</I> Text</html>";
JLabel boldLabel =
  new JLabel(labelText, JLabel.CENTER);
labelText =
  “<html>The Applied Physics Laboratory is..." +
  "of the Johns Hopkins University." +
  "<P>" + ... "...</html>";
```

JLabel: Example Output



Mixed Colors

WHITE and GRAY Text

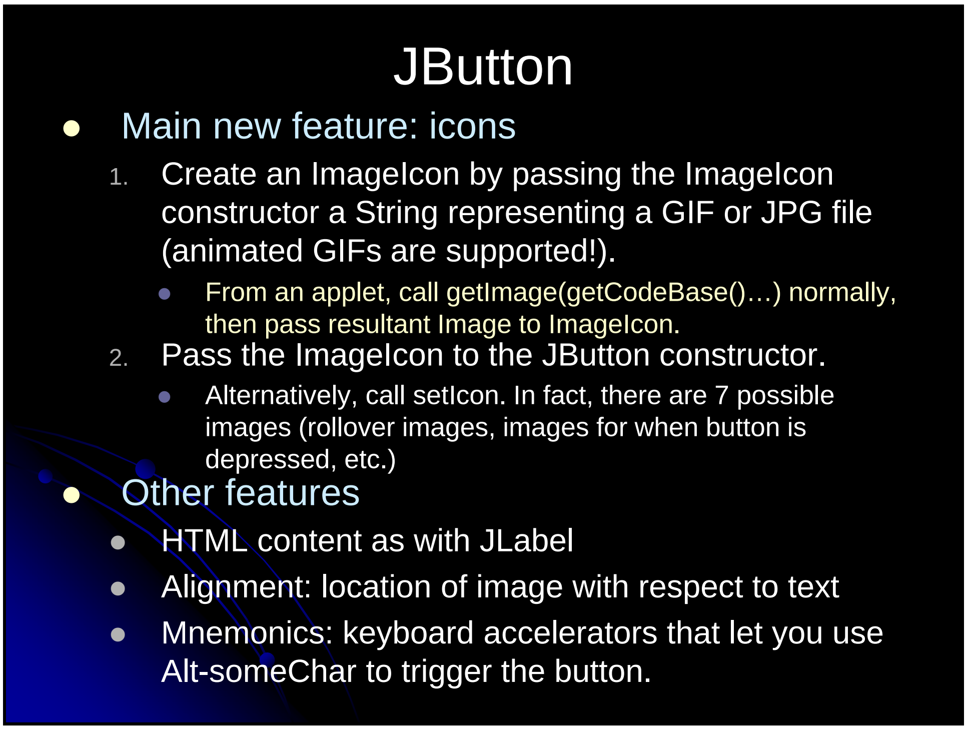
Mixed Fonts

Bold and *Italic* Text

Fancy HTML

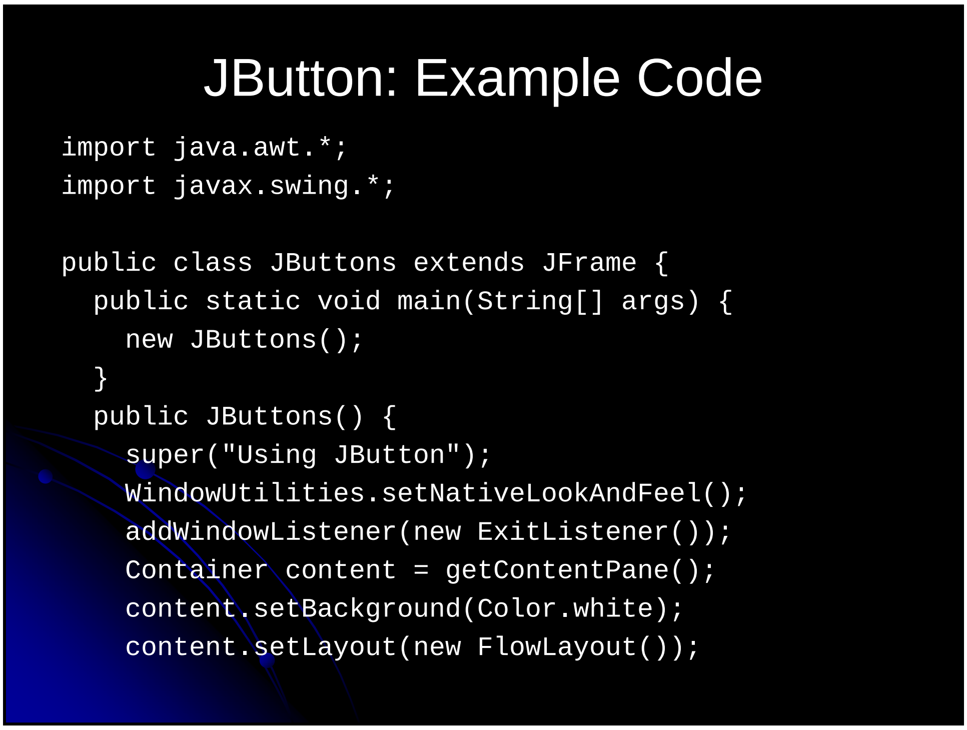
The Applied Physics Laboratory is a division of the Johns Hopkins University. Major JHU divisions include:

- The Applied Physics Laboratory
- The Krieger School of Arts and Sciences
- The Whiting School of Engineering
- The School of Medicine
- The School of Public Health
- The School of Nursing
- The Peabody Institute
- The Nitze School of Advanced International Studies



JButton

- Main new feature: icons
  1. Create an ImageIcon by passing the ImageIcon constructor a String representing a GIF or JPG file (animated GIFs are supported!).
     - From an applet, call getImage(getCodeBase()…) normally, then pass resultant Image to ImageIcon.
  2. Pass the ImageIcon to the JButton constructor.
     - Alternatively, call setIcon. In fact, there are 7 possible images (rollover images, images for when button is depressed, etc.)
- Other features
  - HTML content as with JLabel
  - Alignment: location of image with respect to text
  - Mnemonics: keyboard accelerators that let you use Alt-someChar to trigger the button.

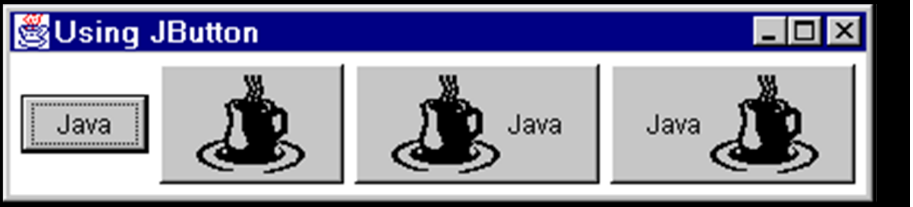


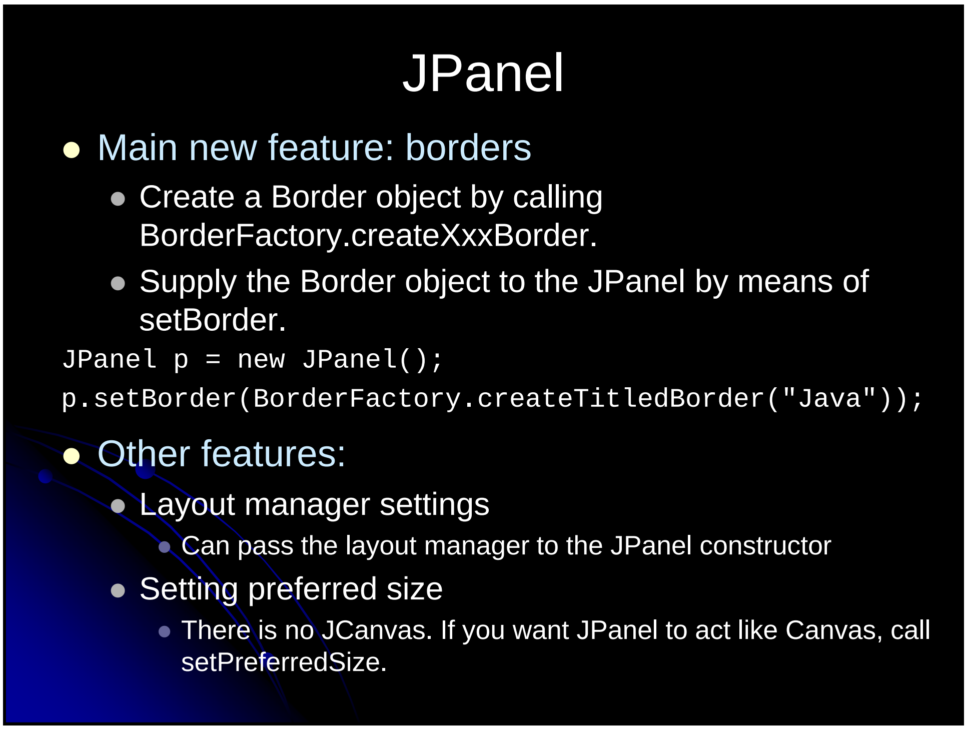
JButton: Example Code

```
import java.awt.*;
import javax.swing.*;

public class JButtons extends JFrame {
  public static void main(String[] args) {
    new JButtons();
  }
  public JButtons() {
    super("Using JButton");
    WindowUtilities.setNativeLookAndFeel();
    addWindowListener(new ExitListener());
    Container content = getContentPane();
    content.setBackground(Color.white);
    content.setLayout(new FlowLayout());
```

```
        JButton button1 = new JButton("Java");
        content.add(button1);
        ImageIcon cup = new ImageIcon("images/cup.gif");
        JButton button2 = new JButton(cup);
        content.add(button2);
        JButton button3 = new JButton("Java", cup);
        content.add(button3);
        JButton button4 = new JButton("Java", cup);
        button4.setHorizontalTextPosition
                                 (SwingConstants.LEFT);
        content.add(button4);
        pack();
        setVisible(true);
      }
    }
```

JButton: Example Output



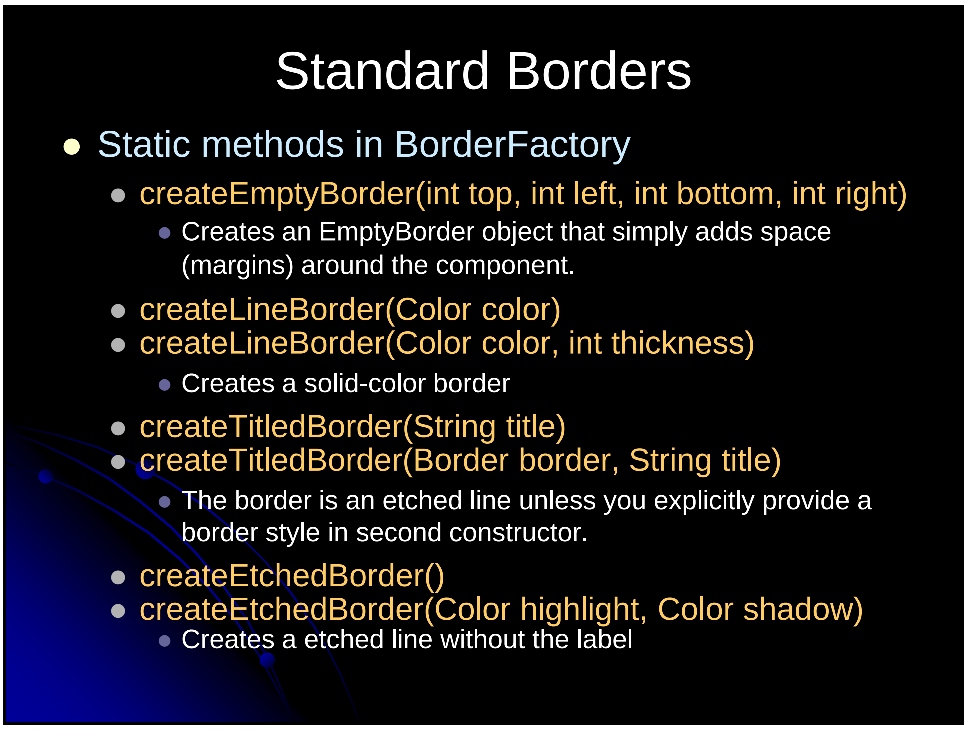


JPanel

- Main new feature: borders
  - Create a Border object by calling BorderFactory.createXxxBorder.
  - Supply the Border object to the JPanel by means of setBorder.

```
JPanel p = new JPanel();
p.setBorder(BorderFactory.createTitledBorder("Java"));
```

- Other features:
  - Layout manager settings
    - Can pass the layout manager to the JPanel constructor
  - Setting preferred size
    - There is no JCanvas. If you want JPanel to act like Canvas, call setPreferredSize.



Standard Borders

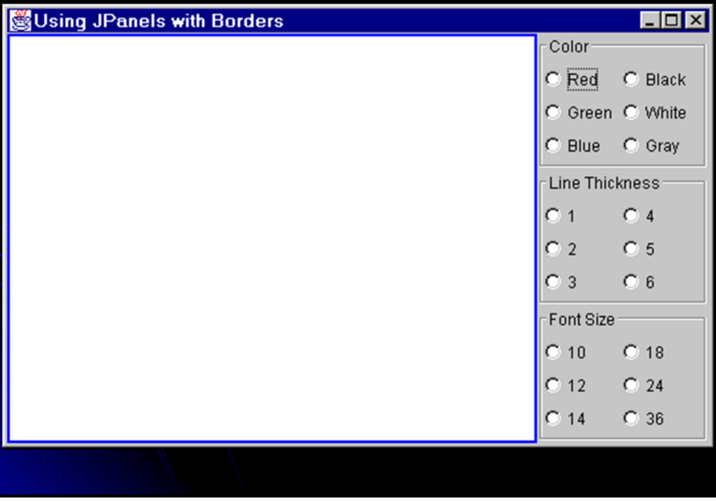
- Static methods in BorderFactory
  - createEmptyBorder(int top, int left, int bottom, int right)
    - Creates an EmptyBorder object that simply adds space (margins) around the component.
  - createLineBorder(Color color)
  - createLineBorder(Color color, int thickness)
    - Creates a solid-color border
  - createTitledBorder(String title)
  - createTitledBorder(Border border, String title)
    - The border is an etched line unless you explicitly provide a border style in second constructor.
  - createEtchedBorder()
  - createEtchedBorder(Color highlight, Color shadow)
    - Creates a etched line without the label

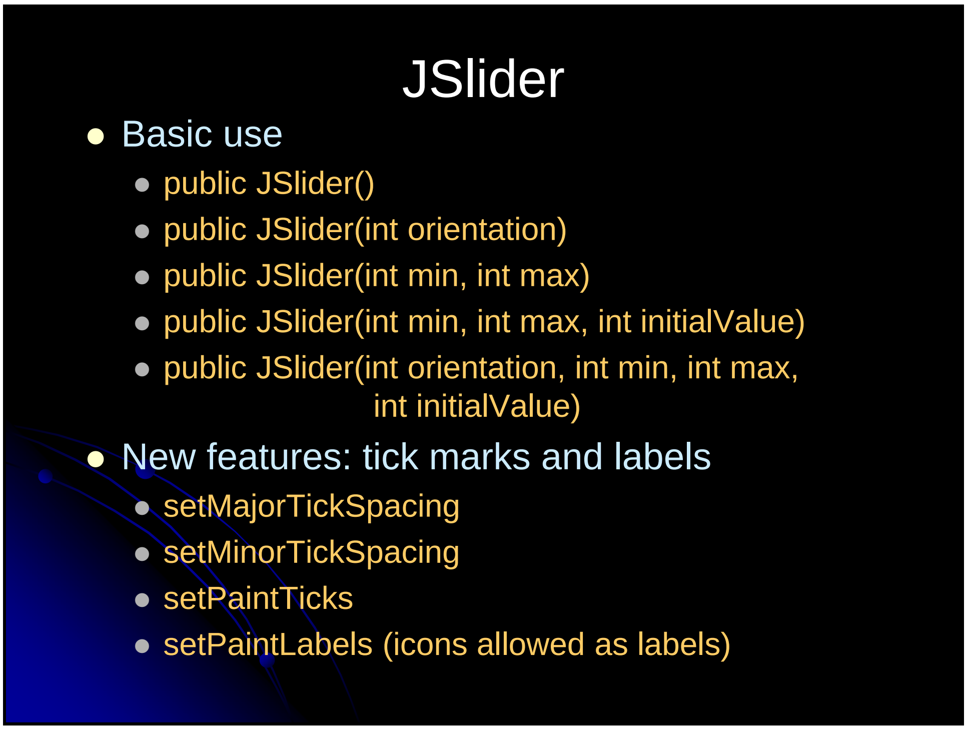
JPanel: Example Code

```
public class SixChoicePanel extends JPanel {
  public SixChoicePanel(String title, String[] buttonLabels){
    super(new GridLayout(3, 2));
    setBackground(Color.lightGray);
    setBorder(BorderFactory.createTitledBorder(title));
    ButtonGroup group = new ButtonGroup();
    JRadioButton option;
    int halfLength = buttonLabels.length/2;
    for(int i=0; i<halfLength; i++) {
      option = new JRadioButton(buttonLabels[i]);
      group.add(option);
      add(option);
      option = new RadioButton(buttonLabels[i+halfLength]);
      group.add(option);
      add(option);
    }
  }
}
```

JPanel: Example Output

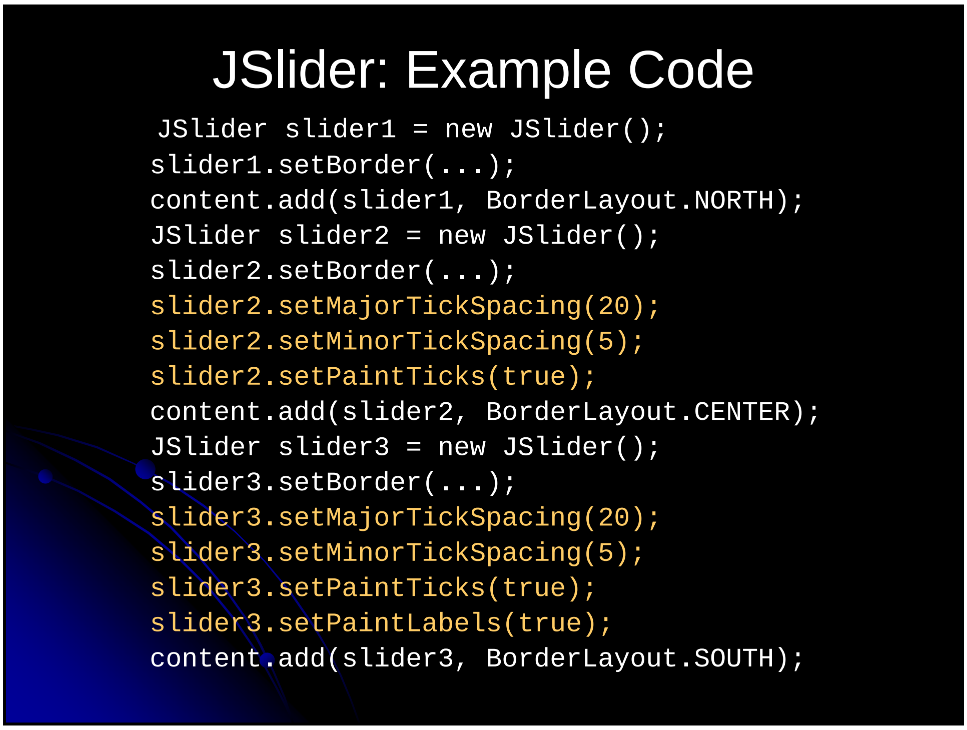
- Left window uses createLineBorder
- Right window has three SixChoicePanels





JSlider

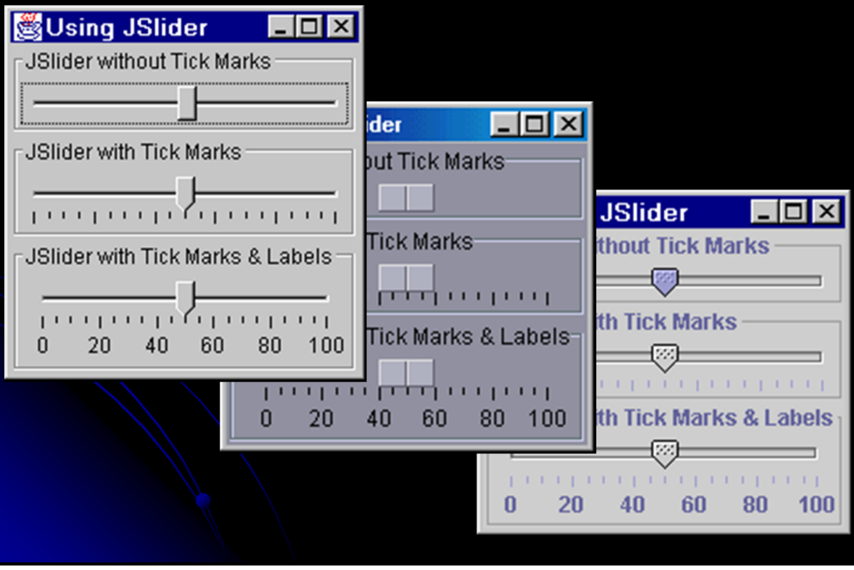
- Basic use
  - public JSlider()
  - public JSlider(int orientation)
  - public JSlider(int min, int max)
  - public JSlider(int min, int max, int initialValue)
  - public JSlider(int orientation, int min, int max, int initialValue)
- New features: tick marks and labels
  - setMajorTickSpacing
  - setMinorTickSpacing
  - setPaintTicks
  - setPaintLabels (icons allowed as labels)



JSlider: Example Code

```
JSlider slider1 = new JSlider();
slider1.setBorder(...);
content.add(slider1, BorderLayout.NORTH);
JSlider slider2 = new JSlider();
slider2.setBorder(...);
slider2.setMajorTickSpacing(20);
slider2.setMinorTickSpacing(5);
slider2.setPaintTicks(true);
content.add(slider2, BorderLayout.CENTER);
JSlider slider3 = new JSlider();
slider3.setBorder(...);
slider3.setMajorTickSpacing(20);
slider3.setMinorTickSpacing(5);
slider3.setPaintTicks(true);
slider3.setPaintLabels(true);
content.add(slider3, BorderLayout.SOUTH);
```

JSlider: Example Output (Windows, Motif, Java LAF)



JColorChooser

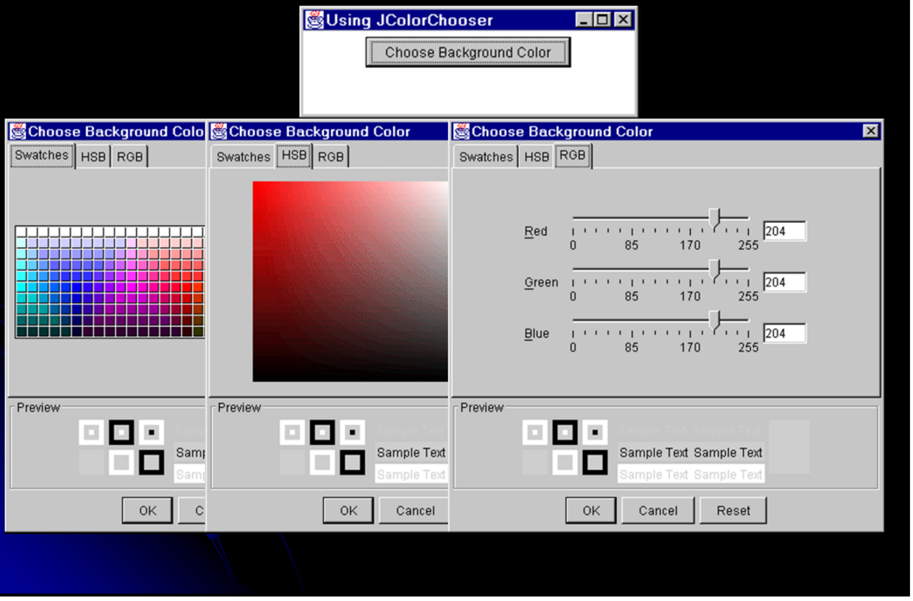
- Open
  - Call JColorChooser.showDialog
    - First argument: parent component
    - Second argument: title string
    - Third argument: initially-selected Color
- Return value
  - Selected Color if "OK" chosen
  - null if "Cancel" chosen

JColorChooser: Example Code

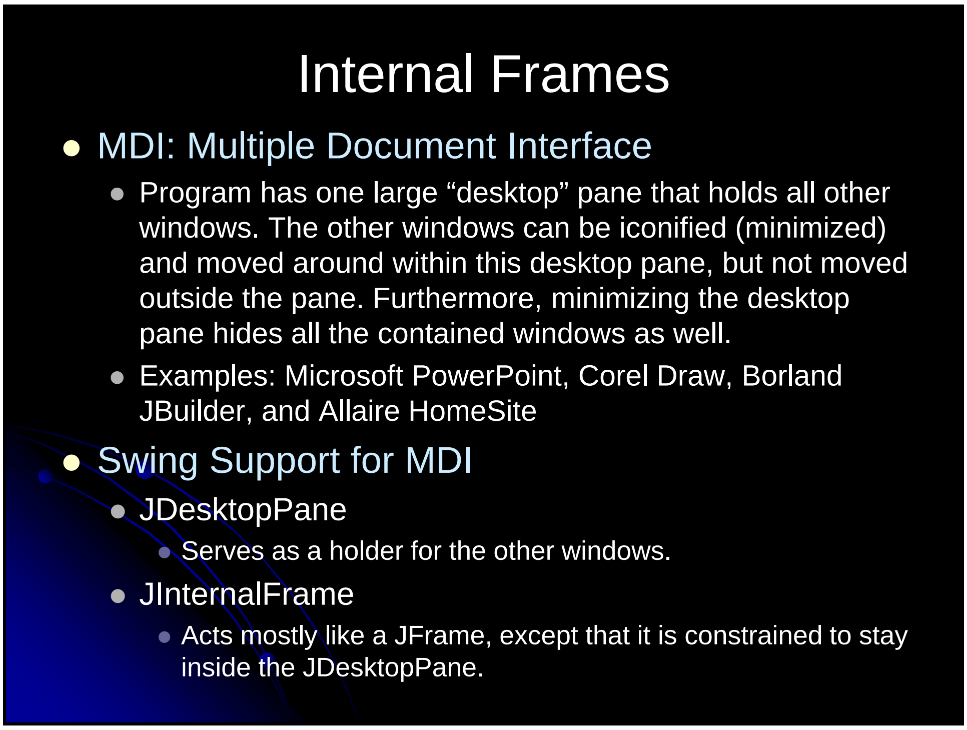
- Button that lets you change color of window

```
public void actionPerformed(ActionEvent e) {
  Color bgColor
    = JColorChooser.showDialog
                  (this,
                   "Choose Background Color",
                   getBackground());
  if (bgColor != null)
    getContentPane().setBackground(bgColor);
}
```

JColorChooser: Example Output

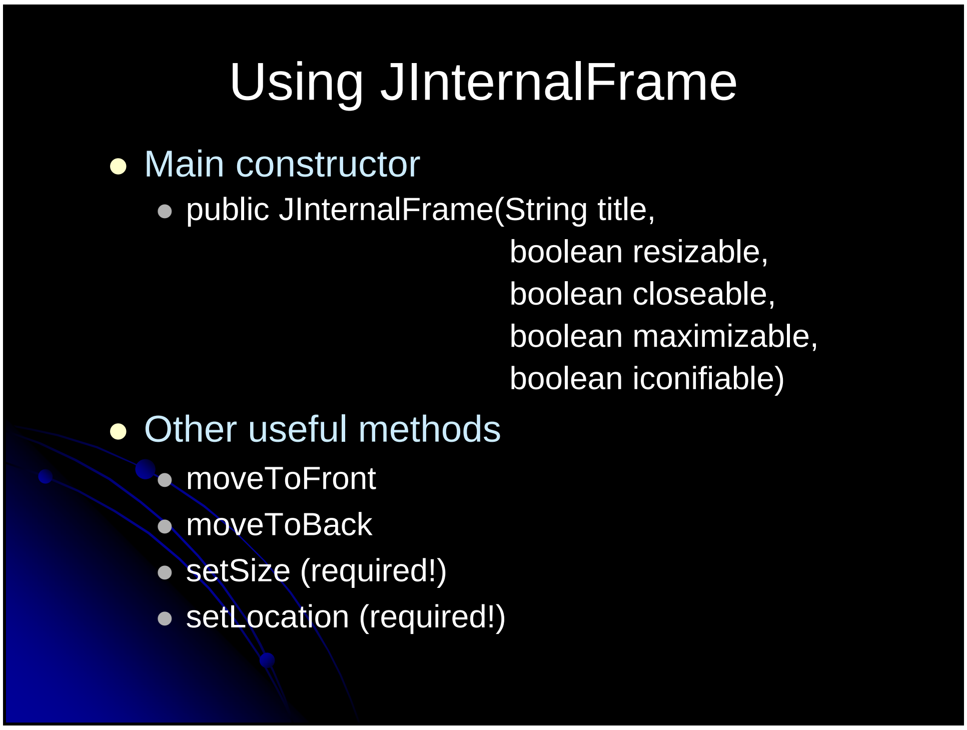


Red 204
Green 204
Blue 204



Internal Frames

- MDI: Multiple Document Interface
  - Program has one large “desktop” pane that holds all other windows. The other windows can be iconified (minimized) and moved around within this desktop pane, but not moved outside the pane. Furthermore, minimizing the desktop pane hides all the contained windows as well.
  - Examples: Microsoft PowerPoint, Corel Draw, Borland JBuilder, and Allaire HomeSite
- Swing Support for MDI
  - JDesktopPane
    - Serves as a holder for the other windows.
  - JInternalFrame
    - Acts mostly like a JFrame, except that it is constrained to stay inside the JDesktopPane.

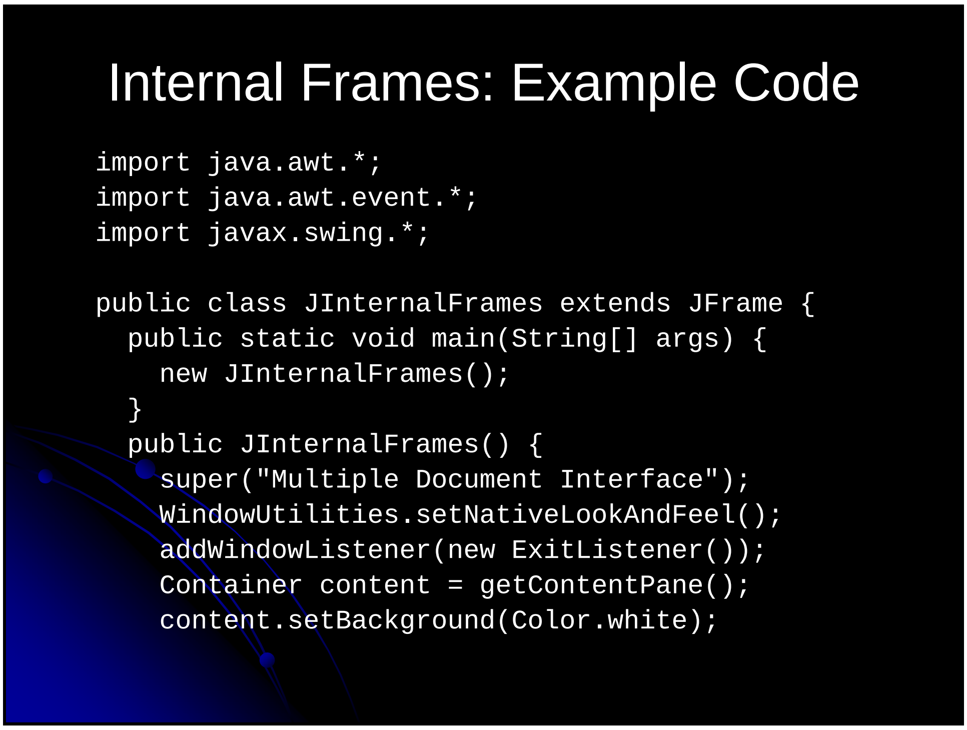


Using JInternalFrame

- Main constructor
  - public JInternalFrame(String title,

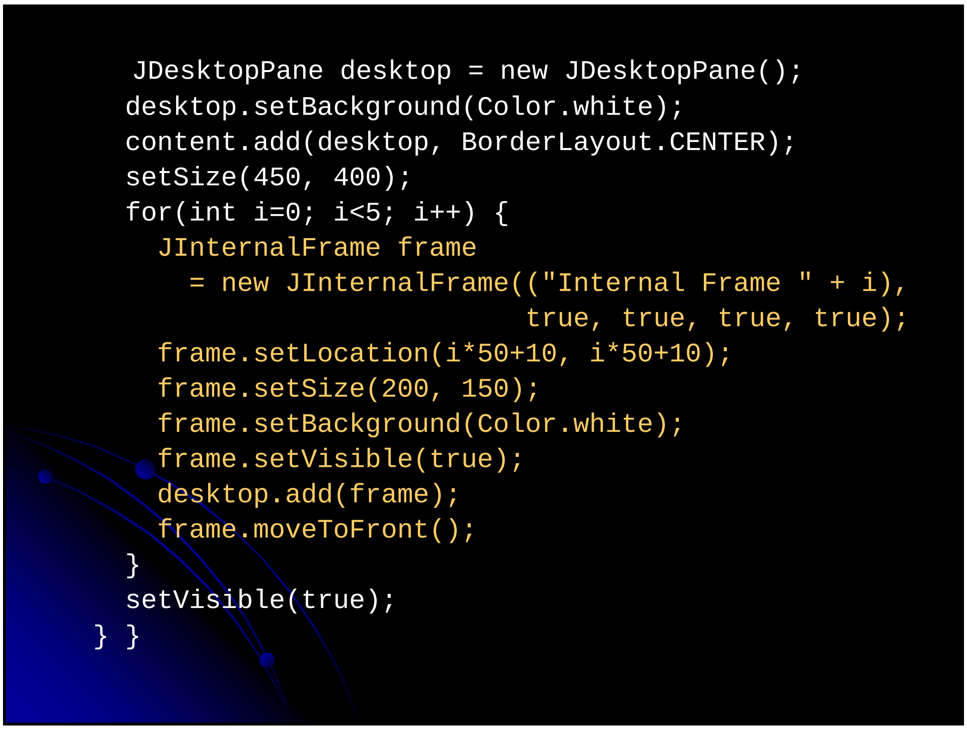
```
                        boolean resizable,
                        boolean closeable,
                        boolean maximizable,
                        boolean iconifiable)
```

- Other useful methods
  - moveToFront
  - moveToBack
  - setSize (required!)
  - setLocation (required!)



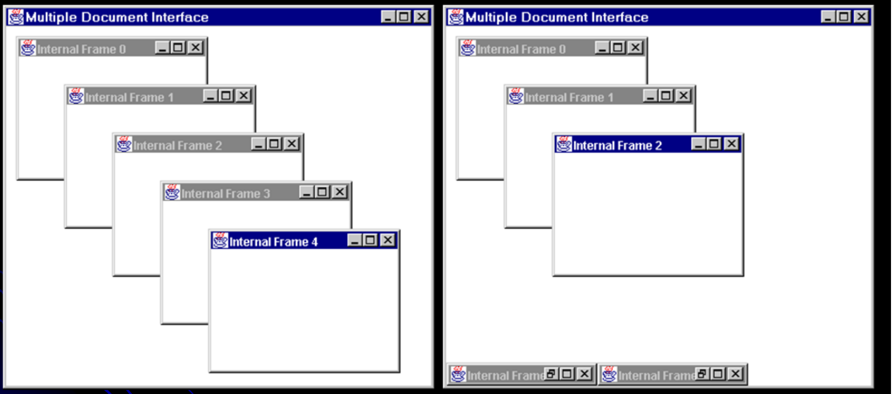
Internal Frames: Example Code

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class JInternalFrames extends JFrame {
  public static void main(String[] args) {
    new JInternalFrames();
  }
  public JInternalFrames() {
    super("Multiple Document Interface");
    WindowUtilities.setNativeLookAndFeel();
    addWindowListener(new ExitListener());
    Container content = getContentPane();
    content.setBackground(Color.white);
```



```
  JDesktopPane desktop = new JDesktopPane();
  desktop.setBackground(Color.white);
  content.add(desktop, BorderLayout.CENTER);
  setSize(450, 400);
  for(int i=0; i<5; i++) {
    JInternalFrame frame
      = new JInternalFrame(("Internal Frame " + i),
                           true, true, true, true);
    frame.setLocation(i*50+10, i*50+10);
    frame.setSize(200, 150);
    frame.setBackground(Color.white);
    frame.setVisible(true);
    desktop.add(frame);
    frame.moveToFront();
  }
  setVisible(true);
} }
```

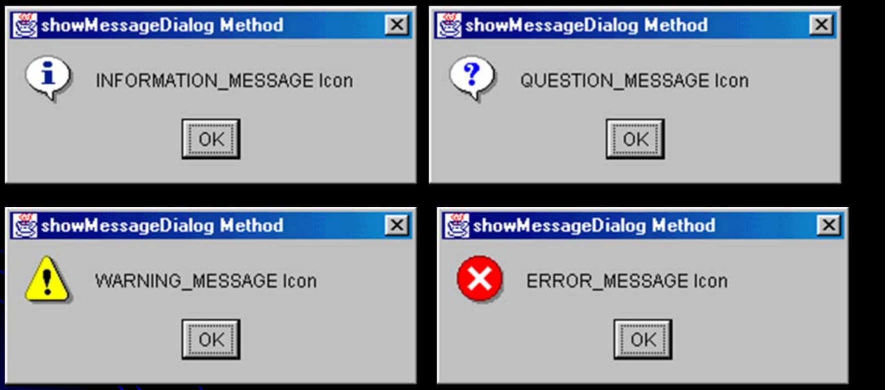
Internal Frames: Example Output



JOptionPane

- Very rich class with many options for different types of dialog boxes.
- Five main static methods
  - JOptionPane.showMessageDialog
    - Icon, message, OK button
  - JOptionPane.showConfirmDialog
    - Icon, message, and buttons:
      OK, OK/Cancel, Yes/No, or Yes/No/Cancel
  - JOptionPane.showInputDialog (2 versions)
    - Icon, message, textfield or combo box, buttons
  - JOptionPane.showOptionDialog
    - Icon, message, array of buttons or other components

JOptionPane Message Dialogs (Windows LAF)



showMessageDialog Method

INFORMATION_MESSAGE Icon

showMessageDialog Method

QUESTION_MESSAGE Icon

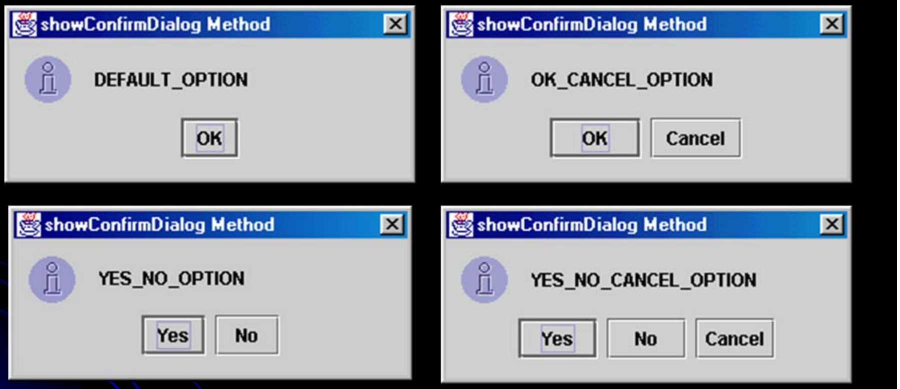
showMessageDialog Method

WARNING_MESSAGE Icon

showMessageDialog Method

ERROR_MESSAGE Icon

JOptionPane Confirmation Dialogs (Java LAF)



showConfirmDialog Method

DEFAULT_OPTION

showConfirmDialog Method

OK_CANCEL_OPTION

showConfirmDialog Method

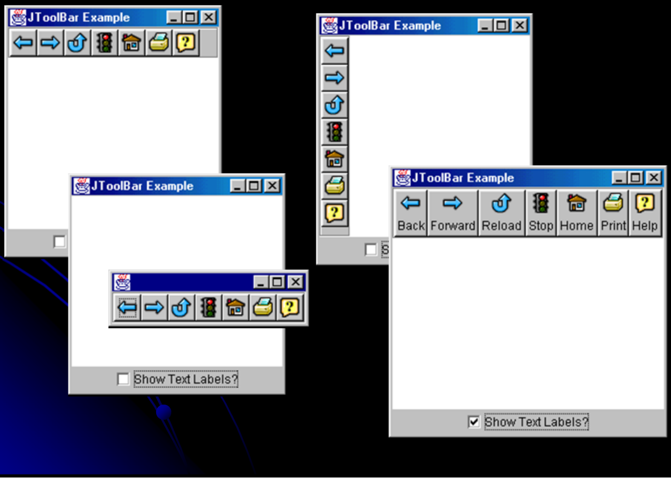
YES_NO_OPTION

showConfirmDialog Method

YES_NO_CANCEL_OPTION

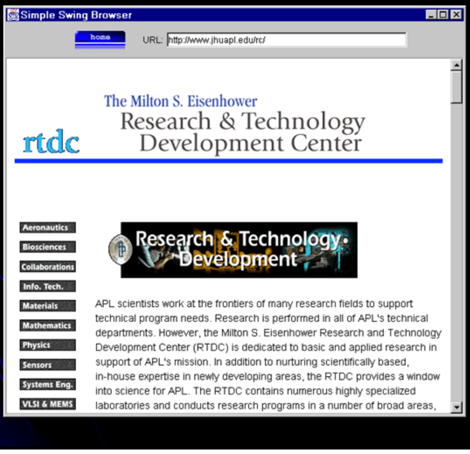
JToolBar

- Acts mostly like a JPanel for buttons
- Dockable: can be dragged and dropped



JEditorPane

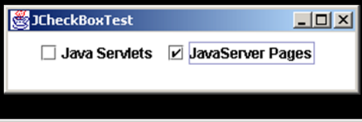
- Acts somewhat like a text area
- Can display HTML and, if HyperLinkListener attached, can follow links



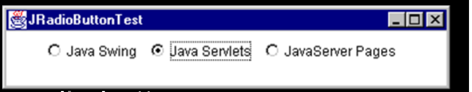
APL scientists work at the frontiers of many research fields to support technical program needs. Research is performed in all of APL's technical departments. However, the Milton S. Eisenhower Research and Technology Development Center (RTDC) is dedicated to basic and applied research in support of APL's mission. In addition to nurturing scientifically based, in-house expertise in newly developing areas, the RTDC provides a window into science for APL. The RTDC contains numerous highly specialized laboratories and conducts research programs in a number of broad areas,

Other Simple Swing Components

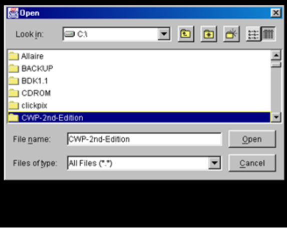
- JCheckBox
  - Note uppercase B (vs. Checkbox in AWT)



- JRadioButton
  - Use a ButtonGroup to link radio buttons



- JTextField
  - Just like AWT TextField except that it does not act as a password field (use JPasswordField for that)
- JTextArea
  - Place in JScrollPane if you want scrolling
- JFileChooser

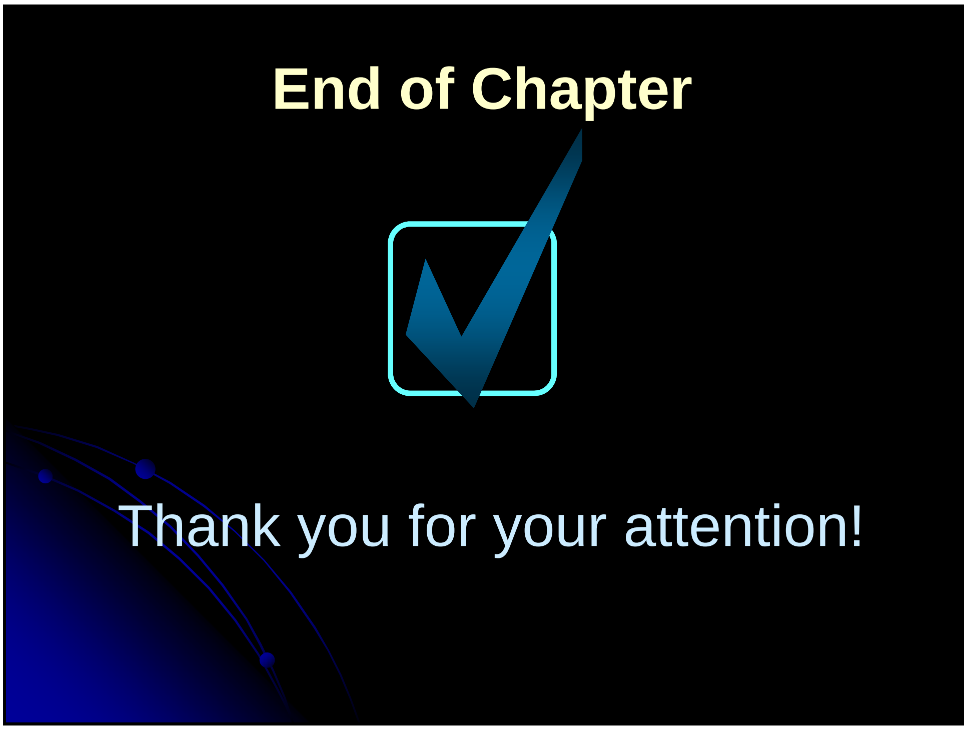


File name: CWP-2nd-Edition

Files of type: All Files (*.*)

Summary

- Port simple AWT components to Swing by adding J to front of class name
- Put custom drawing in paintComponent
  - Call super.paintComponent at beginning unless you turn off double buffering
- Java look and feel is default
  - But you almost always want native look and feel
- Frames and applets use content pane
  - Don't put anything directly in window
- Most components support borders & icons
- Many new components



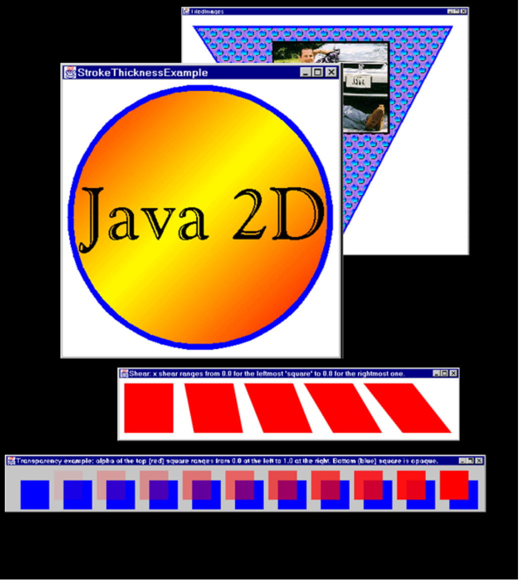
End of Chapter

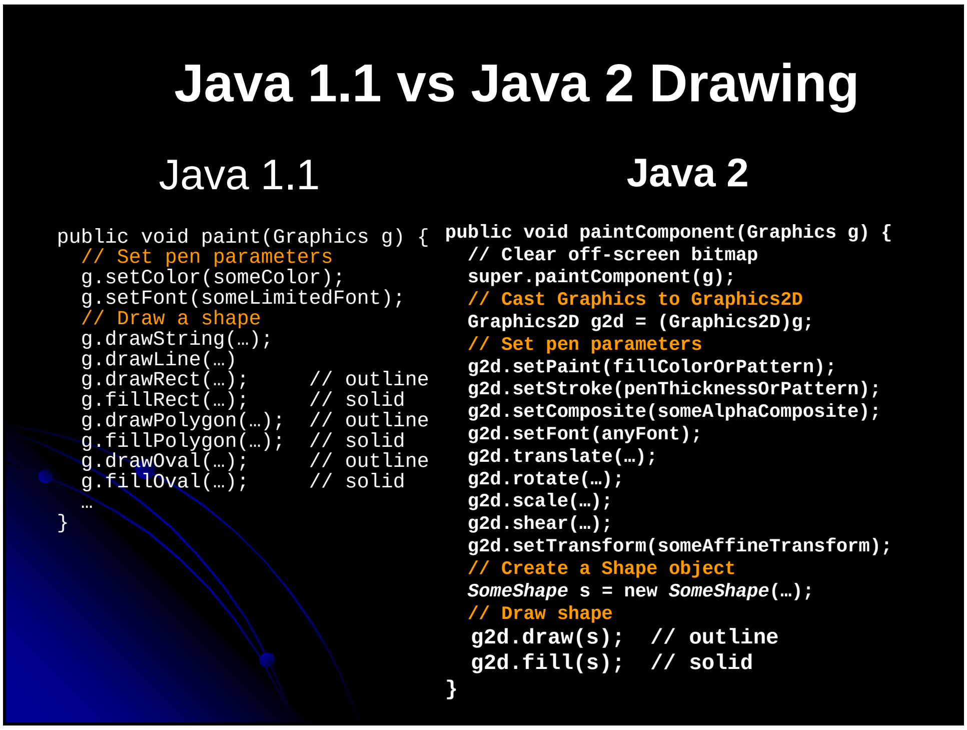
Thank you for your attention!

Drawing in Java 2

Agenda

- Overview
- Drawing Shapes
- Paint Styles
- Transparency
- Using Local Fonts
- Stroke Styles
- Coordinate Transformations
- Requesting Drawing Accuracy


TiledImages
StrokeThicknessExample
Java 2D
Shear: x shear ranges from 0.0 for the leftmost 'square' to 0.8 for the rightmost one.
Transparency example: alpha of the top (red) square ranges from 0.0 at the left to 1.0 at the right. Bottom (blue) square is opaque.



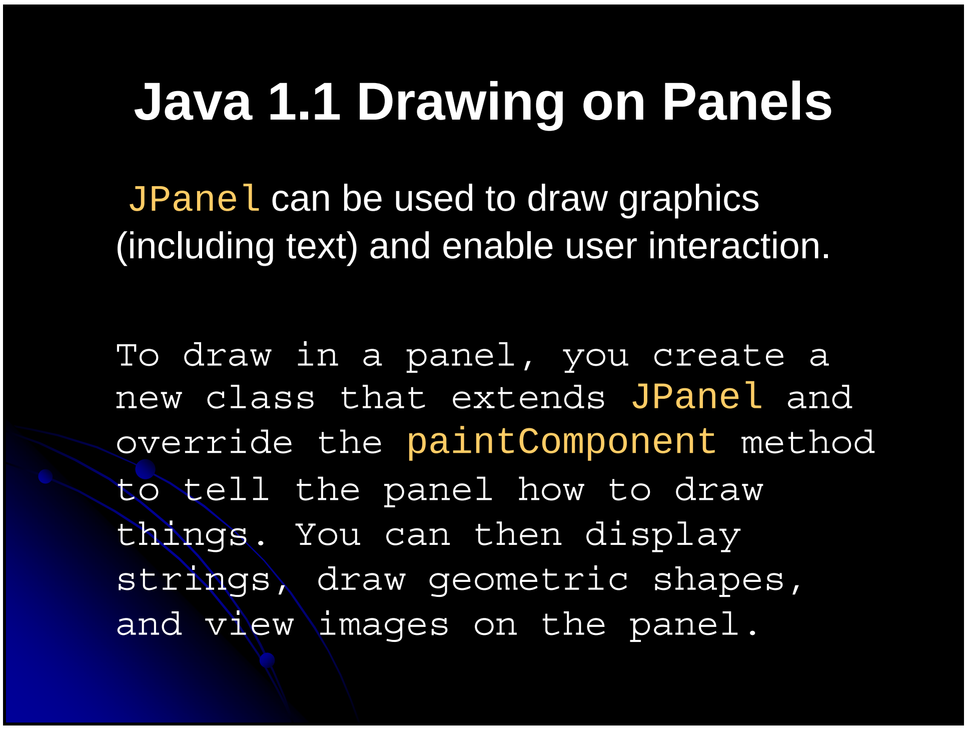
Java 1.1 vs Java 2 Drawing

Java 1.1

```
public void paint(Graphics g) {
  // Set pen parameters
  g.setColor(someColor);
  g.setFont(someLimitedFont);
  // Draw a shape
  g.drawString(…);
  g.drawLine(…)
  g.drawRect(…);     // outline
  g.fillRect(…);     // solid
  g.drawPolygon(…);  // outline
  g.fillPolygon(…);  // solid
  g.drawOval(…);     // outline
  g.fillOval(…);     // solid
  …
}
```

Java 2

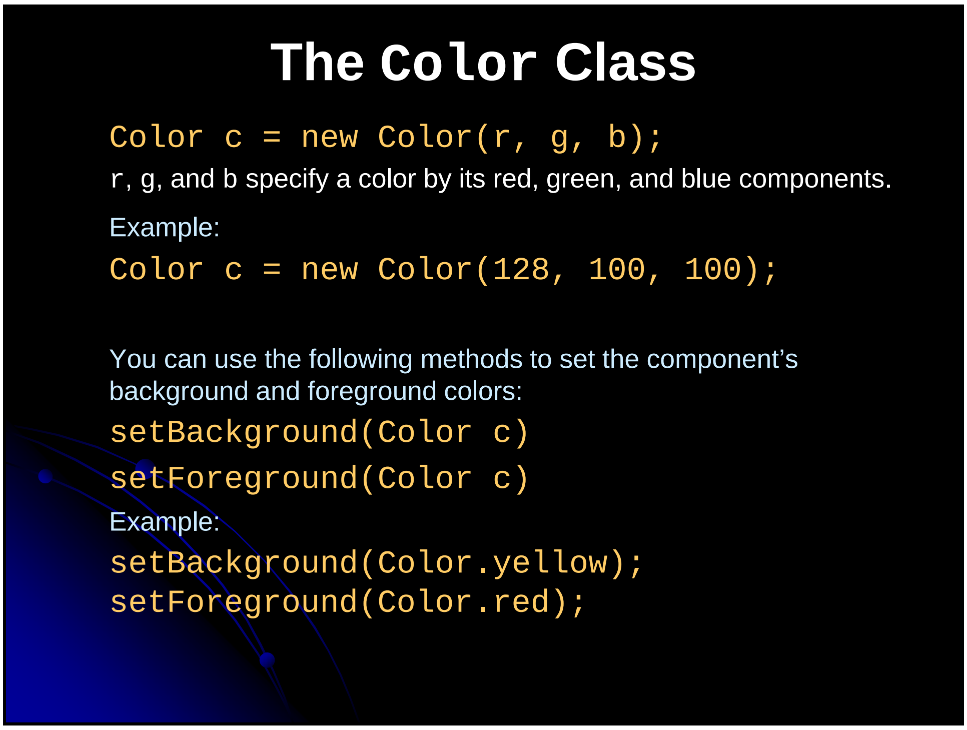
```
public void paintComponent(Graphics g) {
  // Clear off-screen bitmap
  super.paintComponent(g);
  // Cast Graphics to Graphics2D
  Graphics2D g2d = (Graphics2D)g;
  // Set pen parameters
  g2d.setPaint(fillColorOrPattern);
  g2d.setStroke(penThicknessOrPattern);
  g2d.setComposite(someAlphaComposite);
  g2d.setFont(anyFont);
  g2d.translate(…);
  g2d.rotate(…);
  g2d.scale(…);
  g2d.shear(…);
  g2d.setTransform(someAffineTransform);
  // Create a Shape object
  SomeShape s = new SomeShape(…);
  // Draw shape
  g2d.draw(s);  // outline
  g2d.fill(s);  // solid
}
```



Java 1.1 Drawing on Panels

`JPanel` can be used to draw graphics (including text) and enable user interaction.

To draw in a panel, you create a new class that extends `JPanel` and override the `paintComponent` method to tell the panel how to draw things. You can then display strings, draw geometric shapes, and view images on the panel.



The Color Class

```
Color c = new Color(r, g, b);
```

r, g, and b specify a color by its red, green, and blue components.

Example:

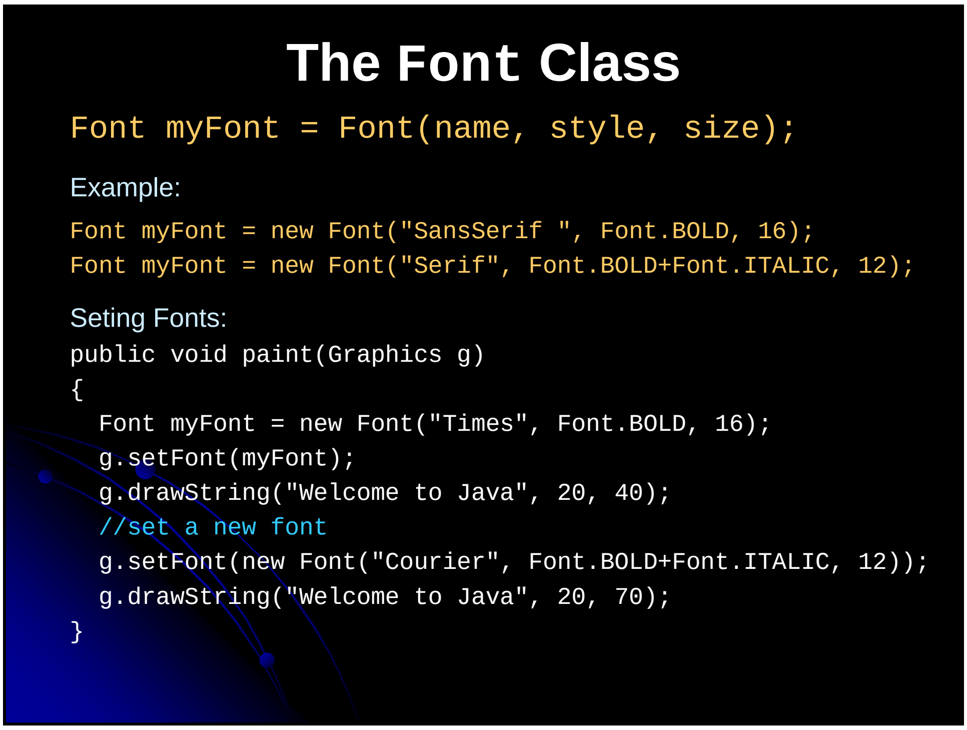
```
Color c = new Color(128, 100, 100);
```

You can use the following methods to set the component’s background and foreground colors:

```
setBackground(Color c)
setForeground(Color c)
```

Example:

```
setBackground(Color.yellow);
setForeground(Color.red);
```



The Font Class

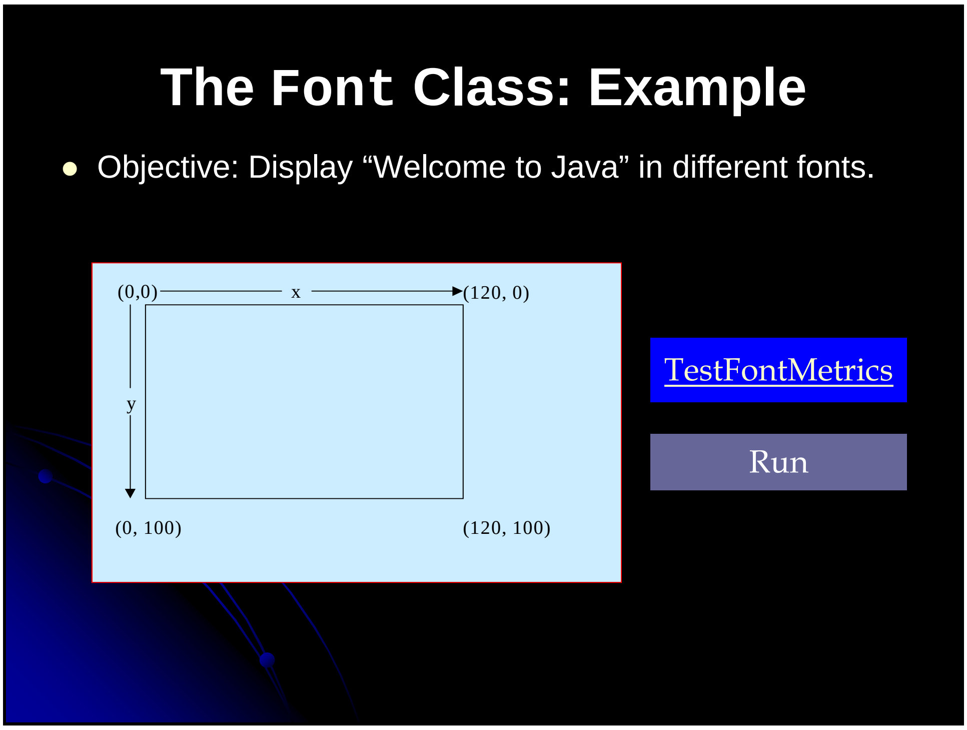
```
Font myFont = Font(name, style, size);
```

Example:

```
Font myFont = new Font("SansSerif ", Font.BOLD, 16);
Font myFont = new Font("Serif", Font.BOLD+Font.ITALIC, 12);
```

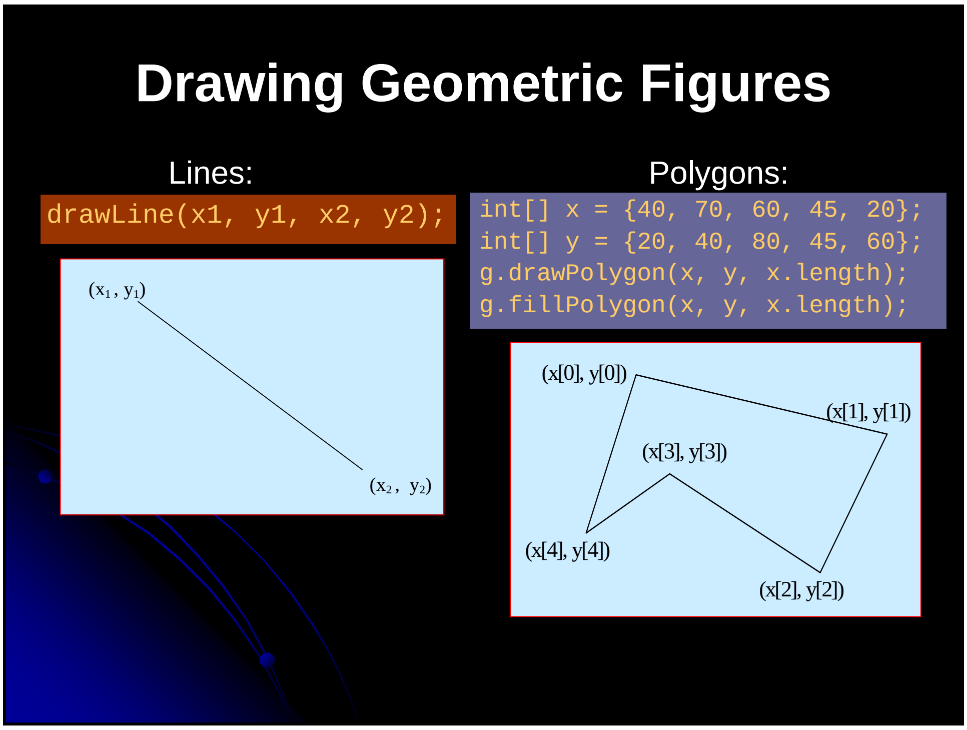
Seting Fonts:

```
public void paint(Graphics g)
{
  Font myFont = new Font("Times", Font.BOLD, 16);
  g.setFont(myFont);
  g.drawString("Welcome to Java", 20, 40);
  //set a new font
  g.setFont(new Font("Courier", Font.BOLD+Font.ITALIC, 12));
  g.drawString("Welcome to Java", 20, 70);
}
```



The Font Class: Example

- Objective: Display “Welcome to Java” in different fonts.



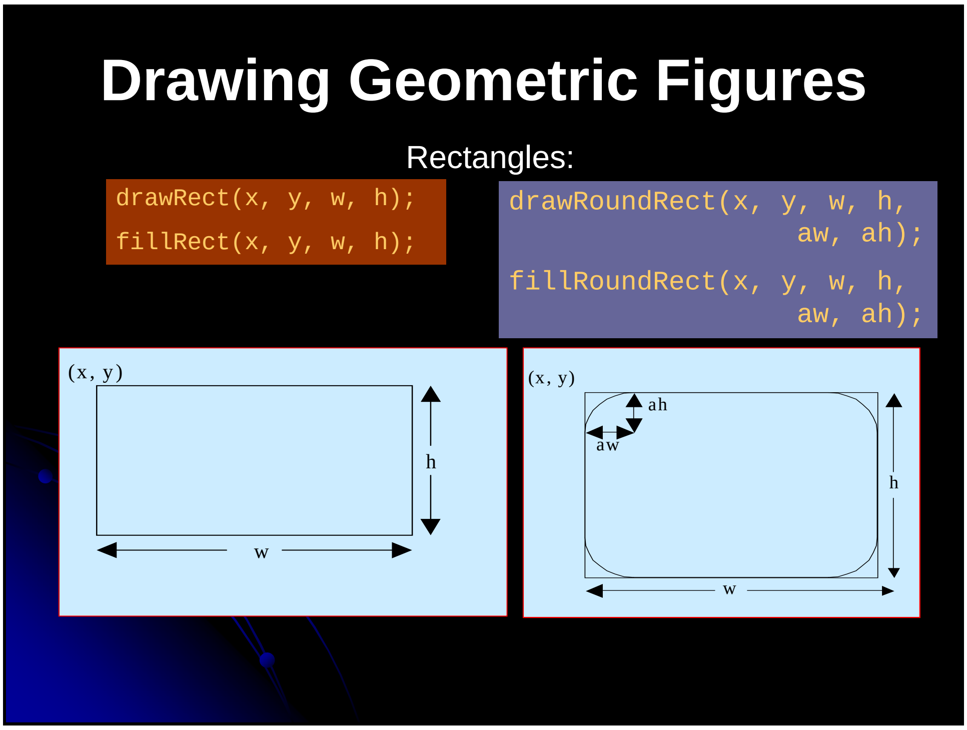
Drawing Geometric Figures

Lines:

```
drawLine(x1, y1, x2, y2);
```

Polygons:

```
int[] x = {40, 70, 60, 45, 20};
int[] y = {20, 40, 80, 45, 60};
g.drawPolygon(x, y, x.length);
g.fillPolygon(x, y, x.length);
```

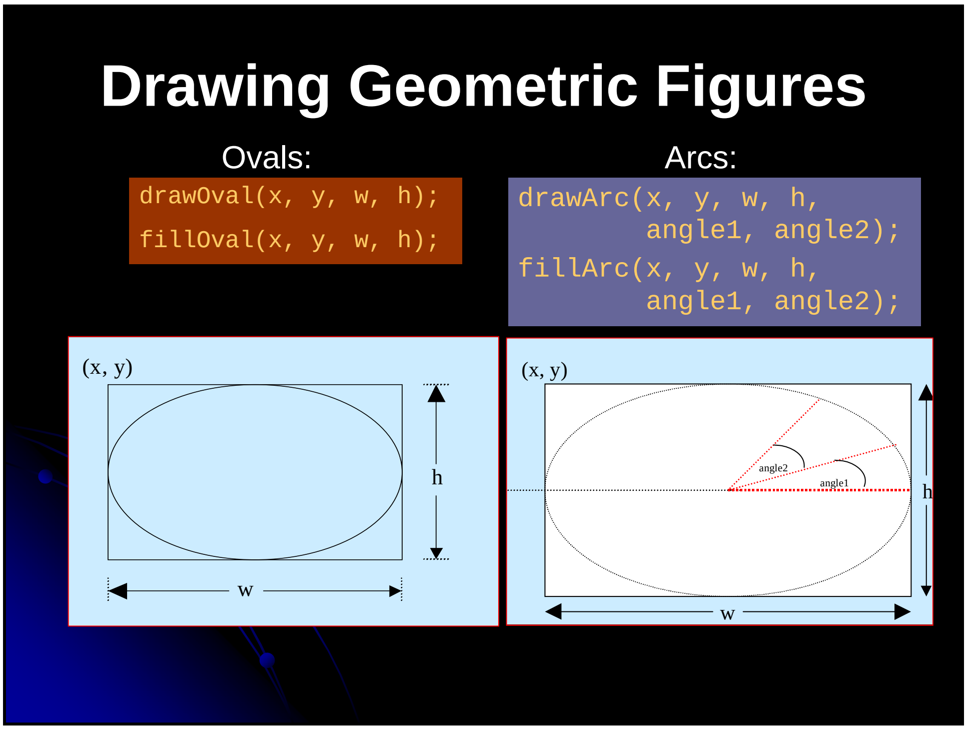


Drawing Geometric Figures

Rectangles:

```
drawRect(x, y, w, h);
fillRect(x, y, w, h);
```

```
drawRoundRect(x, y, w, h,
                    aw, ah);
fillRoundRect(x, y, w, h,
                    aw, ah);
```



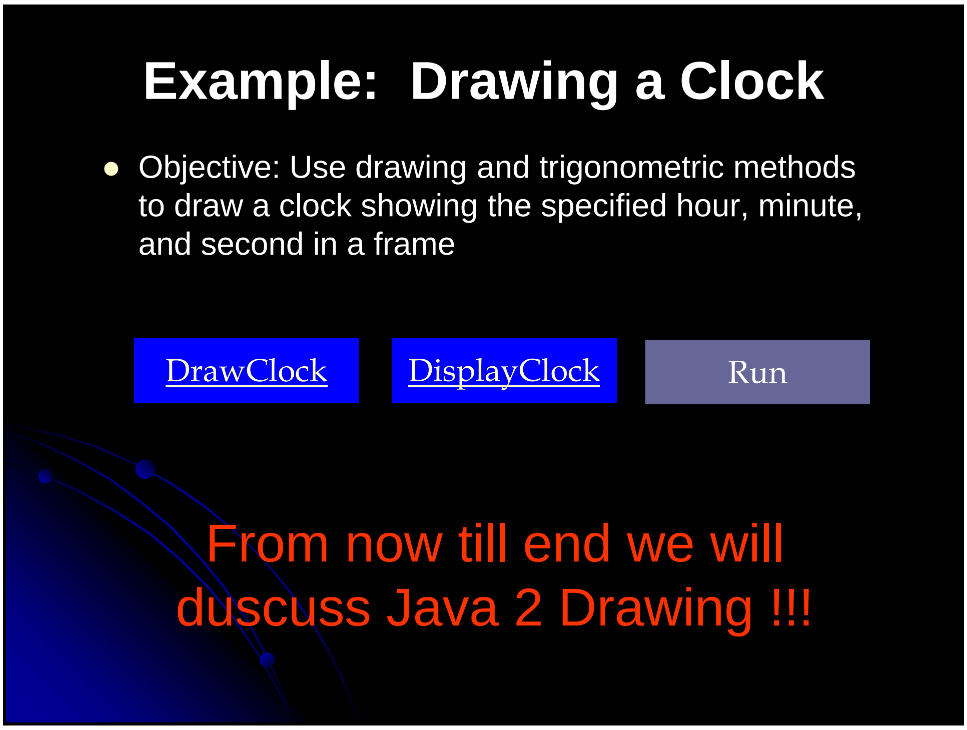
Drawing Geometric Figures

Ovals:

```
drawOval(x, y, w, h);
fillOval(x, y, w, h);
```

Arcs:

```
drawArc(x, y, w, h,
         angle1, angle2);
fillArc(x, y, w, h,
         angle1, angle2);
```

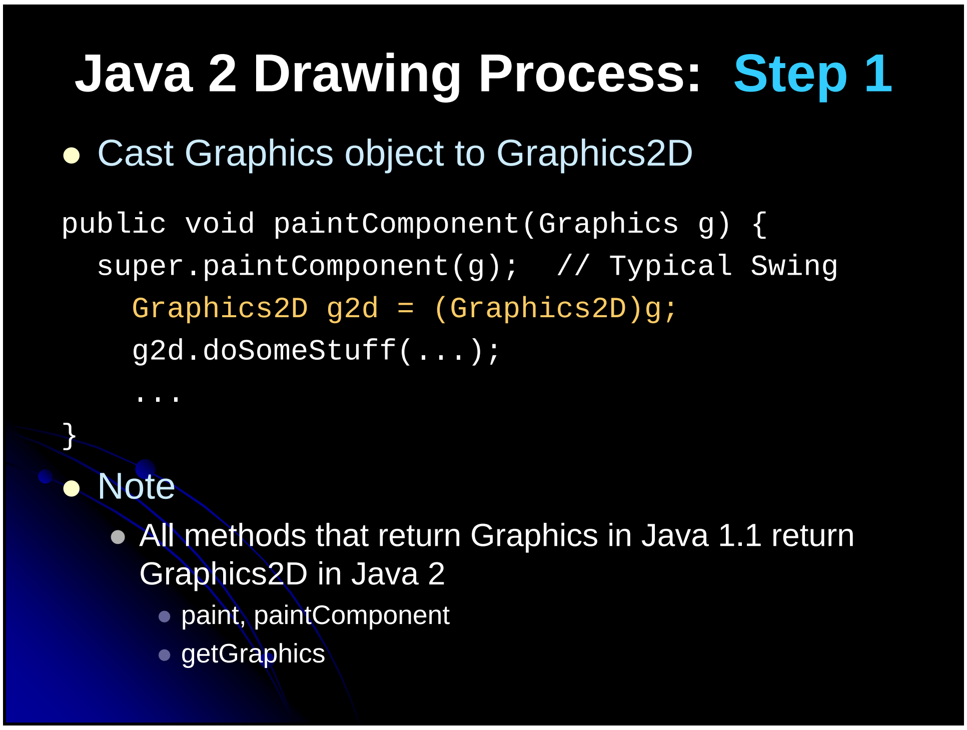


Example: Drawing a Clock

- Objective: Use drawing and trigonometric methods to draw a clock showing the specified hour, minute, and second in a frame

DrawClock DisplayClock Run

From now till end we will duscuss Java 2 Drawing !!!

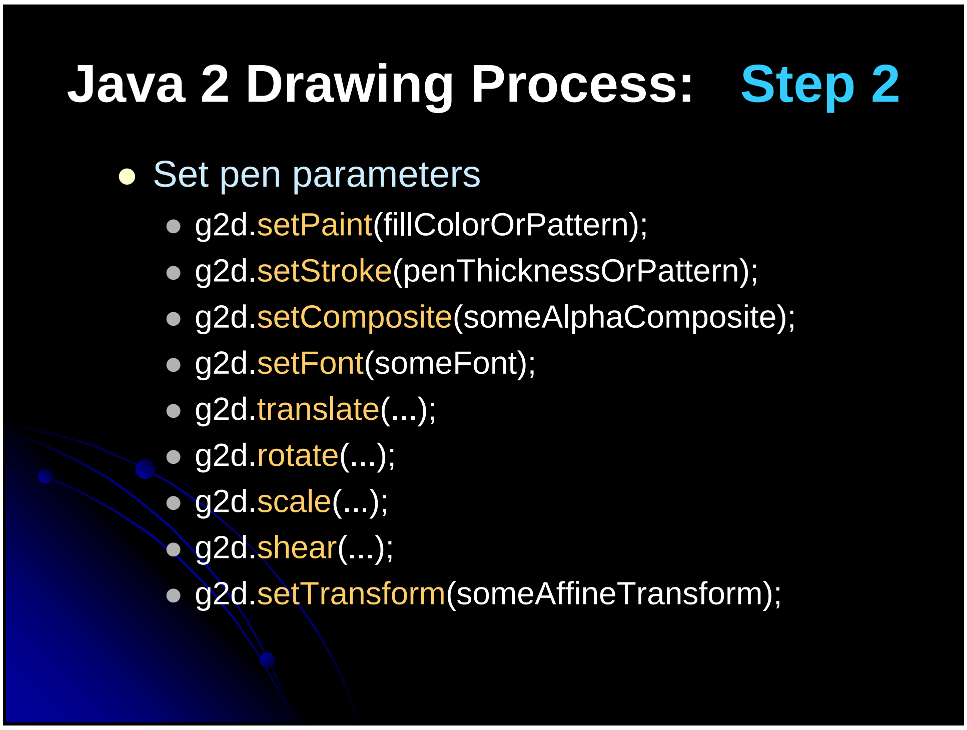


Java 2 Drawing Process: Step 1

- Cast Graphics object to Graphics2D

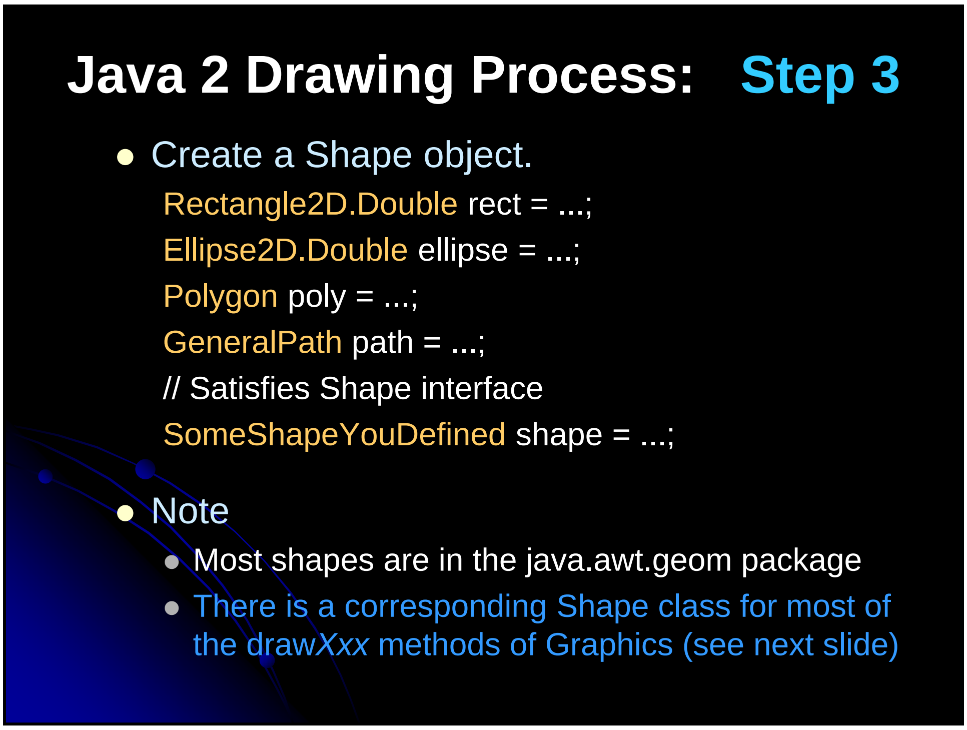
```
public void paintComponent(Graphics g) {
  super.paintComponent(g);  // Typical Swing
    Graphics2D g2d = (Graphics2D)g;
    g2d.doSomeStuff(...);
    ...
}
```

- Note
  - All methods that return Graphics in Java 1.1 return Graphics2D in Java 2
    - paint, paintComponent
    - getGraphics



Java 2 Drawing Process: Step 2

- Set pen parameters
  - g2d.setPaint(fillColorOrPattern);
  - g2d.setStroke(penThicknessOrPattern);
  - g2d.setComposite(someAlphaComposite);
  - g2d.setFont(someFont);
  - g2d.translate(...);
  - g2d.rotate(...);
  - g2d.scale(...);
  - g2d.shear(...);
  - g2d.setTransform(someAffineTransform);

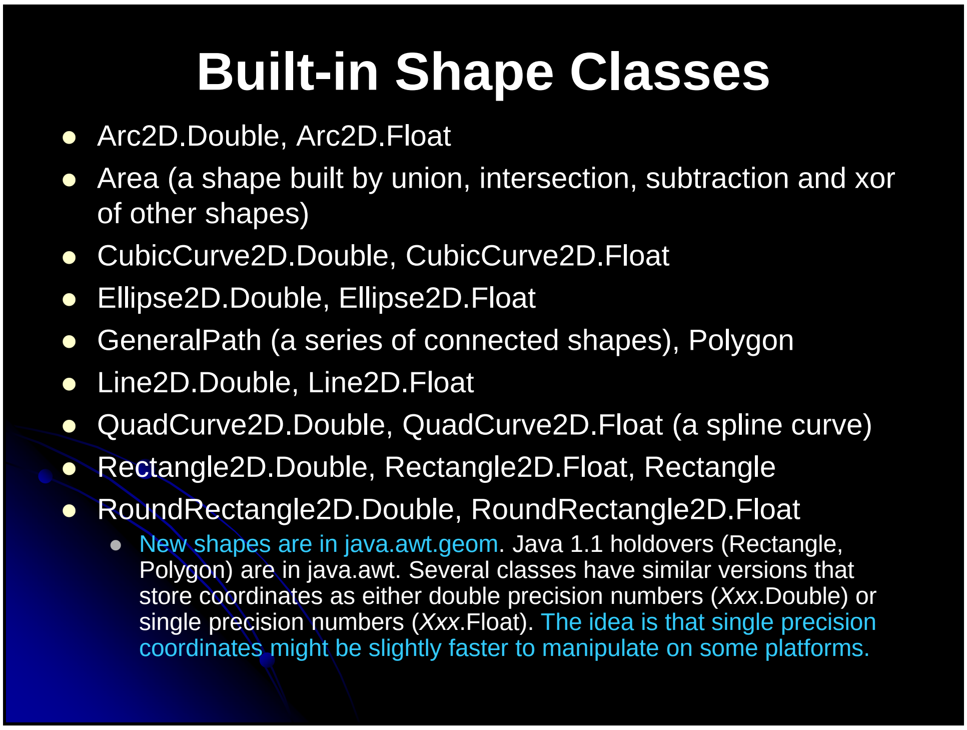


Java 2 Drawing Process: Step 3

- Create a Shape object.

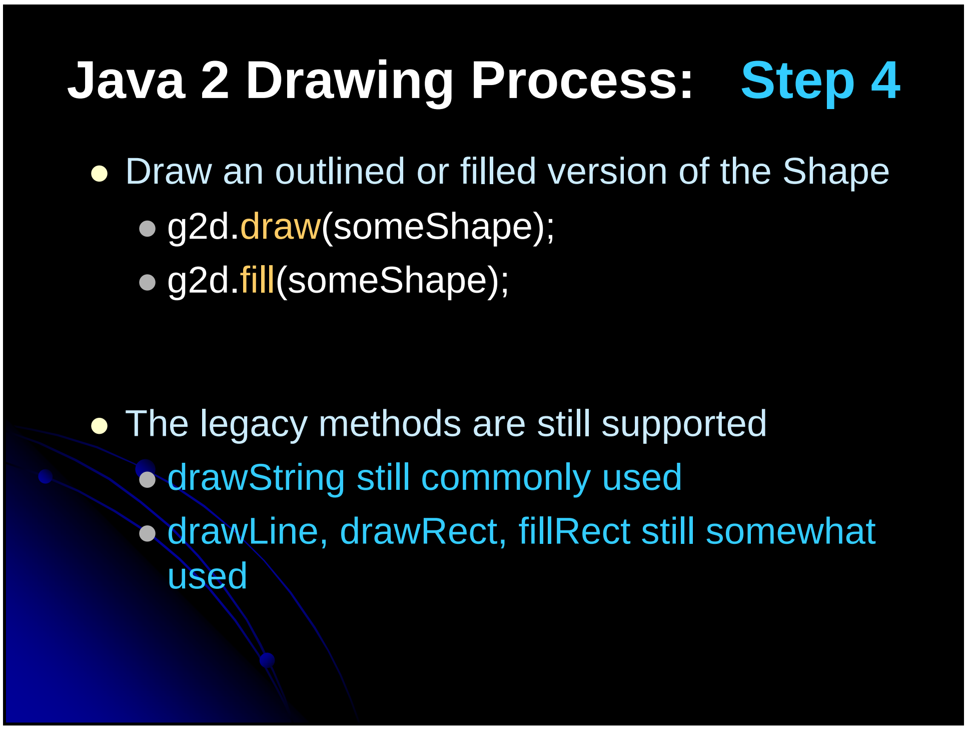
```
Rectangle2D.Double rect = ...;
Ellipse2D.Double ellipse = ...;
Polygon poly = ...;
GeneralPath path = ...;
// Satisfies Shape interface
SomeShapeYouDefined shape = ...;
```

- Note
  - Most shapes are in the java.awt.geom package
  - There is a corresponding Shape class for most of the draw*Xxx* methods of Graphics (see next slide)



Built-in Shape Classes

- Arc2D.Double, Arc2D.Float
- Area (a shape built by union, intersection, subtraction and xor of other shapes)
- CubicCurve2D.Double, CubicCurve2D.Float
- Ellipse2D.Double, Ellipse2D.Float
- GeneralPath (a series of connected shapes), Polygon
- Line2D.Double, Line2D.Float
- QuadCurve2D.Double, QuadCurve2D.Float (a spline curve)
- Rectangle2D.Double, Rectangle2D.Float, Rectangle
- RoundRectangle2D.Double, RoundRectangle2D.Float
  - New shapes are in java.awt.geom. Java 1.1 holdovers (Rectangle, Polygon) are in java.awt. Several classes have similar versions that store coordinates as either double precision numbers (*Xxx*.Double) or single precision numbers (*Xxx*.Float). The idea is that single precision coordinates might be slightly faster to manipulate on some platforms.



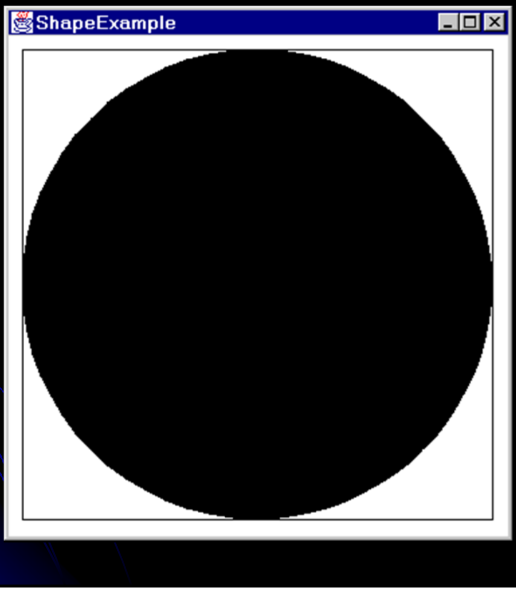
Java 2 Drawing Process: Step 4

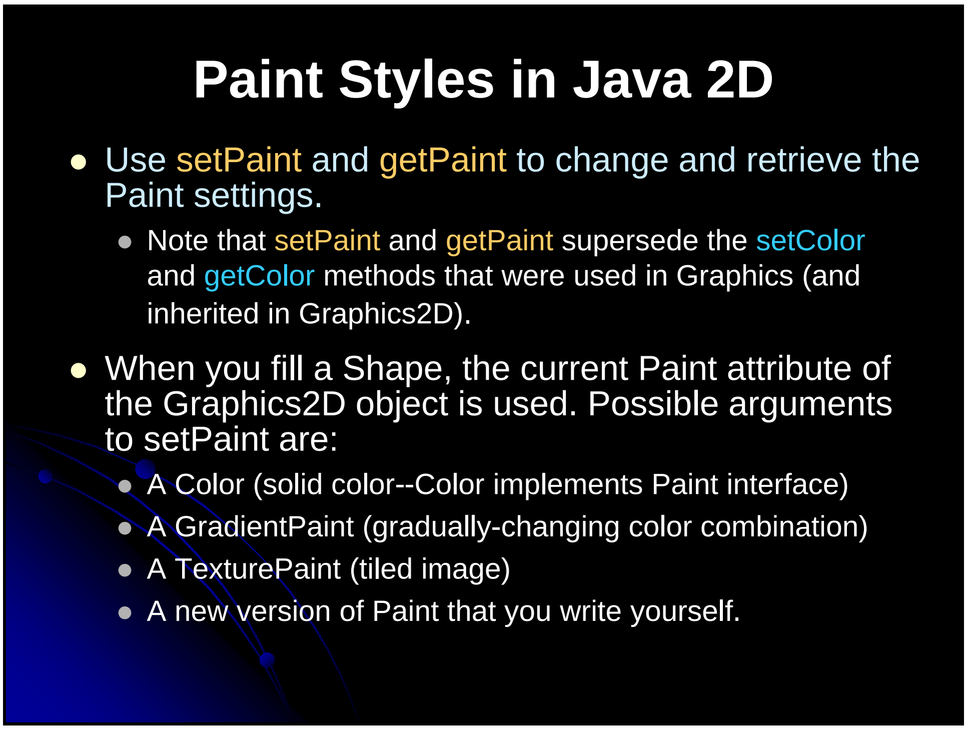
- Draw an outlined or filled version of the Shape
  - g2d.draw(someShape);
  - g2d.fill(someShape);
- The legacy methods are still supported
  - drawString still commonly used
  - drawLine, drawRect, fillRect still somewhat used

Drawing Shapes: Example Code

```
import javax.swing.*;   // For JPanel, etc.
import java.awt.*;      // For Graphics, etc.
import java.awt.geom.*; // For Ellipse2D, etc.
public class ShapeExample extends JPanel {
  private Ellipse2D.Double circle =
    new Ellipse2D.Double(10, 10, 350, 350);
  private Rectangle2D.Double square =
    new Rectangle2D.Double(10, 10, 350, 350);

  public void paintComponent(Graphics g) {
    clear(g);  // ie super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    g2d.fill(circle);
    g2d.draw(square);
  }
     // Code to put JPanel in JFrame omitted.
}
```

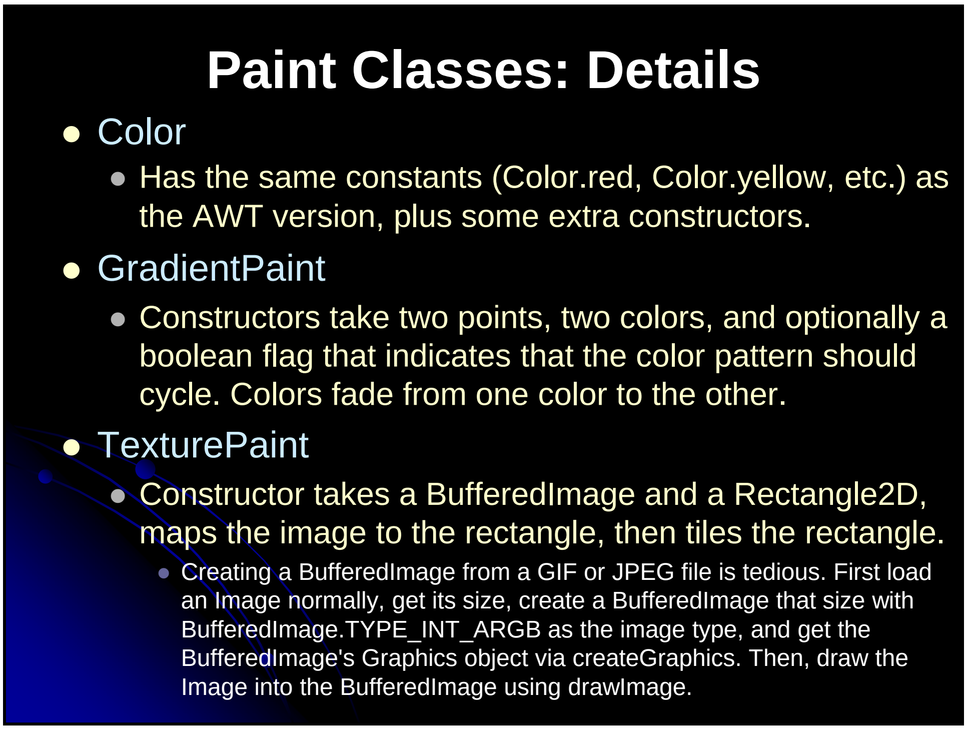
Drawing Shapes: Example Output





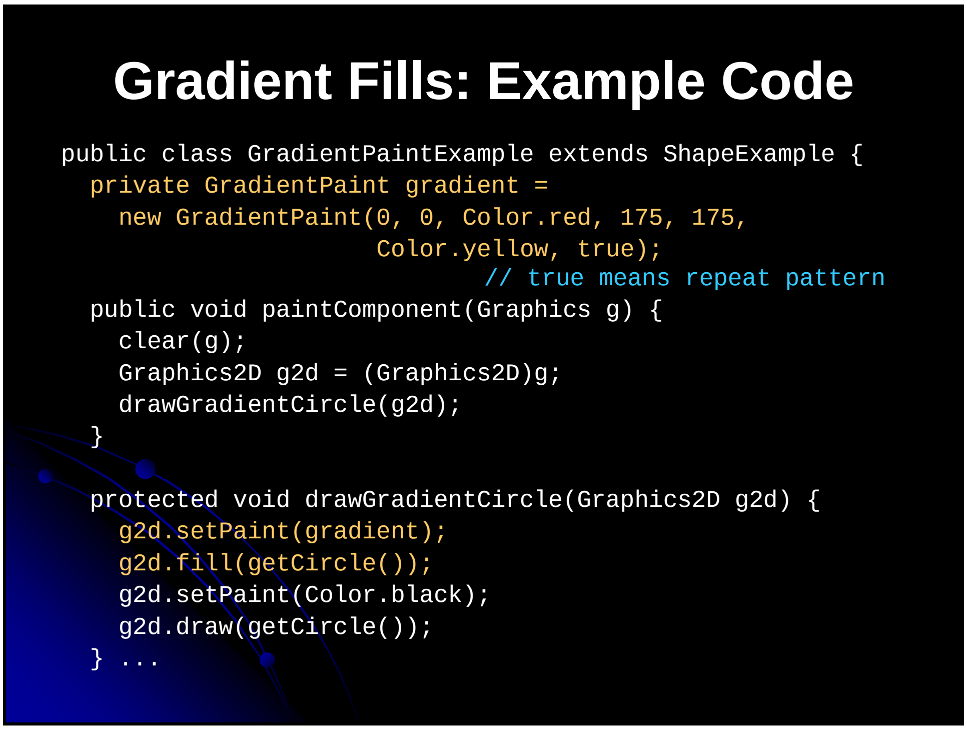
Paint Styles in Java 2D

- Use setPaint and getPaint to change and retrieve the Paint settings.
  - Note that setPaint and getPaint supersede the setColor and getColor methods that were used in Graphics (and inherited in Graphics2D).
- When you fill a Shape, the current Paint attribute of the Graphics2D object is used. Possible arguments to setPaint are:
  - A Color (solid color--Color implements Paint interface)
  - A GradientPaint (gradually-changing color combination)
  - A TexturePaint (tiled image)
  - A new version of Paint that you write yourself.



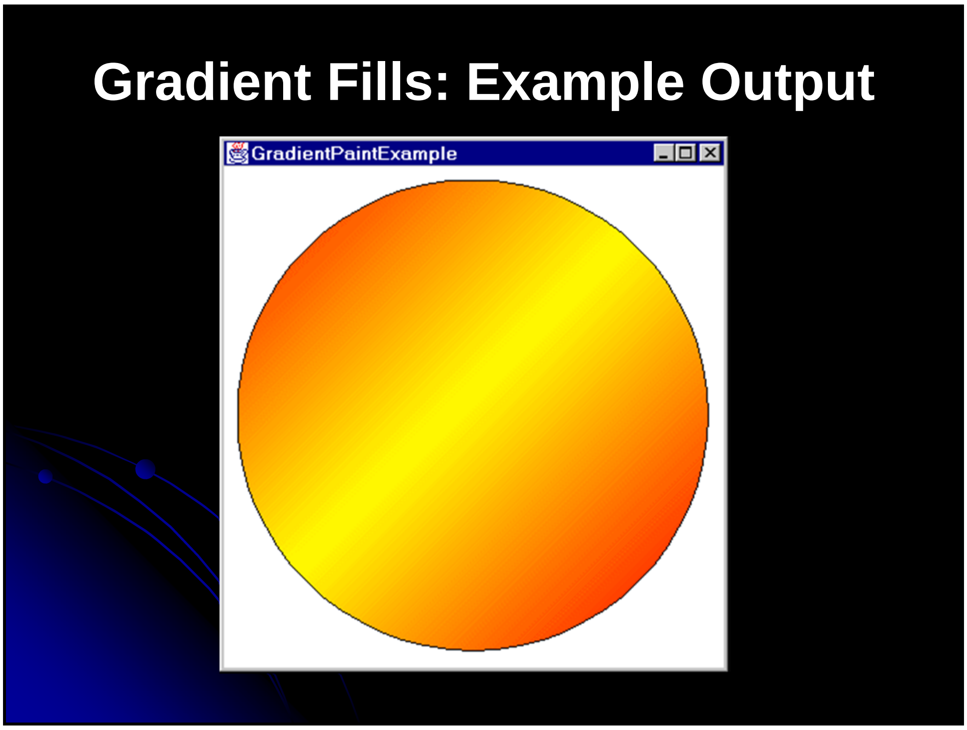
Paint Classes: Details

- Color
  - Has the same constants (Color.red, Color.yellow, etc.) as the AWT version, plus some extra constructors.
- GradientPaint
  - Constructors take two points, two colors, and optionally a boolean flag that indicates that the color pattern should cycle. Colors fade from one color to the other.
- TexturePaint
  - Constructor takes a BufferedImage and a Rectangle2D, maps the image to the rectangle, then tiles the rectangle.
    - Creating a BufferedImage from a GIF or JPEG file is tedious. First load an Image normally, get its size, create a BufferedImage that size with BufferedImage.TYPE_INT_ARGB as the image type, and get the BufferedImage's Graphics object via createGraphics. Then, draw the Image into the BufferedImage using drawImage.

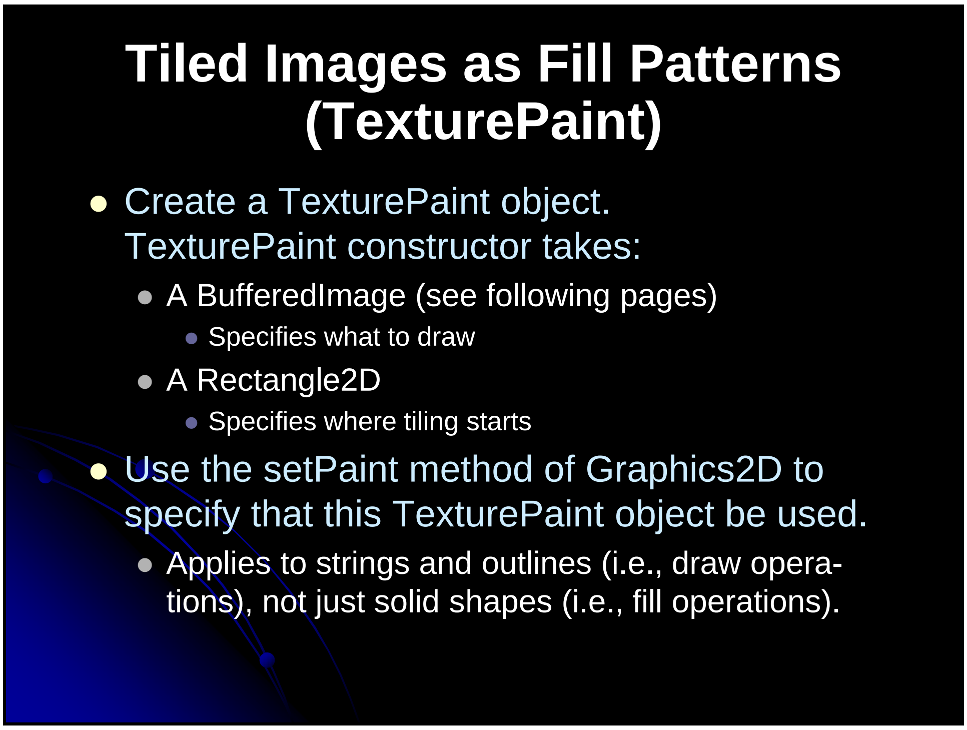


Gradient Fills: Example Code

```
public class GradientPaintExample extends ShapeExample {
  private GradientPaint gradient =
    new GradientPaint(0, 0, Color.red, 175, 175,
                      Color.yellow, true);
                              // true means repeat pattern
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
  }

  protected void drawGradientCircle(Graphics2D g2d) {
    g2d.setPaint(gradient);
    g2d.fill(getCircle());
    g2d.setPaint(Color.black);
    g2d.draw(getCircle());
  } ...
```

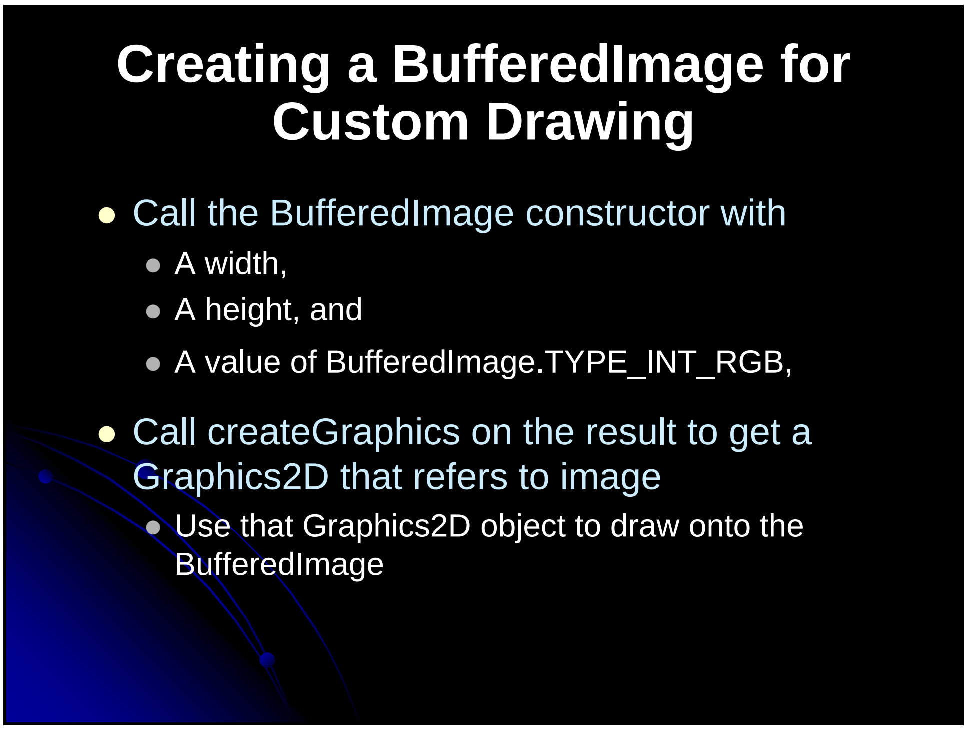


Gradient Fills: Example Output



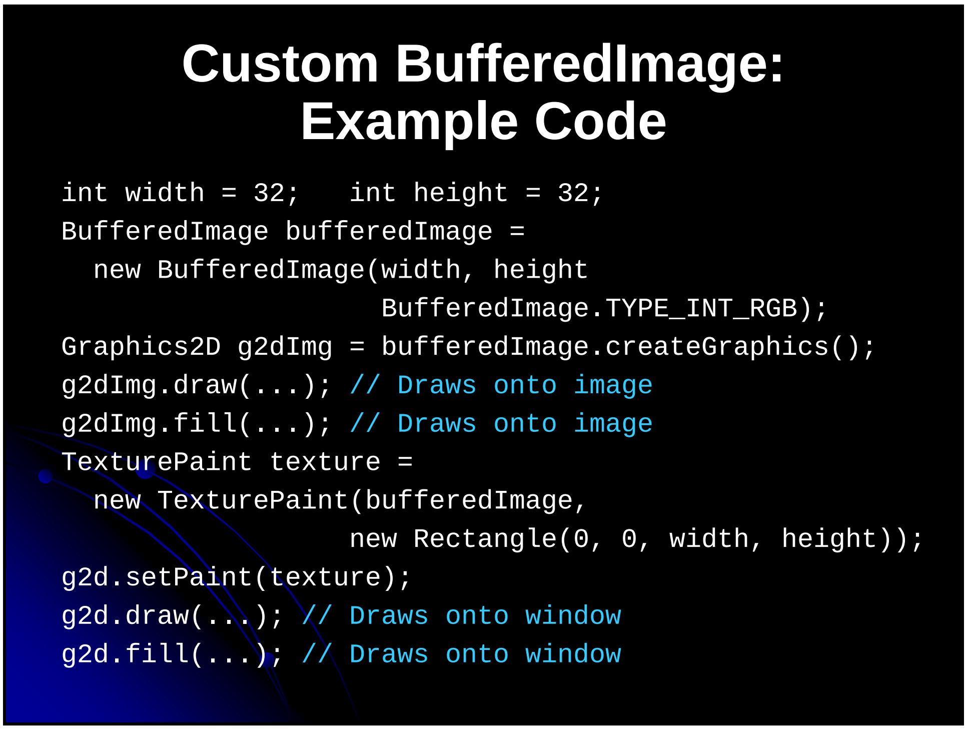
Tiled Images as Fill Patterns (TexturePaint)

- Create a TexturePaint object. TexturePaint constructor takes:
  - A BufferedImage (see following pages)
    - Specifies what to draw
  - A Rectangle2D
    - Specifies where tiling starts
- Use the setPaint method of Graphics2D to specify that this TexturePaint object be used.
  - Applies to strings and outlines (i.e., draw operations), not just solid shapes (i.e., fill operations).



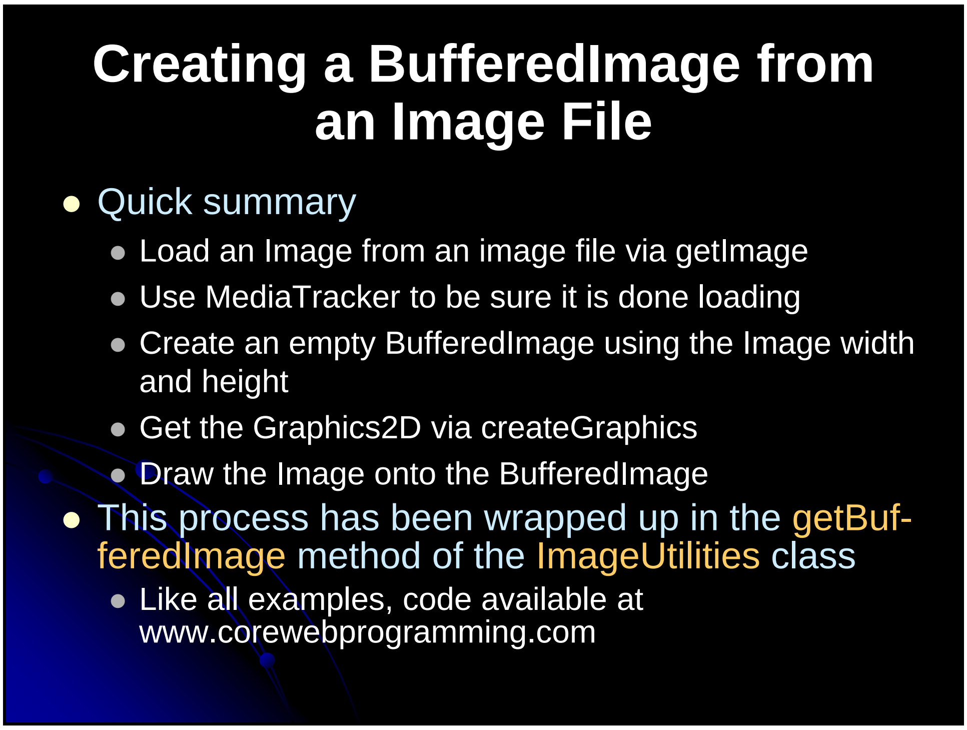
Creating a BufferedImage for Custom Drawing

- Call the BufferedImage constructor with
  - A width,
  - A height, and
  - A value of BufferedImage.TYPE_INT_RGB,
- Call createGraphics on the result to get a Graphics2D that refers to image
  - Use that Graphics2D object to draw onto the BufferedImage



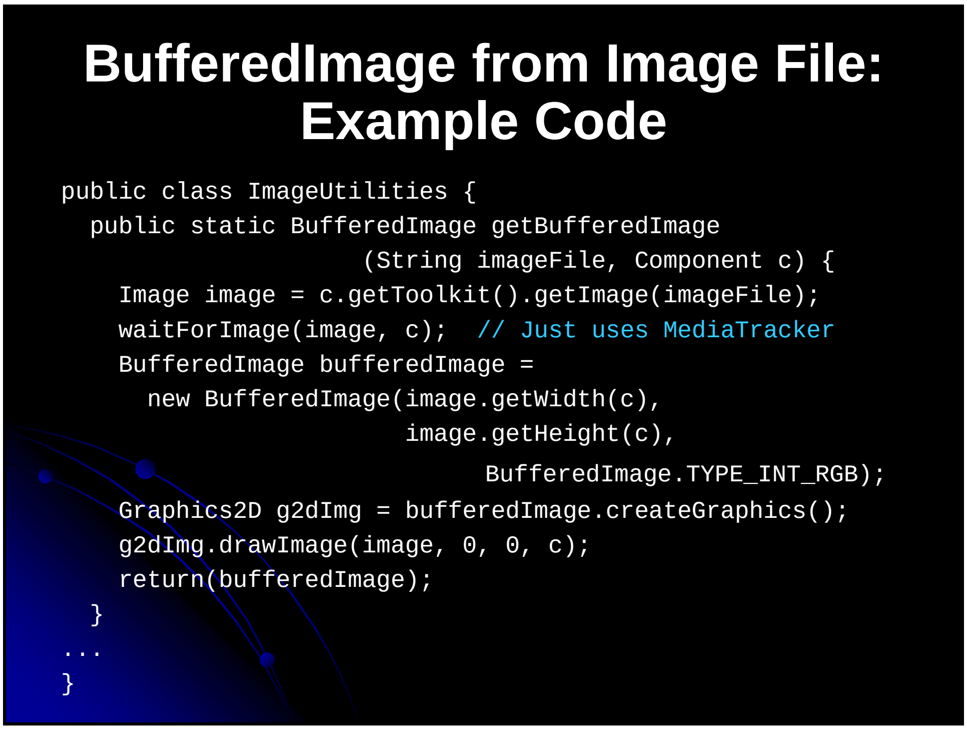
Custom BufferedImage: Example Code

```
int width = 32;   int height = 32;
BufferedImage bufferedImage =
  new BufferedImage(width, height
                    BufferedImage.TYPE_INT_RGB);
Graphics2D g2dImg = bufferedImage.createGraphics();
g2dImg.draw(...); // Draws onto image
g2dImg.fill(...); // Draws onto image
TexturePaint texture =
  new TexturePaint(bufferedImage,
                  new Rectangle(0, 0, width, height));
g2d.setPaint(texture);
g2d.draw(...); // Draws onto window
g2d.fill(...); // Draws onto window
```



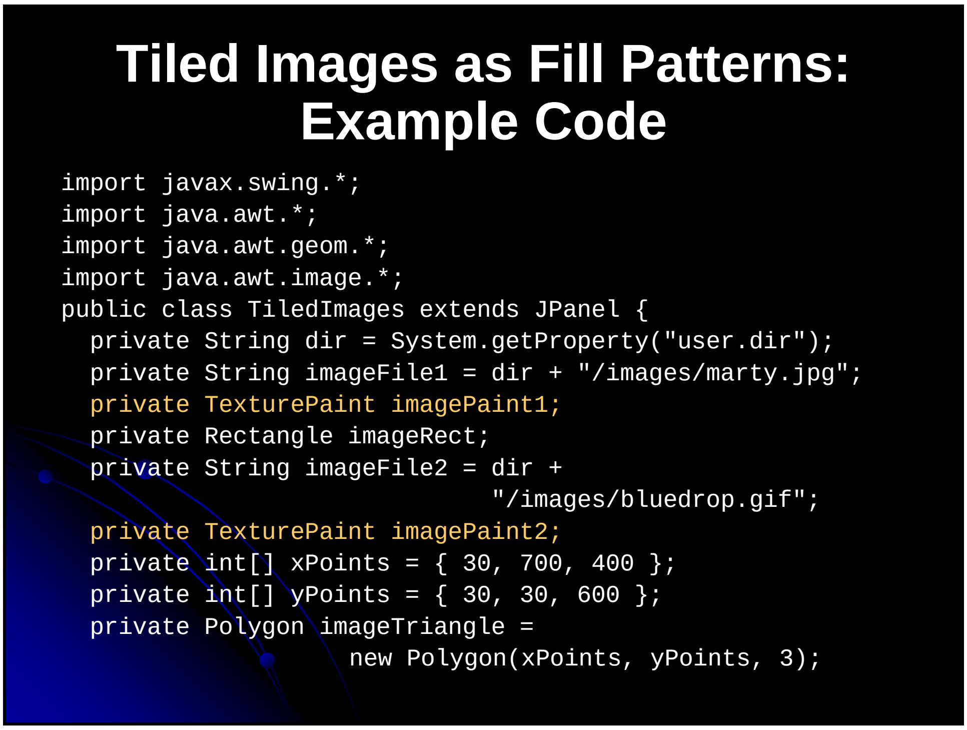
Creating a BufferedImage from an Image File

- Quick summary
  - Load an Image from an image file via getImage
  - Use MediaTracker to be sure it is done loading
  - Create an empty BufferedImage using the Image width and height
  - Get the Graphics2D via createGraphics
  - Draw the Image onto the BufferedImage
- This process has been wrapped up in the getBufferedImage method of the ImageUtilities class
  - Like all examples, code available at www.corewebprogramming.com



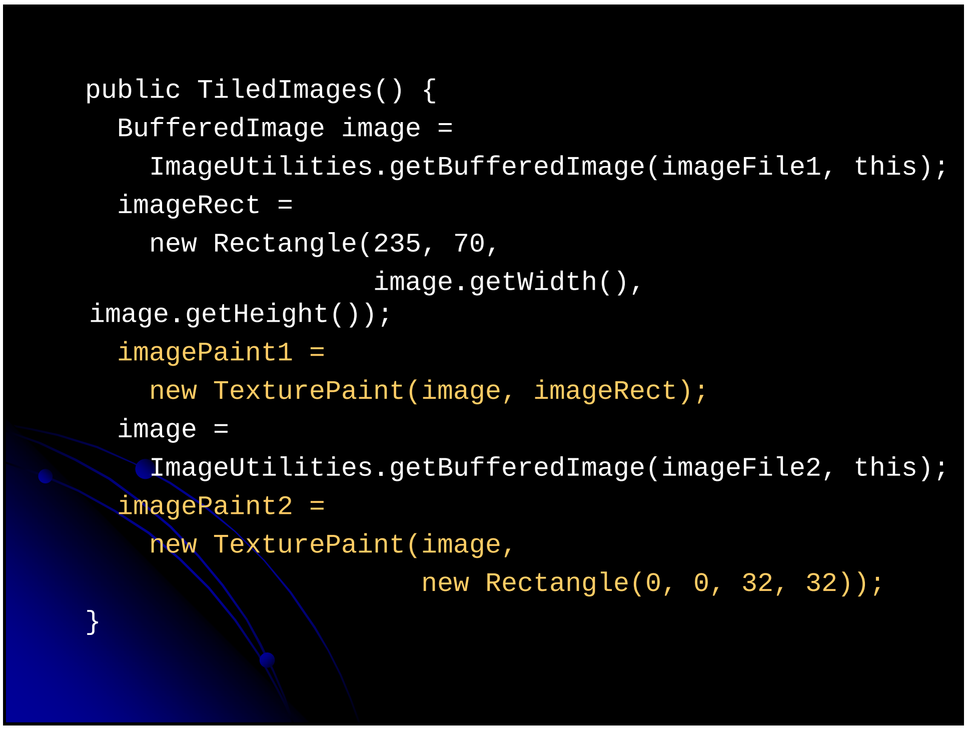
BufferedImage from Image File: Example Code

```
public class ImageUtilities {
  public static BufferedImage getBufferedImage
                   (String imageFile, Component c) {
    Image image = c.getToolkit().getImage(imageFile);
    waitForImage(image, c);  // Just uses MediaTracker
    BufferedImage bufferedImage =
      new BufferedImage(image.getWidth(c),
                        image.getHeight(c),
                              BufferedImage.TYPE_INT_RGB);
    Graphics2D g2dImg = bufferedImage.createGraphics();
    g2dImg.drawImage(image, 0, 0, c);
    return(bufferedImage);
  }
...
}
```

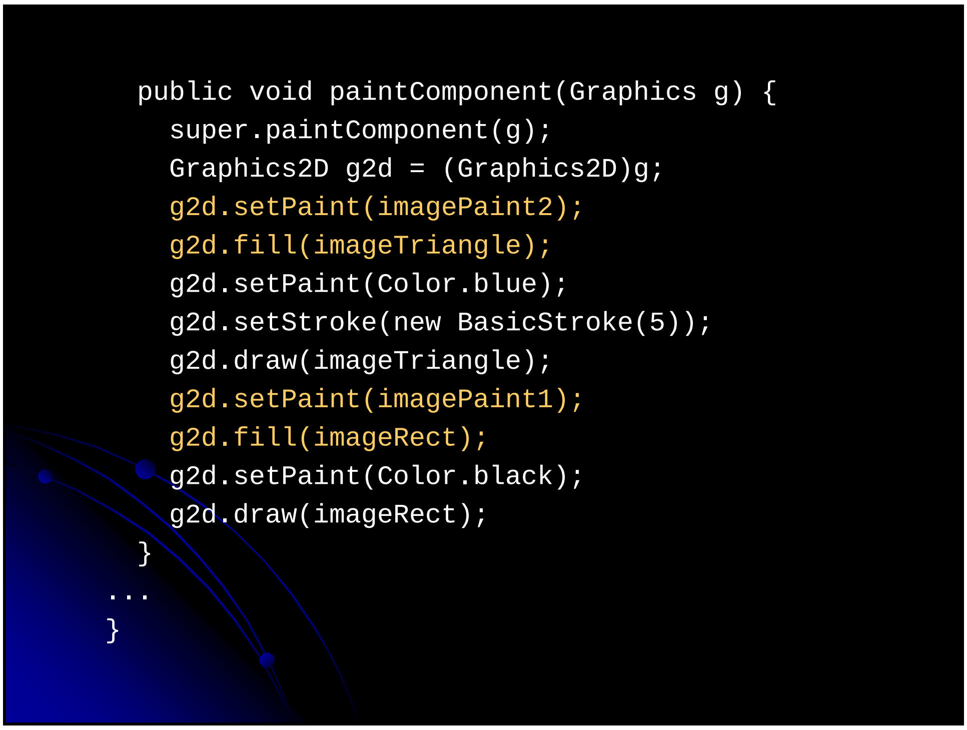


Tiled Images as Fill Patterns: Example Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
import java.awt.image.*;
public class TiledImages extends JPanel {
  private String dir = System.getProperty("user.dir");
  private String imageFile1 = dir + "/images/marty.jpg";
  private TexturePaint imagePaint1;
  private Rectangle imageRect;
  private String imageFile2 = dir +
                            "/images/bluedrop.gif";
  private TexturePaint imagePaint2;
  private int[] xPoints = { 30, 700, 400 };
  private int[] yPoints = { 30, 30, 600 };
  private Polygon imageTriangle =
                    new Polygon(xPoints, yPoints, 3);
```

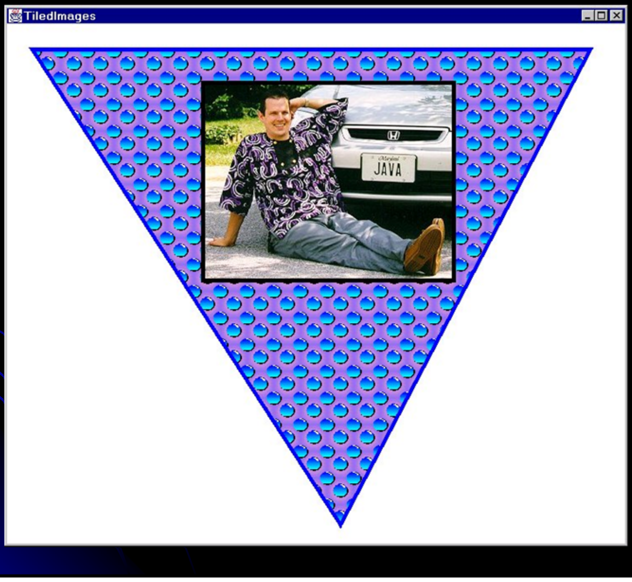


```
public TiledImages() {
  BufferedImage image =
    ImageUtilities.getBufferedImage(imageFile1, this);
  imageRect =
    new Rectangle(235, 70,
                  image.getWidth(),
image.getHeight());
  imagePaint1 =
    new TexturePaint(image, imageRect);
  image =
    ImageUtilities.getBufferedImage(imageFile2, this);
  imagePaint2 =
    new TexturePaint(image,
                     new Rectangle(0, 0, 32, 32));
}
```



```
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    g2d.setPaint(imagePaint2);
    g2d.fill(imageTriangle);
    g2d.setPaint(Color.blue);
    g2d.setStroke(new BasicStroke(5));
    g2d.draw(imageTriangle);
    g2d.setPaint(imagePaint1);
    g2d.fill(imageRect);
    g2d.setPaint(Color.black);
    g2d.draw(imageRect);
  }
...
}
```

Tiled Images as Fill Patterns: Example Output



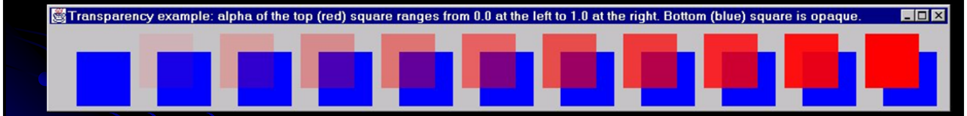
Transparent Drawing: Overview

- Idea
  - Assign transparency (alpha) values to drawing operations so that the underlying graphics partially shows through when you draw shapes or images.
- Normal steps
  - Create an AlphaComposite object
    - Call AlphaComposite.getInstance with a mixing rule designator and a transparency (or "alpha") value.
    - There are 8 built-in mixing rules (see the AlphaComposite API for details), but you only care about AlphaComposite.SRC_OVER.
    - Alpha values range from 0.0F (completely transparent) to 1.0F (completely opaque).
  - Pass the AlphaComposite object to the setComposite method of the Graphics2D

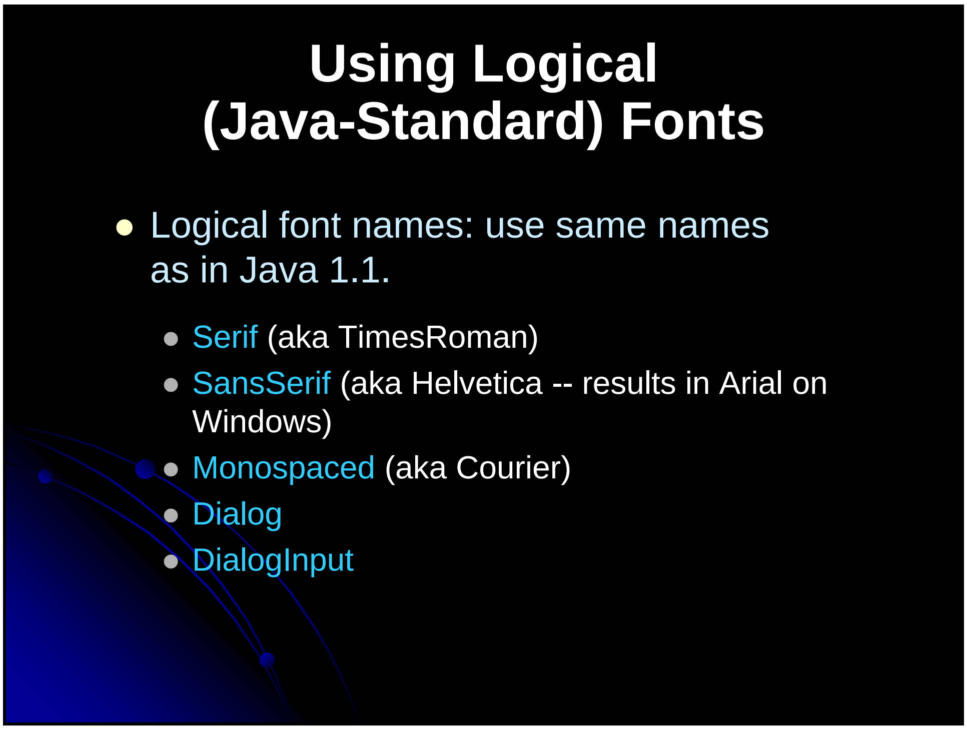
Transparent Drawing: Example Code

```
public class TransparencyExample extends JPanel {
  ...
  private AlphaComposite makeComposite(float alpha) {
    int type = AlphaComposite.SRC_OVER;
    return(AlphaComposite.getInstance(type, alpha));
  }
  private void drawSquares(Graphics2D g2d, float alpha) {
    Composite originalComposite = g2d.getComposite();
    g2d.setPaint(Color.blue);
    g2d.fill(blueSquare);
    g2d.setComposite(makeComposite(alpha));
    g2d.setPaint(Color.red);
    g2d.fill(redSquare);
    g2d.setComposite(originalComposite);
  }
...
```

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2d = (Graphics2D)g;
  for(int i=0; i<11; i++) {
    drawSquares(g2d, i*0.1F); // 2nd arg is transparency
    g2d.translate(deltaX, 0);
}
```

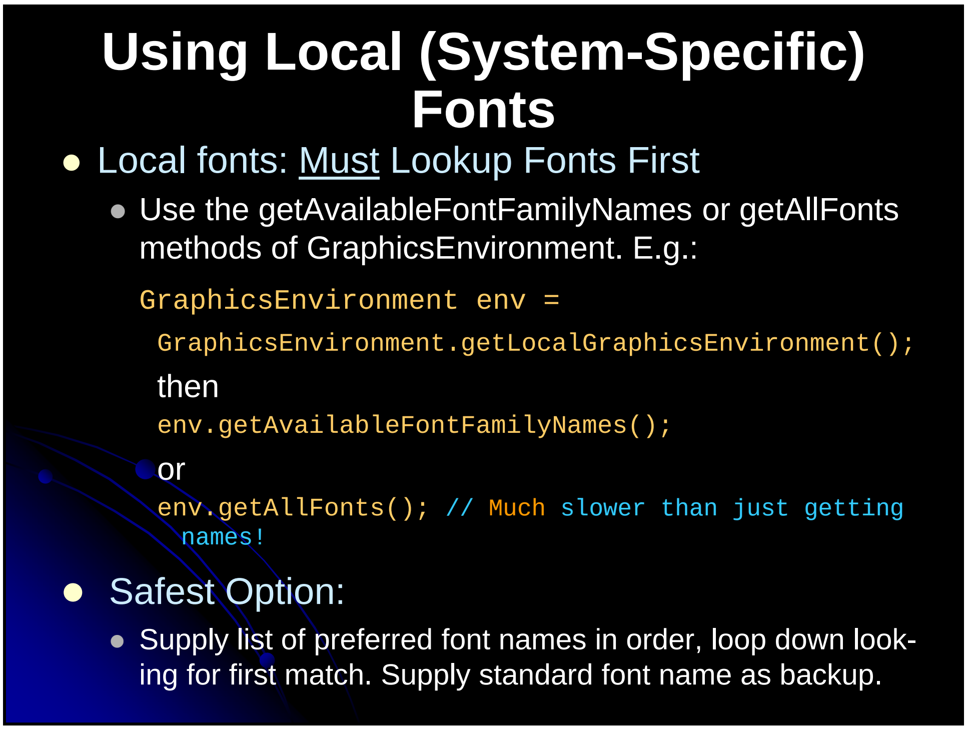


Transparency example: alpha of the top (red) square ranges from 0.0 at the left to 1.0 at the right. Bottom (blue) square is opaque.



Using Logical (Java-Standard) Fonts

- Logical font names: use same names as in Java 1.1.
  - Serif (aka TimesRoman)
  - SansSerif (aka Helvetica -- results in Arial on Windows)
  - Monospaced (aka Courier)
  - Dialog
  - DialogInput



Using Local (System-Specific) Fonts

- Local fonts: Must Lookup Fonts First
  - Use the getAvailableFontFamilyNames or getAllFonts methods of GraphicsEnvironment. E.g.:

```
GraphicsEnvironment env =
  GraphicsEnvironment.getLocalGraphicsEnvironment();
```

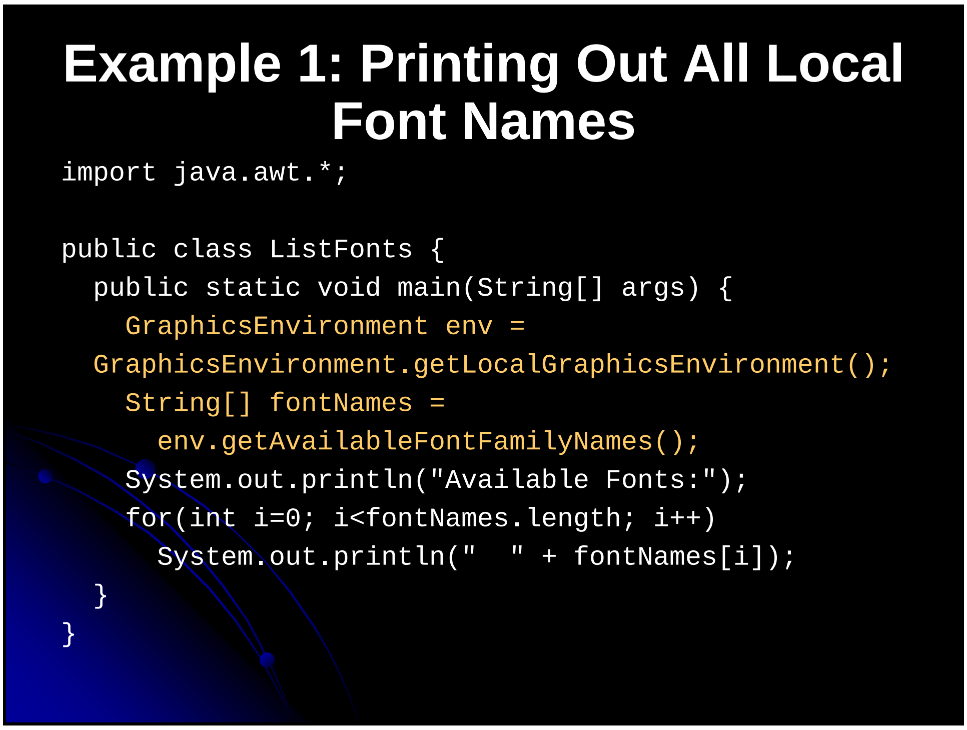
then

```
env.getAvailableFontFamilyNames();
```

or

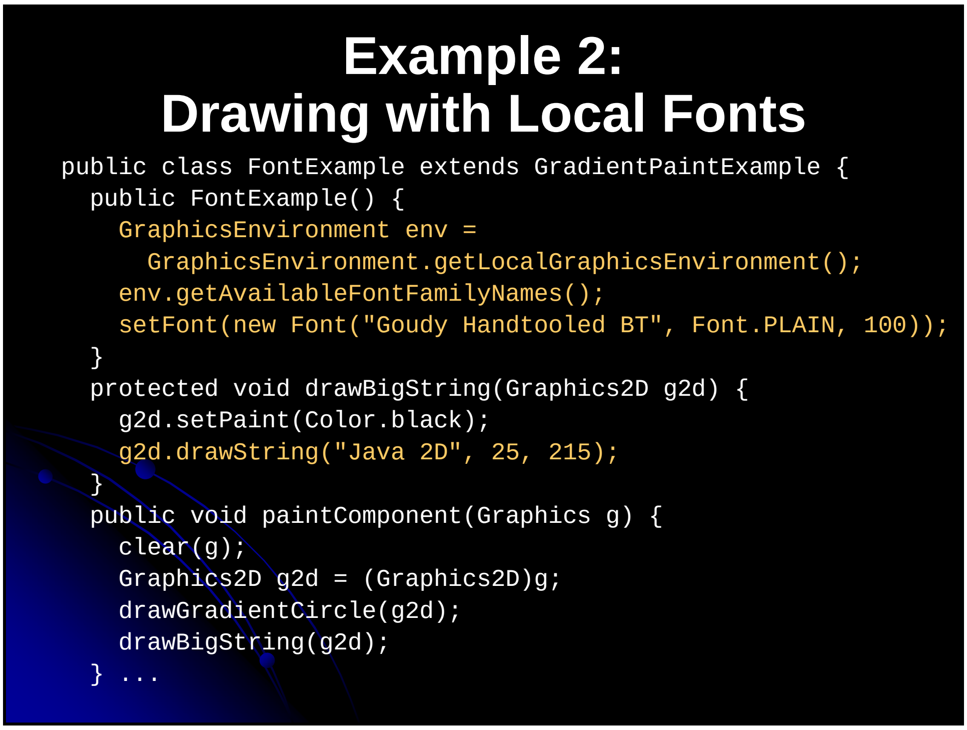
```
env.getAllFonts(); // Much slower than just getting
  names!
```

- Safest Option:
  - Supply list of preferred font names in order, loop down looking for first match. Supply standard font name as backup.



Example 1: Printing Out All Local Font Names

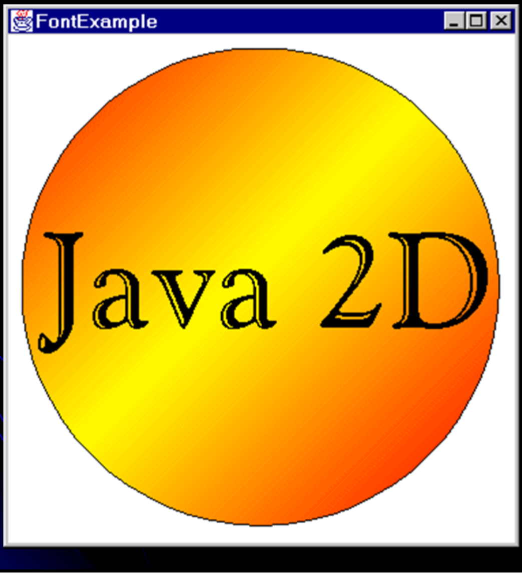
```
import java.awt.*;

public class ListFonts {
  public static void main(String[] args) {
    GraphicsEnvironment env =
  GraphicsEnvironment.getLocalGraphicsEnvironment();
    String[] fontNames =
      env.getAvailableFontFamilyNames();
    System.out.println("Available Fonts:");
    for(int i=0; i<fontNames.length; i++)
      System.out.println("  " + fontNames[i]);
  }
}
```



Example 2: Drawing with Local Fonts

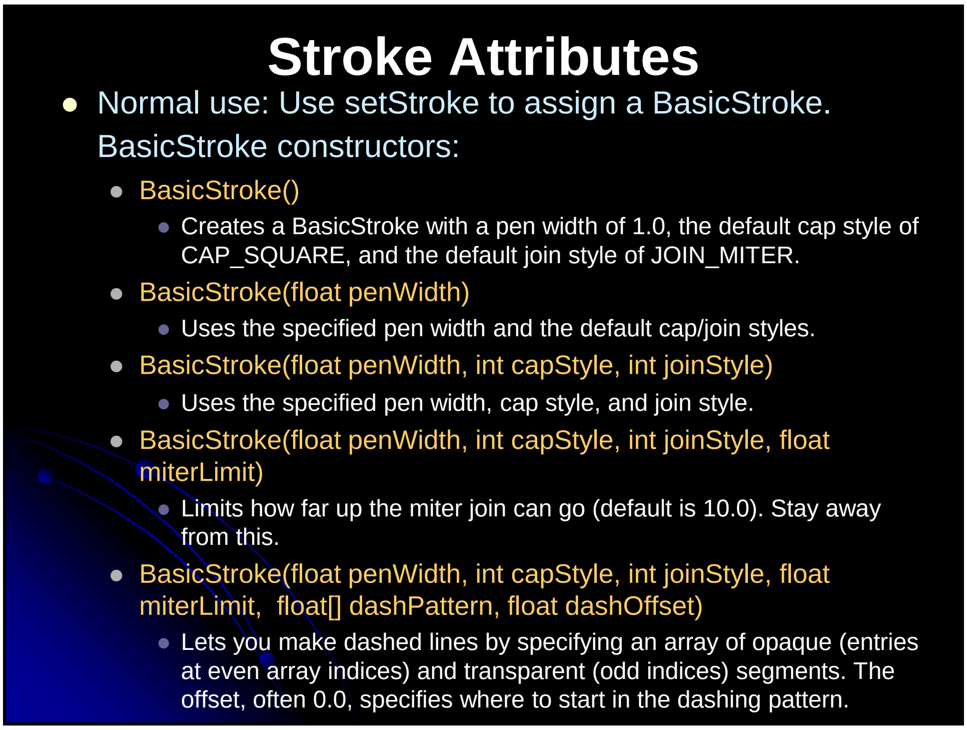
```
public class FontExample extends GradientPaintExample {
  public FontExample() {
    GraphicsEnvironment env =
      GraphicsEnvironment.getLocalGraphicsEnvironment();
    env.getAvailableFontFamilyNames();
    setFont(new Font("Goudy Handtooled BT", Font.PLAIN, 100));
  }
  protected void drawBigString(Graphics2D g2d) {
    g2d.setPaint(Color.black);
    g2d.drawString("Java 2D", 25, 215);
  }
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawBigString(g2d);
  } ...
```

Drawing with Local Fonts: Example Output



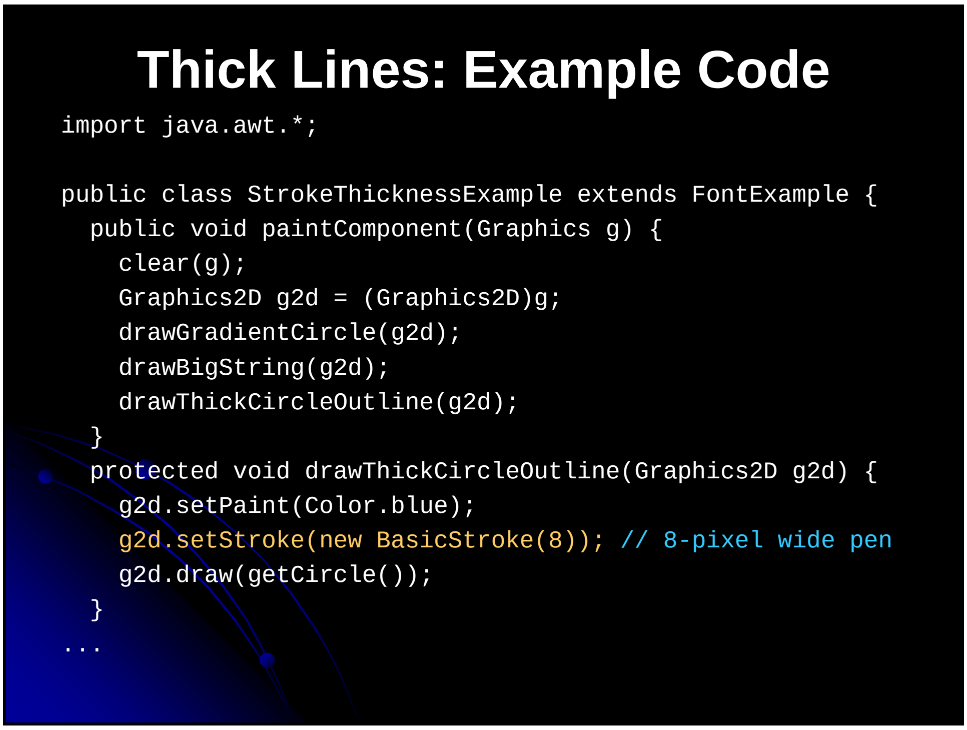
Stroke Styles: Overview

- AWT
  - draw*Xxx* methods of Graphics resulted in solid, 1-pixel wide lines.
  - Predefined line join/cap styles for drawRect, drawPolygon, etc.
- Java2D
  - Pen thickness
  - Dashing pattern
  - Line join/cap styles
- Setting styles
  - Create a BasicStroke object
  - Use the setStroke method to tell the Graphics2D object to use it



Stroke Attributes

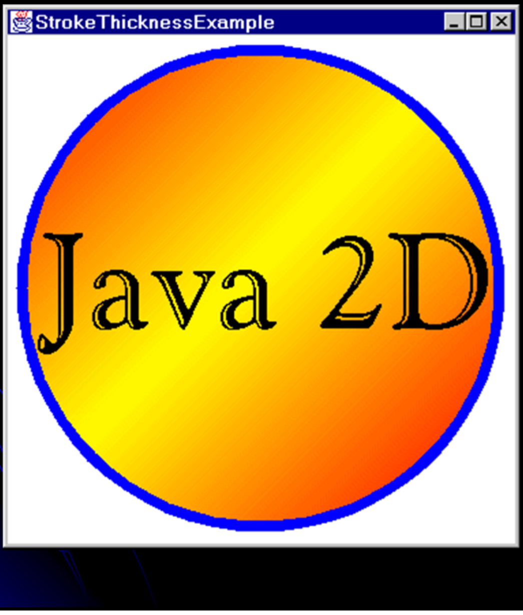
- Normal use: Use setStroke to assign a BasicStroke. BasicStroke constructors:
  - BasicStroke()
    - Creates a BasicStroke with a pen width of 1.0, the default cap style of CAP_SQUARE, and the default join style of JOIN_MITER.
  - BasicStroke(float penWidth)
    - Uses the specified pen width and the default cap/join styles.
  - BasicStroke(float penWidth, int capStyle, int joinStyle)
    - Uses the specified pen width, cap style, and join style.
  - BasicStroke(float penWidth, int capStyle, int joinStyle, float miterLimit)
    - Limits how far up the miter join can go (default is 10.0). Stay away from this.
  - BasicStroke(float penWidth, int capStyle, int joinStyle, float miterLimit, float[] dashPattern, float dashOffset)
    - Lets you make dashed lines by specifying an array of opaque (entries at even array indices) and transparent (odd indices) segments. The offset, often 0.0, specifies where to start in the dashing pattern.

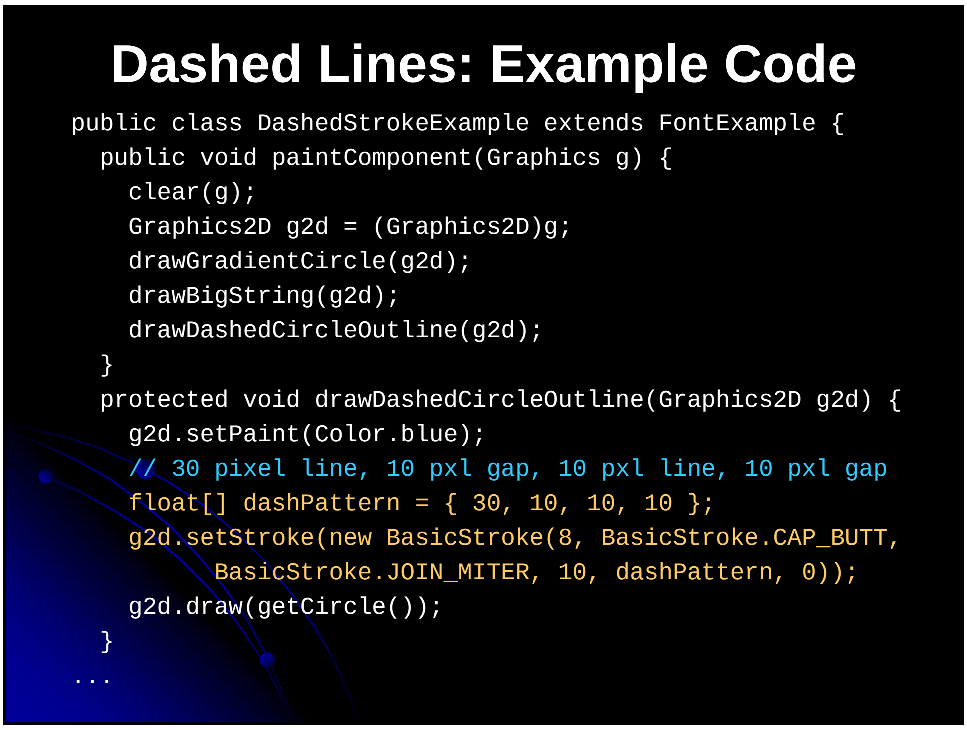


Thick Lines: Example Code

```
import java.awt.*;

public class StrokeThicknessExample extends FontExample {
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawBigString(g2d);
    drawThickCircleOutline(g2d);
  }
  protected void drawThickCircleOutline(Graphics2D g2d) {
    g2d.setPaint(Color.blue);
    g2d.setStroke(new BasicStroke(8)); // 8-pixel wide pen
    g2d.draw(getCircle());
  }
...
```

Thick Lines: Example Output

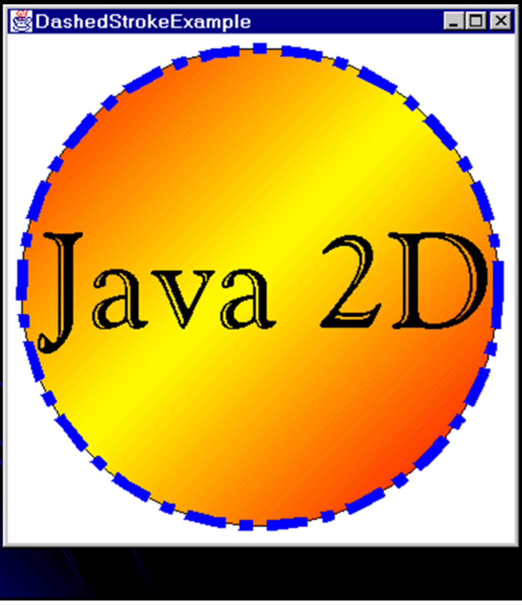




Dashed Lines: Example Code

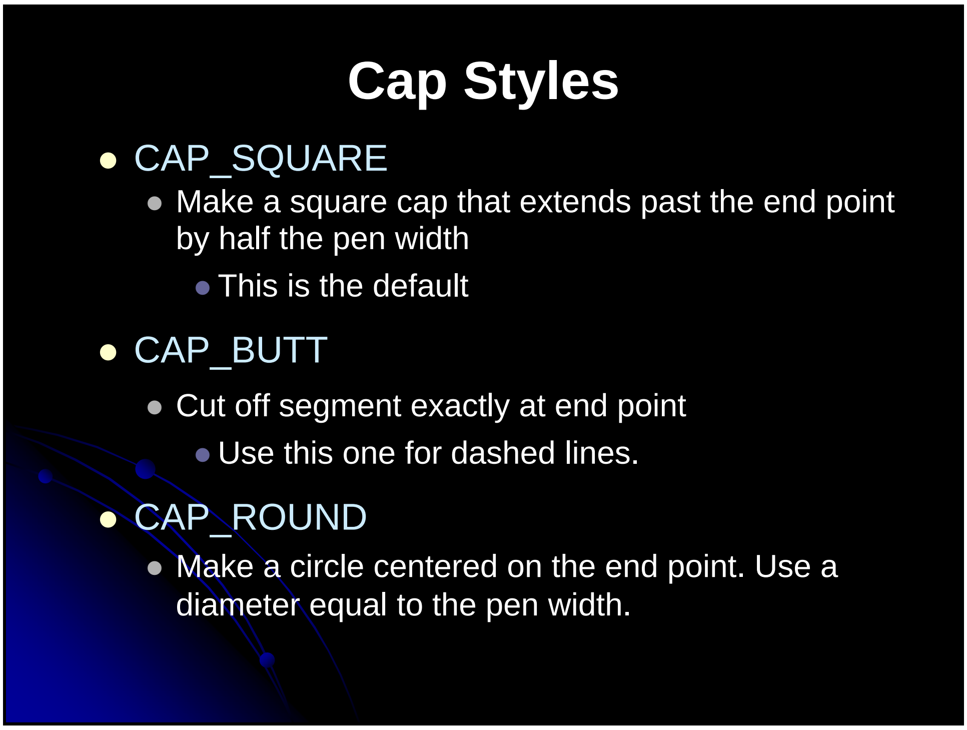
```
public class DashedStrokeExample extends FontExample {
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawBigString(g2d);
    drawDashedCircleOutline(g2d);
  }
  protected void drawDashedCircleOutline(Graphics2D g2d) {
    g2d.setPaint(Color.blue);
    // 30 pixel line, 10 pxl gap, 10 pxl line, 10 pxl gap
    float[] dashPattern = { 30, 10, 10, 10 };
    g2d.setStroke(new BasicStroke(8, BasicStroke.CAP_BUTT,
          BasicStroke.JOIN_MITER, 10, dashPattern, 0));
    g2d.draw(getCircle());
  }
...
```

Dashed Lines: Example Output



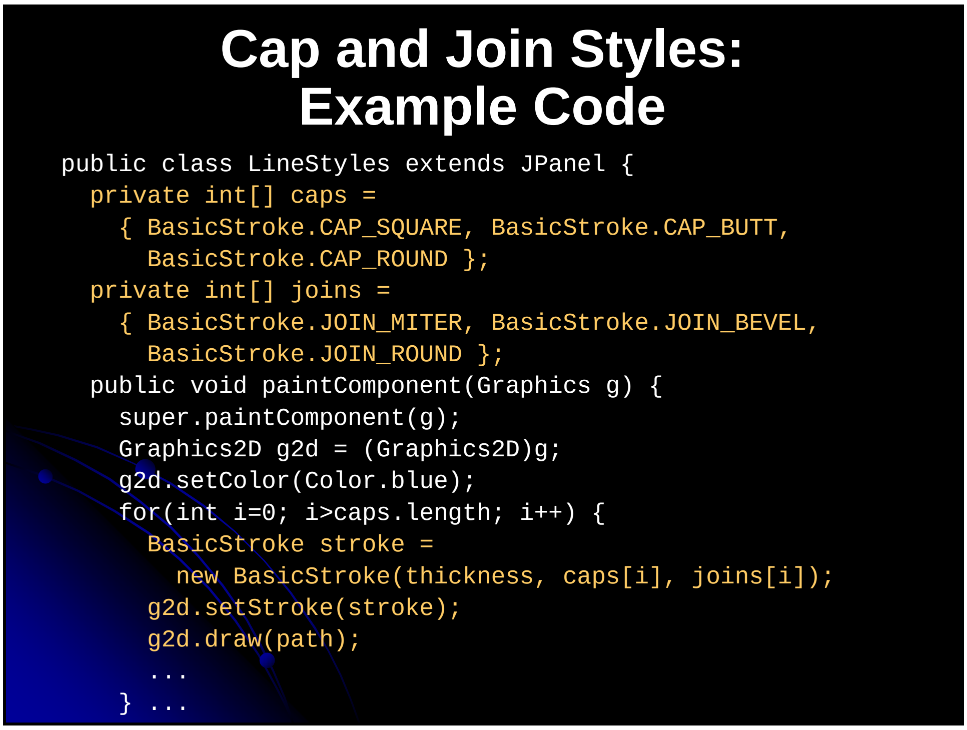
Join Styles

- JOIN_MITER
  - Extend outside edges of lines until they meet
    - This is the default
- JOIN_BEVEL
  - Connect outside corners of outlines with straight line
- JOIN_ROUND
  - Round off corner with a circle that has diameter equal to the pen width



Cap Styles

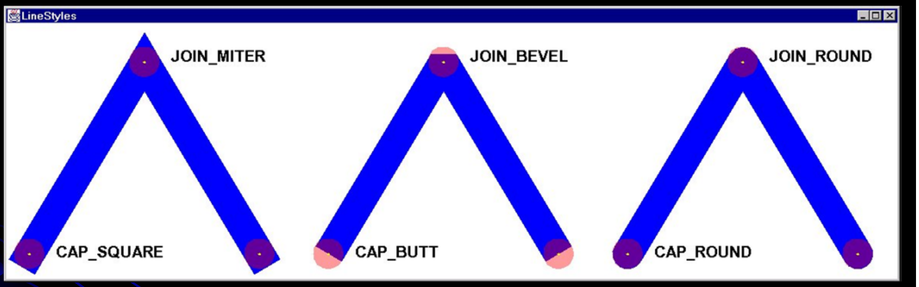
- CAP_SQUARE
  - Make a square cap that extends past the end point by half the pen width
    - This is the default
- CAP_BUTT
  - Cut off segment exactly at end point
    - Use this one for dashed lines.
- CAP_ROUND
  - Make a circle centered on the end point. Use a diameter equal to the pen width.



Cap and Join Styles: Example Code

```
public class LineStyles extends JPanel {
  private int[] caps =
    { BasicStroke.CAP_SQUARE, BasicStroke.CAP_BUTT,
      BasicStroke.CAP_ROUND };
  private int[] joins =
    { BasicStroke.JOIN_MITER, BasicStroke.JOIN_BEVEL,
      BasicStroke.JOIN_ROUND };
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    g2d.setColor(Color.blue);
    for(int i=0; i>caps.length; i++) {
      BasicStroke stroke =
        new BasicStroke(thickness, caps[i], joins[i]);
      g2d.setStroke(stroke);
      g2d.draw(path);
      ...
    } ...
```

Cap and Join Styles: Example Output



Coordinate Transformations

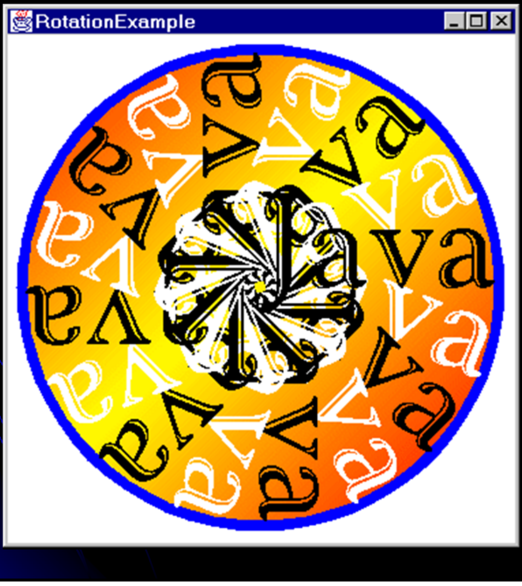
- Idea:
  - Instead of computing new coordinates, move the coordinate system itself
- Available Transformations
  - Translate (move)
  - Rotate (spin)
  - Scale (stretch evenly)
  - Shear (stretch more as points get further from origin)
  - Custom. New point (x2, y2) derived from original point (x1, y1) as follows:

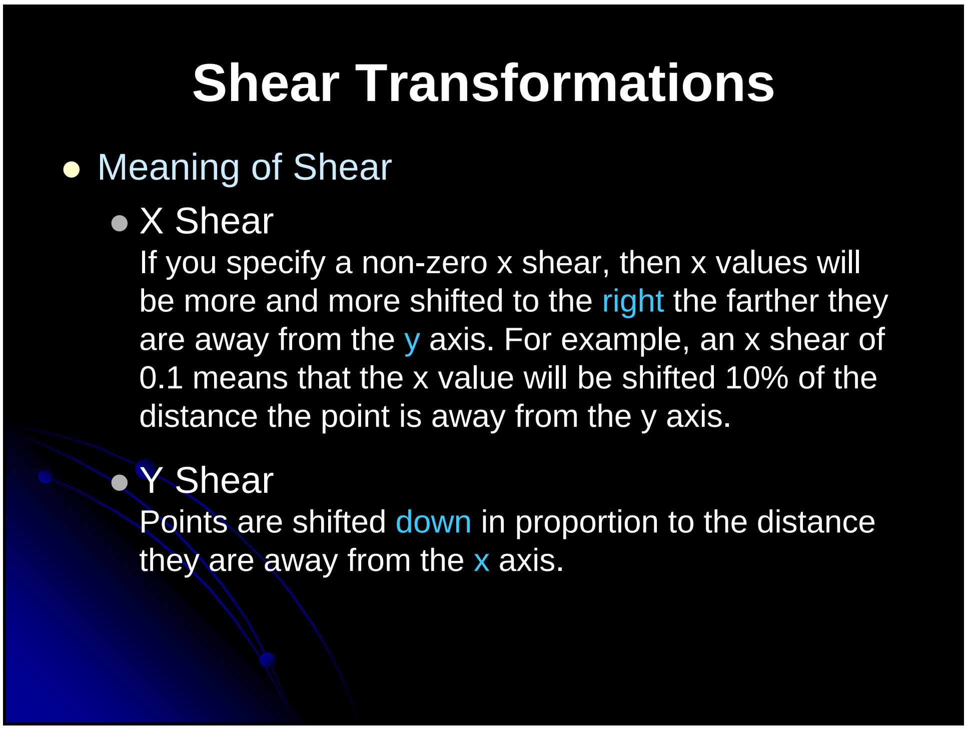
```
[ x2]   [ m00  m01  m02 ] [x1]   [ m00x1 + m01y1 + m02 ]
[ y2] = [ m10  m11  m12 ] [y1] = [ m10x1 + m11y1 + m12 ]
[ 1 ]   [  0    0    1  ] [1 ]   [          1          ]
```

Translations and Rotations: Example Code

```
public class RotationExample extends StrokeThicknessExample {
  private Color[] colors = { Color.white, Color.black };
  public void paintComponent(Graphics g) {
    clear(g);
    Graphics2D g2d = (Graphics2D)g;
    drawGradientCircle(g2d);
    drawThickCircleOutline(g2d);
    // Move the origin to the center of the circle.
    g2d.translate(185.0, 185.0);
    for (int i=0; i<16; i++) {
      // Rotate the coordinate system around current
      // origin, which is at the center of the circle.
      g2d.rotate(Math.PI/8.0);
      g2d.setPaint(colors[i%2]);
      g2d.drawString("Java", 0, 0);
    } ...
```

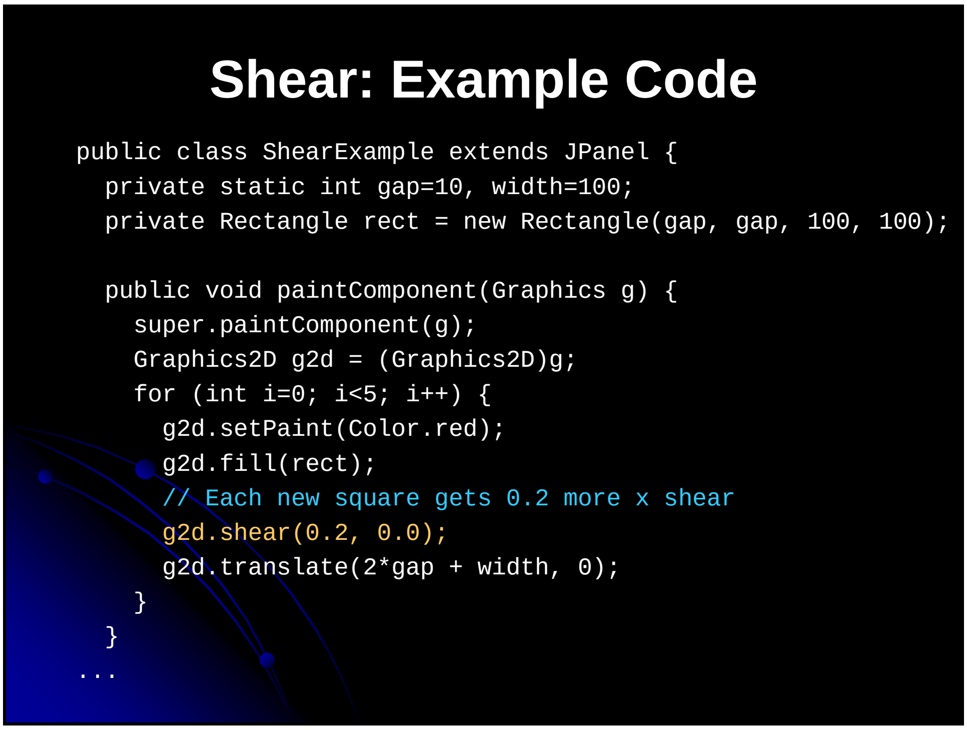
Translations and Rotations: Example Output





Shear Transformations

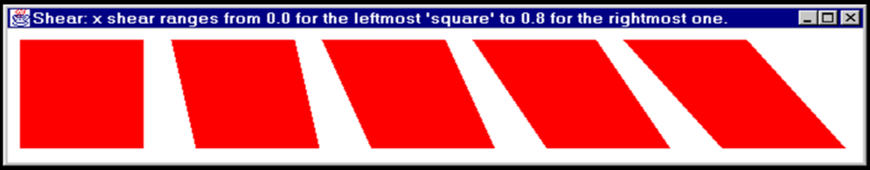
- Meaning of Shear
  - X Shear
    If you specify a non-zero x shear, then x values will be more and more shifted to the right the farther they are away from the y axis. For example, an x shear of 0.1 means that the x value will be shifted 10% of the distance the point is away from the y axis.
  - Y Shear
    Points are shifted down in proportion to the distance they are away from the x axis.



Shear: Example Code

```
public class ShearExample extends JPanel {
  private static int gap=10, width=100;
  private Rectangle rect = new Rectangle(gap, gap, 100, 100);

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d = (Graphics2D)g;
    for (int i=0; i<5; i++) {
      g2d.setPaint(Color.red);
      g2d.fill(rect);
      // Each new square gets 0.2 more x shear
      g2d.shear(0.2, 0.0);
      g2d.translate(2*gap + width, 0);
    }
  }
...
```

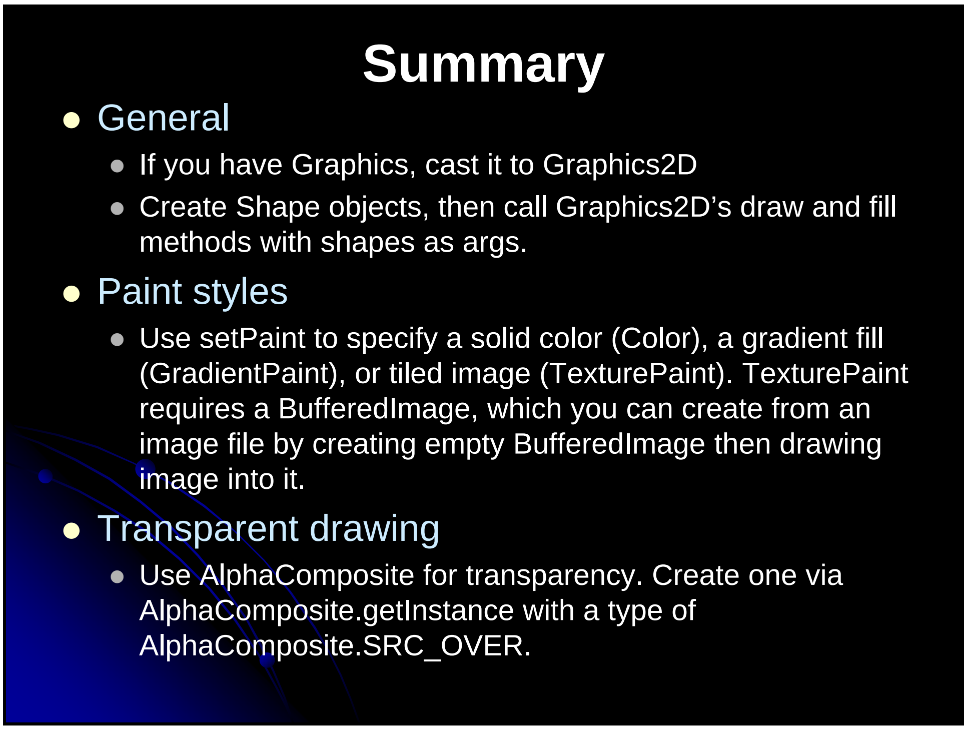
Shear: Example Output


Shear: x shear ranges from 0.0 for the leftmost 'square' to 0.8 for the rightmost one.

Rendering Hints

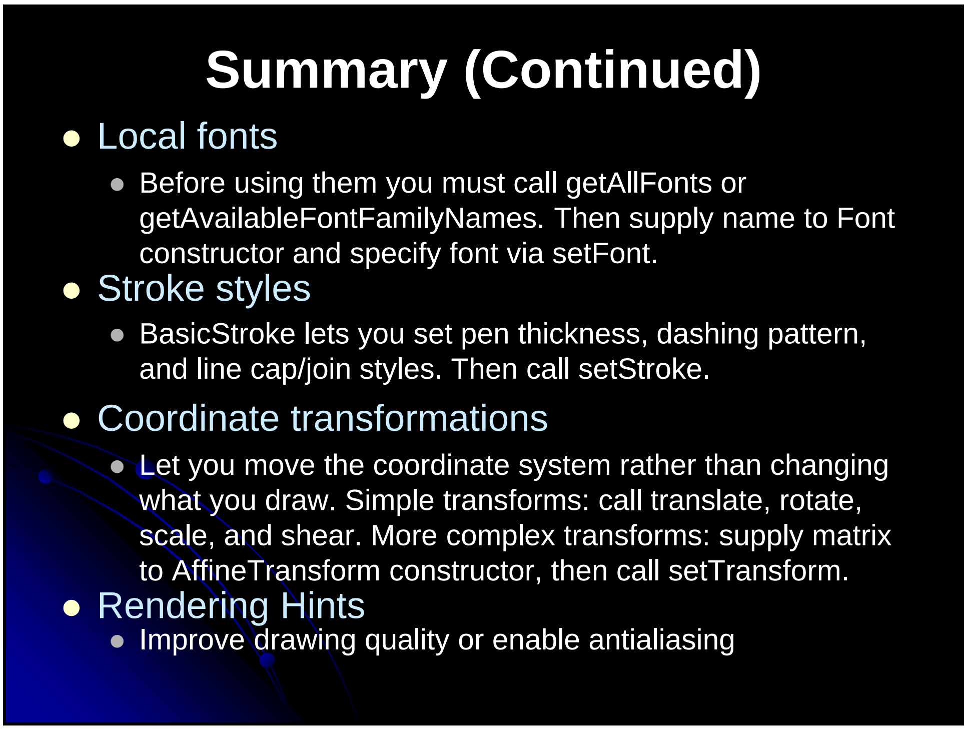
- Default:
  - Faster drawing, possibly less accuracy
- Rendering Hints:
  - Let you request more accurate (but generally slower) drawing. Eg:

```
RenderingHints renderHints =
  new RenderingHints(RenderingHints.KEY_ANTIALIASING,

  RenderingHints.VALUE_ANTIALIAS_ON);
renderHints.put(RenderingHints.KEY_RENDERING,
                RenderingHints.VALUE_RENDER_QUALITY);
...
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  Graphics2D g2d = (Graphics2D)g;
  g2d.setRenderingHints(renderHints);
  ...
```



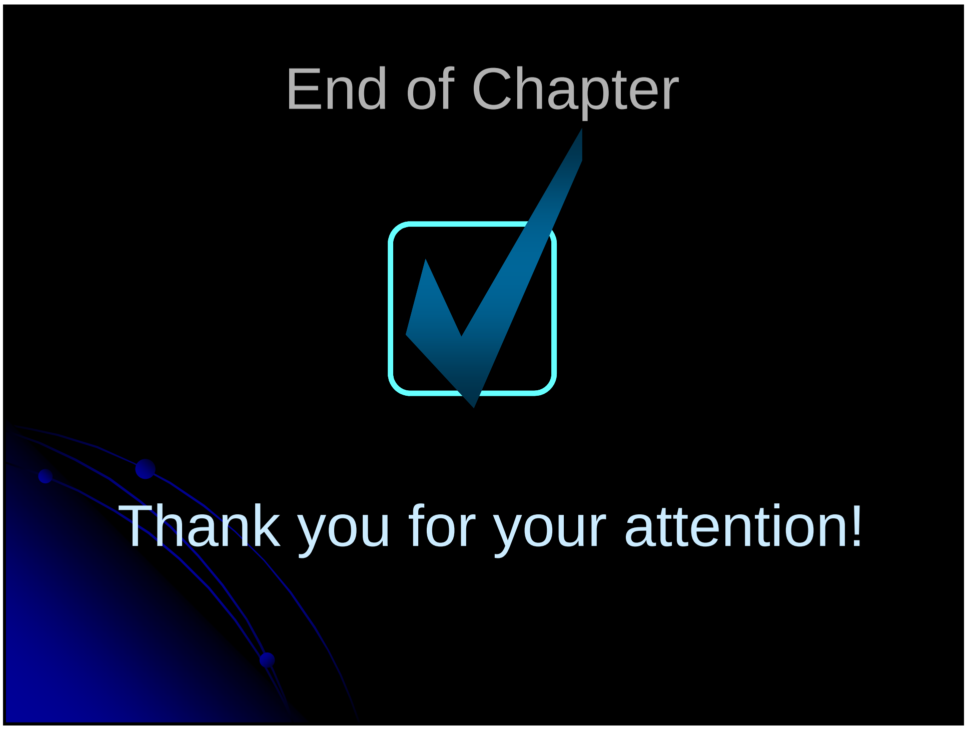
Summary

- General
  - If you have Graphics, cast it to Graphics2D
  - Create Shape objects, then call Graphics2D’s draw and fill methods with shapes as args.
- Paint styles
  - Use setPaint to specify a solid color (Color), a gradient fill (GradientPaint), or tiled image (TexturePaint). TexturePaint requires a BufferedImage, which you can create from an image file by creating empty BufferedImage then drawing image into it.
- Transparent drawing
  - Use AlphaComposite for transparency. Create one via AlphaComposite.getInstance with a type of AlphaComposite.SRC_OVER.



Summary (Continued)

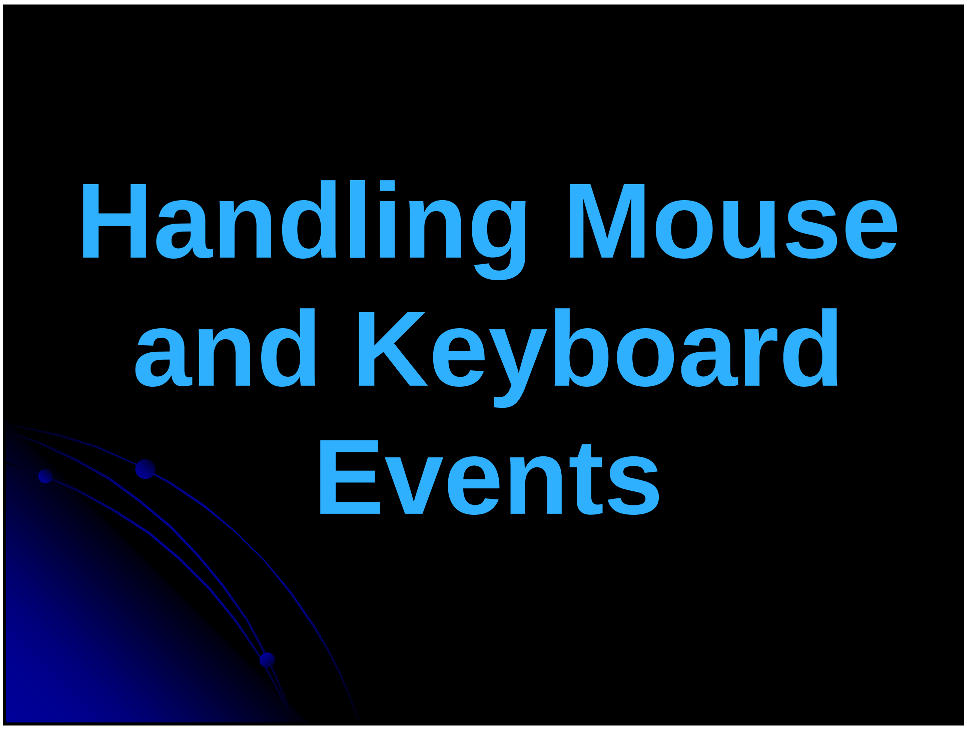
- Local fonts
  - Before using them you must call getAllFonts or getAvailableFontFamilyNames. Then supply name to Font constructor and specify font via setFont.
- Stroke styles
  - BasicStroke lets you set pen thickness, dashing pattern, and line cap/join styles. Then call setStroke.
- Coordinate transformations
  - Let you move the coordinate system rather than changing what you draw. Simple transforms: call translate, rotate, scale, and shear. More complex transforms: supply matrix to AffineTransform constructor, then call setTransform.
- Rendering Hints
  - Improve drawing quality or enable antialiasing



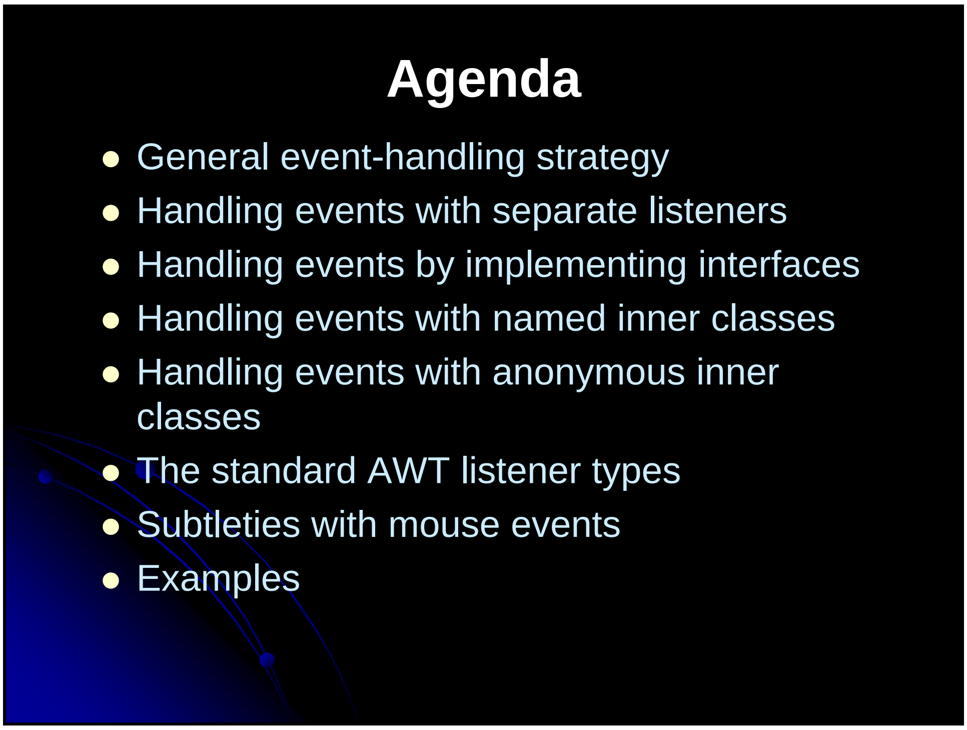
End of Chapter

☑

Thank you for your attention!

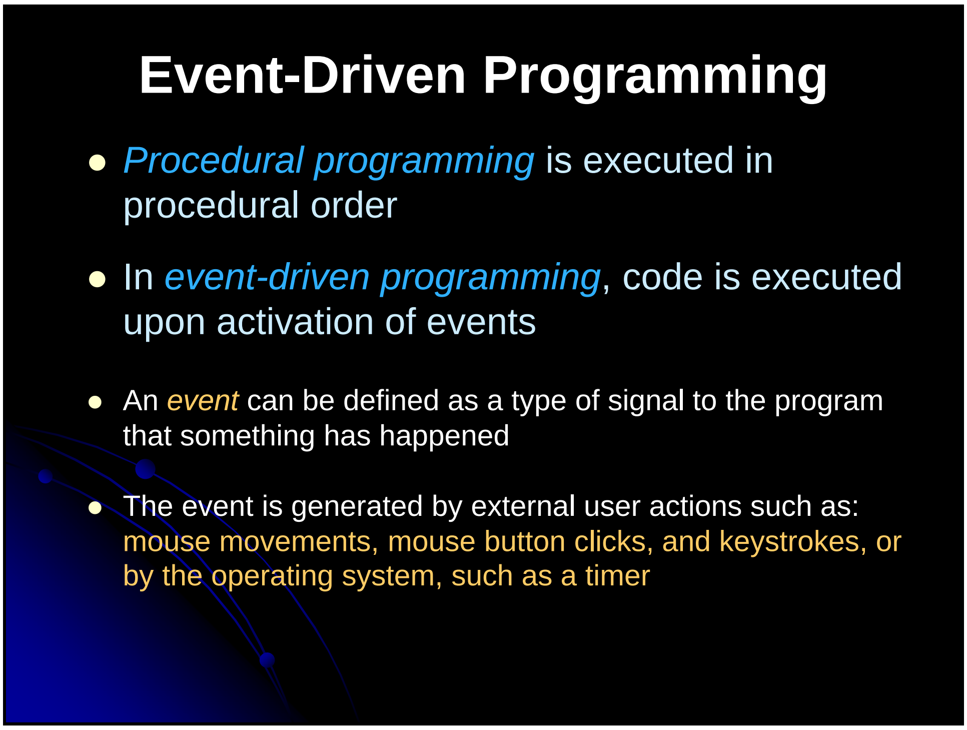


Handling Mouse and Keyboard Events



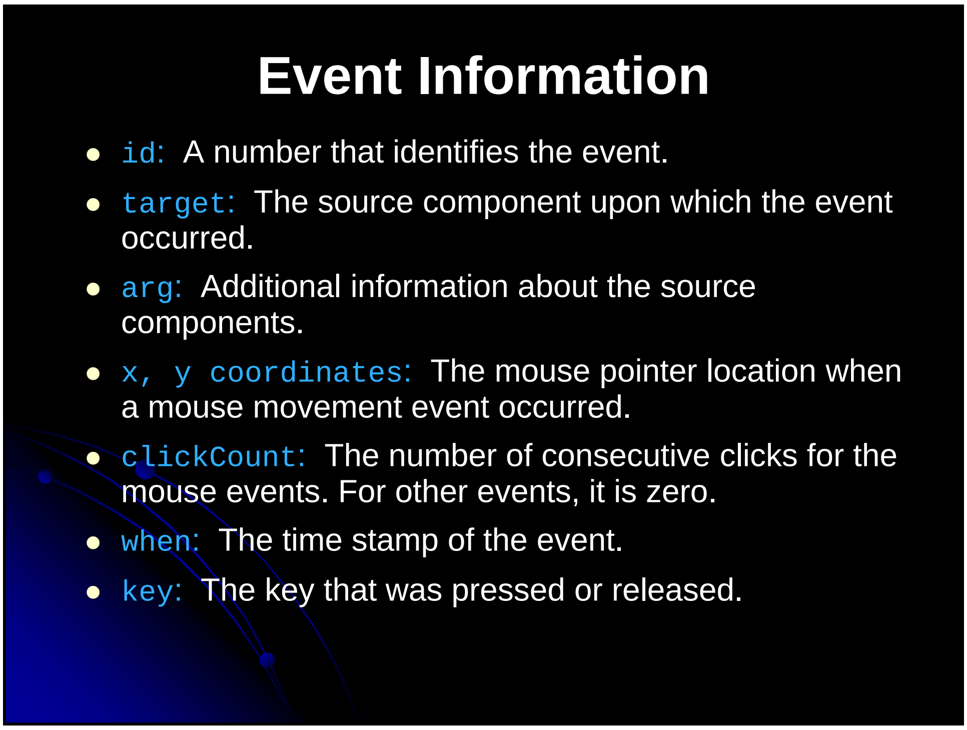
Agenda

- General event-handling strategy
- Handling events with separate listeners
- Handling events by implementing interfaces
- Handling events with named inner classes
- Handling events with anonymous inner classes
- The standard AWT listener types
- Subtleties with mouse events
- Examples



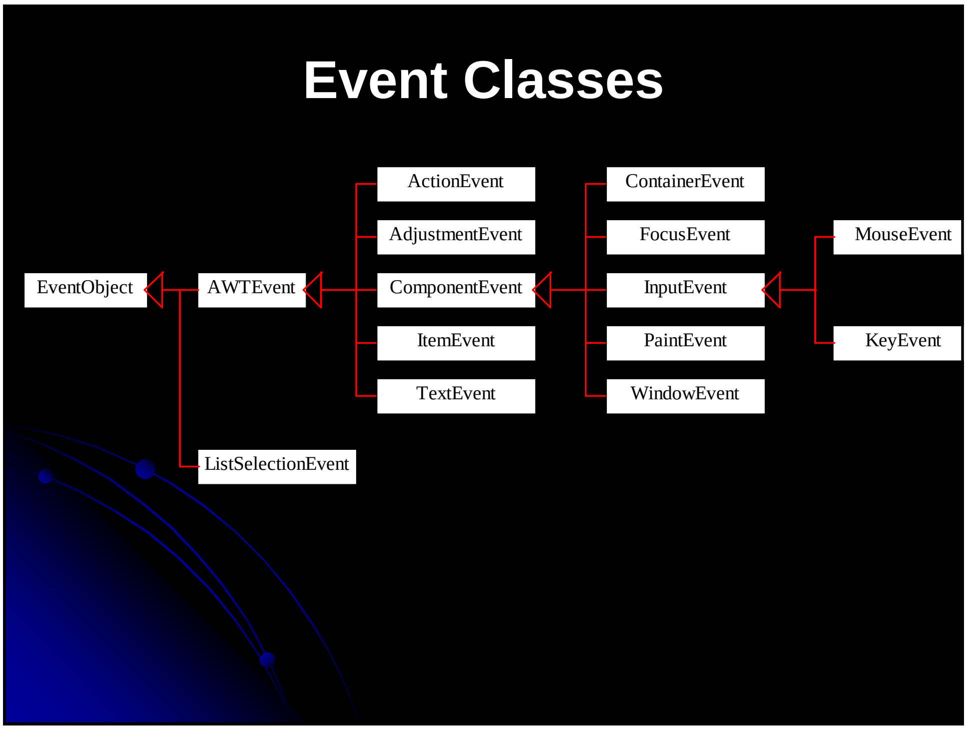
Event-Driven Programming

- *Procedural programming* is executed in procedural order
- In *event-driven programming*, code is executed upon activation of events
- An *event* can be defined as a type of signal to the program that something has happened
- The event is generated by external user actions such as: mouse movements, mouse button clicks, and keystrokes, or by the operating system, such as a timer



Event Information

- `id`: A number that identifies the event.
- `target`: The source component upon which the event occurred.
- `arg`: Additional information about the source components.
- `x, y coordinates`: The mouse pointer location when a mouse movement event occurred.
- `clickCount`: The number of consecutive clicks for the mouse events. For other events, it is zero.
- `when`: The time stamp of the event.
- `key`: The key that was pressed or released.



Event Classes

EventObject

AWTEvent

ListSelectionEvent

ActionEvent

AdjustmentEvent

ComponentEvent

ItemEvent

TextEvent

ContainerEvent

FocusEvent

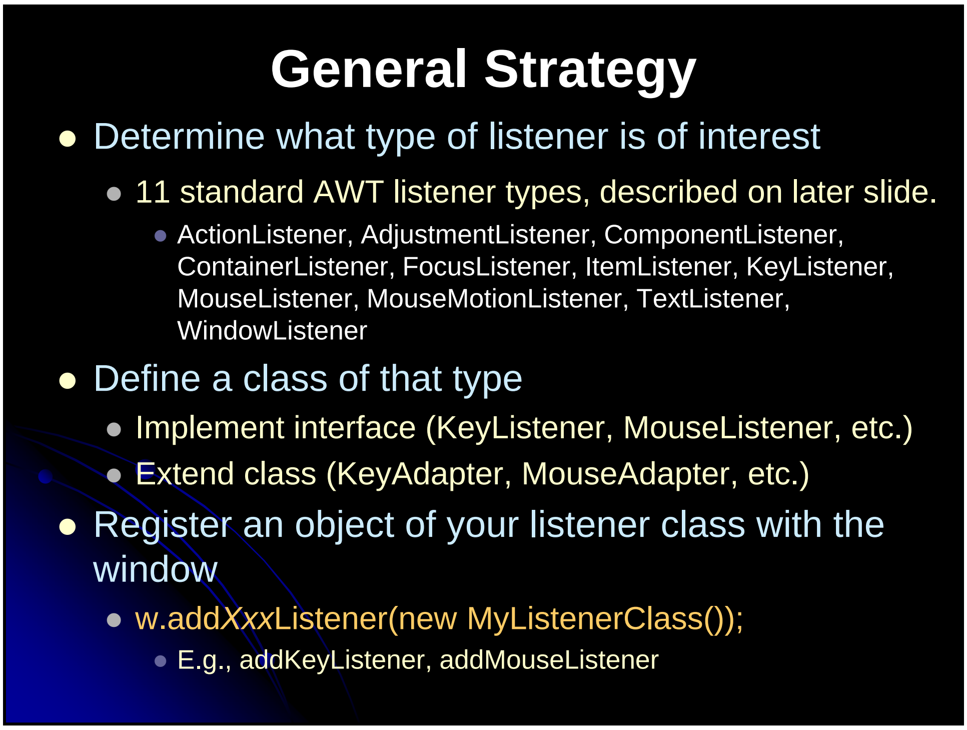
InputEvent

PaintEvent

WindowEvent

MouseEvent

KeyEvent

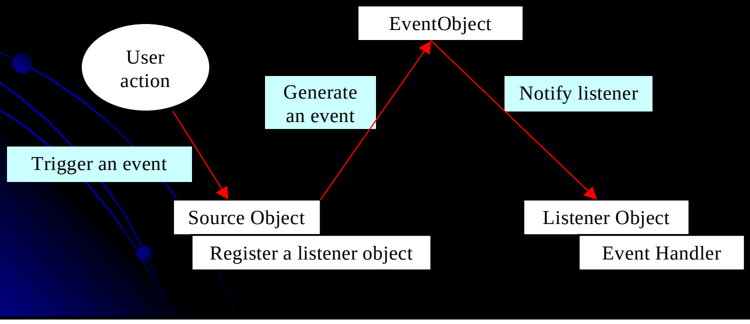


General Strategy

- Determine what type of listener is of interest
  - 11 standard AWT listener types, described on later slide.
    - ActionListener, AdjustmentListener, ComponentListener, ContainerListener, FocusListener, ItemListener, KeyListener, MouseListener, MouseMotionListener, TextListener, WindowListener
- Define a class of that type
  - Implement interface (KeyListener, MouseListener, etc.)
  - Extend class (KeyAdapter, MouseAdapter, etc.)
- Register an object of your listener class with the window
  - w.add*Xxx*Listener(new MyListenerClass());
    - E.g., addKeyListener, addMouseListener

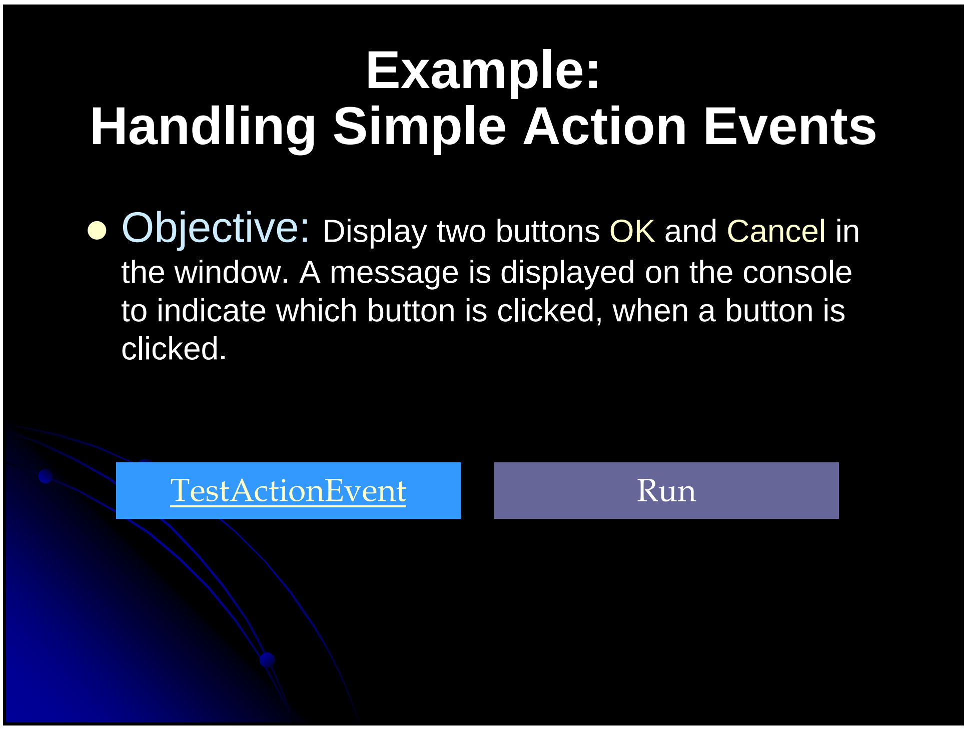
Selected User Actions

User Action	Source Object	Event Type Generated
Clicked on a button	JButton	ActionEvent
Changed text	JTextComponent	TextEvent
Double-clicked on a list item	JList	ActionEvent
Selected or deselected an item with a single click	JList	ItemEvent
Selected or deselected an item	JComboBox	ItemEvent



Selected Event Handlers

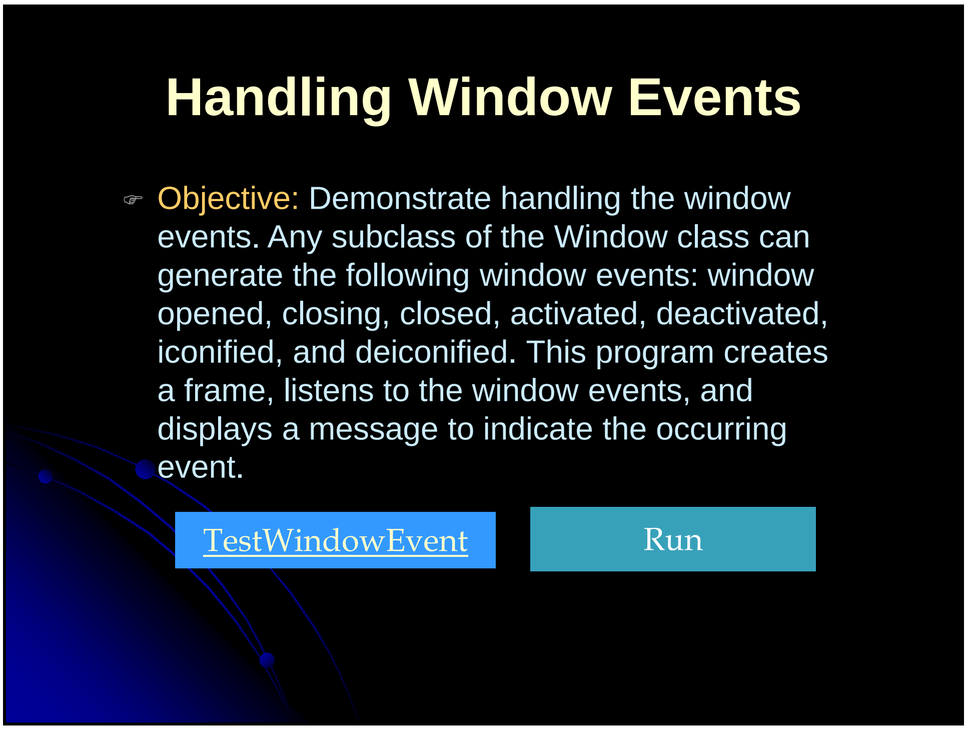
Event Class	Listener Interface	Listener Methods (Handlers)
`ActionEvent`	`ActionListener`	`actionPerformed(ActionEvent)`
`ItemEvent`	`ItemListener`	`itemStateChanged(ItemEvent)`
`WindowEvent`	`WindowListener`	`windowClosing(WindowEvent)`
		`windowOpened(WindowEvent)`
		`windowIconified(WindowEvent)`
		`windowDeiconified(WindowEvent)`
		`windowClosed(WindowEvent)`
		`windowActivated(WindowEvent)`
		`windowDeactivated(WindowEvent)`
`ContainerEvent`	`ContainerListener`	`componentAdded(ContainerEvent)`
		`componentRemoved(ContainerEvent)`



Example: Handling Simple Action Events

- Objective: Display two buttons OK and Cancel in the window. A message is displayed on the console to indicate which button is clicked, when a button is clicked.

TestActionEvent   Run



Handling Window Events

- Objective: Demonstrate handling the window events. Any subclass of the Window class can generate the following window events: window opened, closing, closed, activated, deactivated, iconified, and deiconified. This program creates a frame, listens to the window events, and displays a message to indicate the occurring event.

TestWindowEvent   Run

Example: Multiple Listeners for a Single Source

☞ Objective: This example modifies former Example to add a new listener for each button. The two buttons OK and Cancel use the frame class as the listner. This example creates a new listener class as an additional listener for the action events on the buttons. When a button is clicked, both listeners respond to the action event.

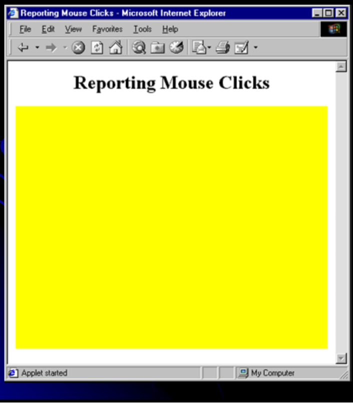
TestMultipleListener　　Run

Handling Events in Applets with a Separate Listener

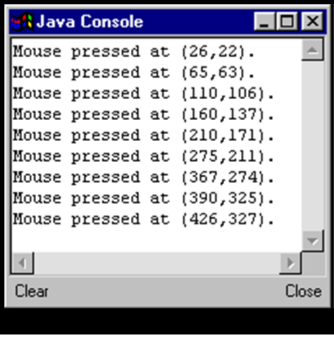
- Listener does not need to call any methods of the window to which it is attached

```
import java.applet.Applet;
import java.awt.*;

public class ClickReporter extends Applet {
  public void init() {
    setBackground(Color.yellow);
    addMouseListener(new ClickListener());
  }
}
```

```
import java.awt.event.*;

public class ClickListener extends MouseAdapter {
  public void mousePressed(MouseEvent event) {
    System.out.println("Mouse pressed at (" +
                       event.getX() + "," +
                       event.getY() + ").");
  }
}
```


Reporting Mouse Clicks - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Applet started
My Computer

Reporting Mouse Clicks



```
Mouse pressed at (26,22).
Mouse pressed at (65,63).
Mouse pressed at (110,106).
Mouse pressed at (160,137).
Mouse pressed at (210,171).
Mouse pressed at (275,211).
Mouse pressed at (367,274).
Mouse pressed at (390,325).
Mouse pressed at (426,327).
```

Generalizing Simple Case

- What if ClickListener wants to draw a circle wherever mouse is clicked?
- Why can’t it just call getGraphics to get a Graphics object with which to draw?
- General solution:
  - Call event.getSource to obtain a reference to window or GUI component from which event originated
  - Cast result to type of interest
  - Call methods on that reference

Handling Events with Separate Listener: General Case

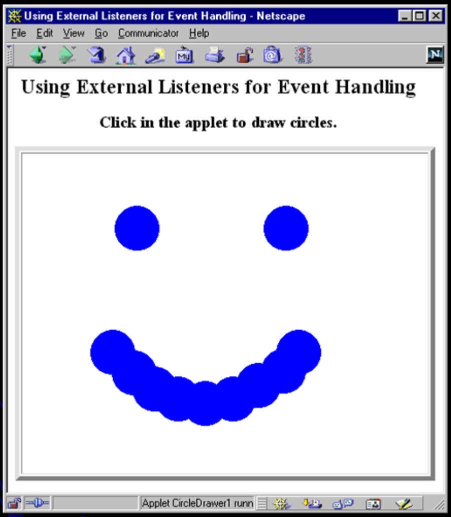
```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class CircleDrawer1 extends Applet {
  public void init() {
    setForeground(Color.blue);
    addMouseListener(new CircleListener());
  }
}
public class CircleListener extends MouseAdapter {
  private int radius = 25;
  public void mousePressed(MouseEvent event) {
    Applet app = (Applet)event.getSource();
    Graphics g = app.getGraphics();
    g.fillOval(event.getX()-radius, event.getY()-radius,
               2*radius, 2*radius);
  }
}
```

Separate Listener: General Case (Results)

Implemented as an applet

TestMouseClick

Run


Using External Listeners for Event Handling - Netscape
File Edit View Go Communicator Help
Applet CircleDrawer1 runn

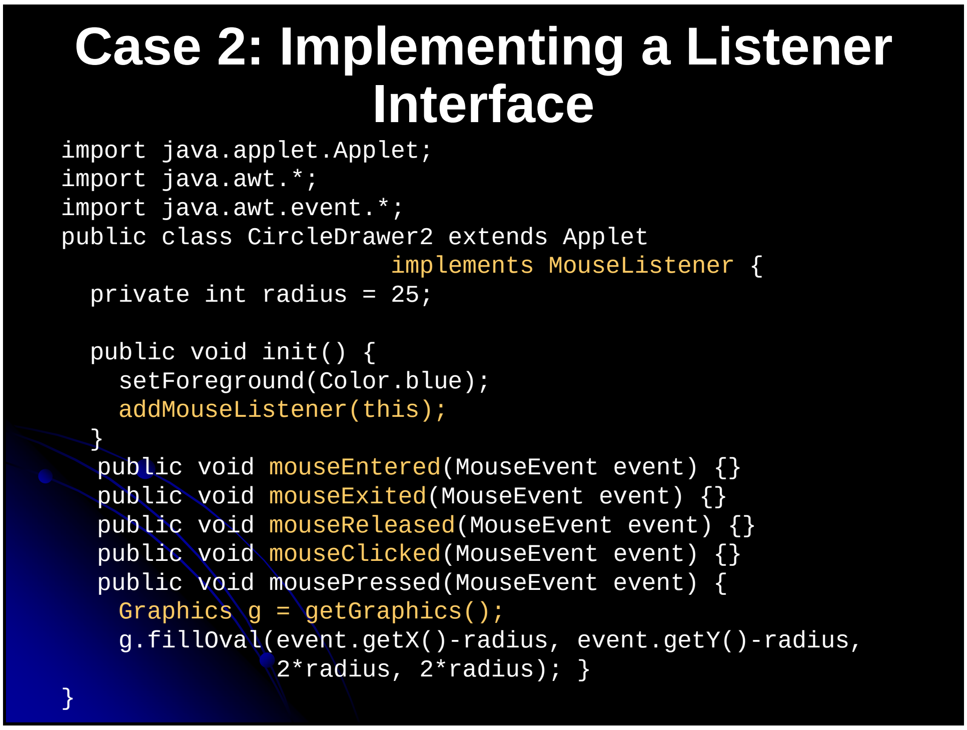
Using External Listeners for Event Handling

Click in the applet to draw circles.

Implemented as an application

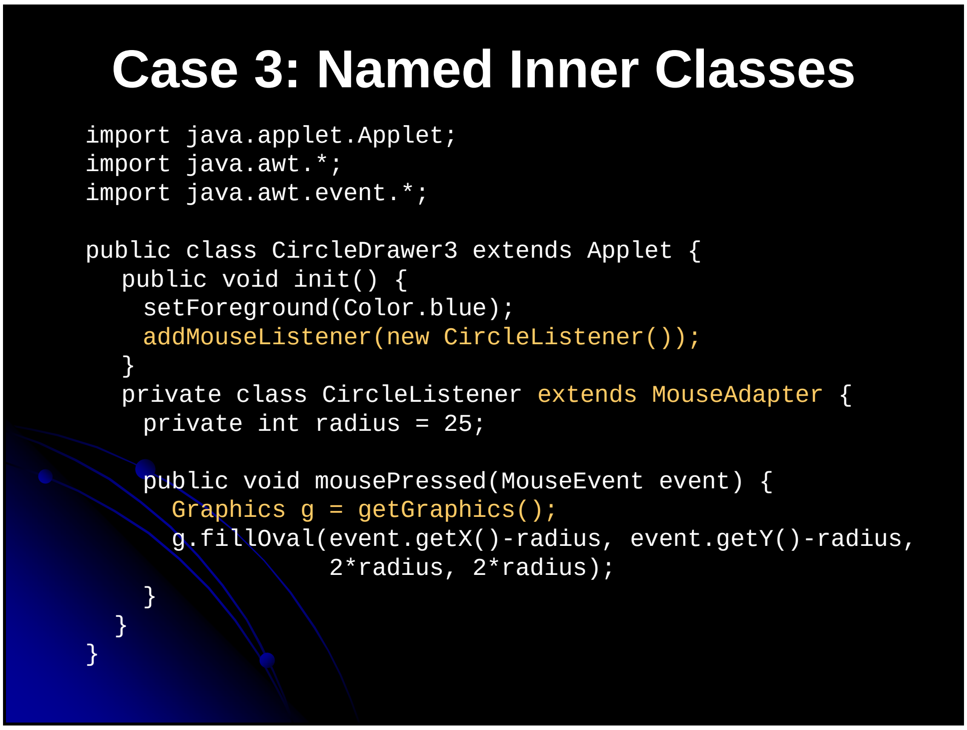
TestMouseClick

Run



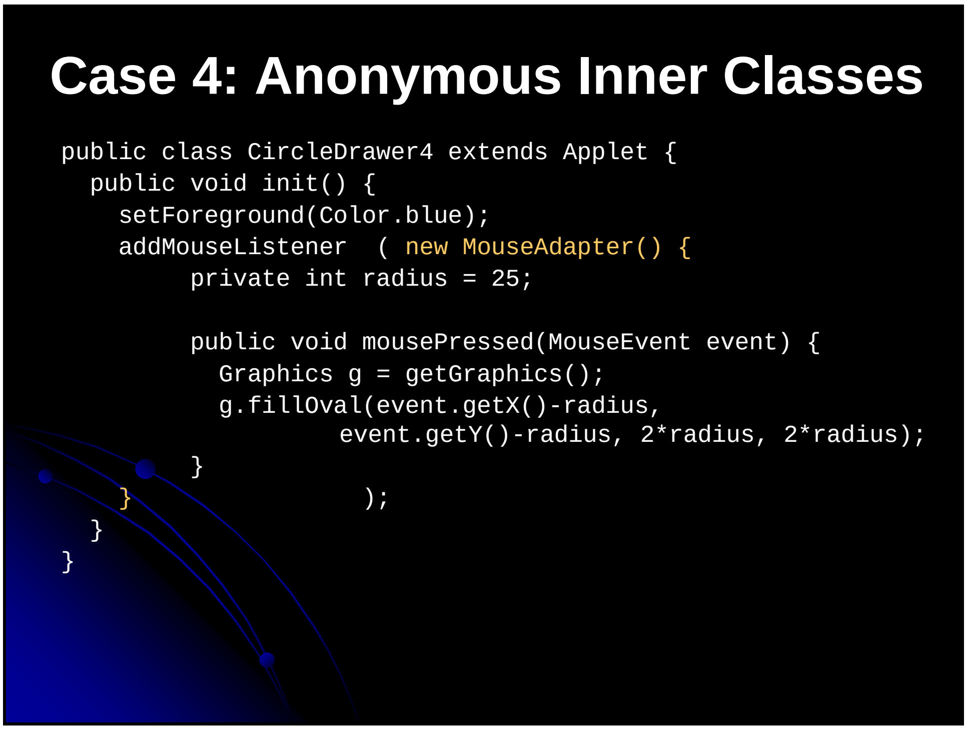
Case 2: Implementing a Listener Interface

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class CircleDrawer2 extends Applet
                       implements MouseListener {
  private int radius = 25;

  public void init() {
    setForeground(Color.blue);
    addMouseListener(this);
  }
  public void mouseEntered(MouseEvent event) {}
  public void mouseExited(MouseEvent event) {}
  public void mouseReleased(MouseEvent event) {}
  public void mouseClicked(MouseEvent event) {}
  public void mousePressed(MouseEvent event) {
    Graphics g = getGraphics();
    g.fillOval(event.getX()-radius, event.getY()-radius,
               2*radius, 2*radius); }
}
```



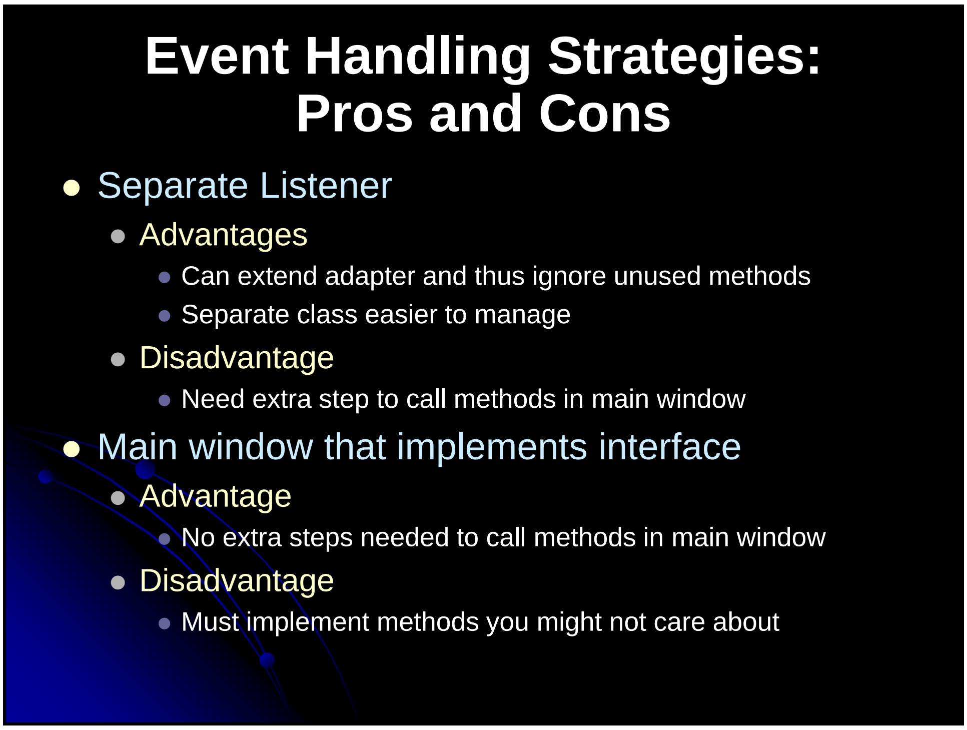
Case 3: Named Inner Classes

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class CircleDrawer3 extends Applet {
   public void init() {
    setForeground(Color.blue);
    addMouseListener(new CircleListener());
   }
   private class CircleListener extends MouseAdapter {
    private int radius = 25;

    public void mousePressed(MouseEvent event) {
      Graphics g = getGraphics();
      g.fillOval(event.getX()-radius, event.getY()-radius,
                 2*radius, 2*radius);
    }
  }
}
```



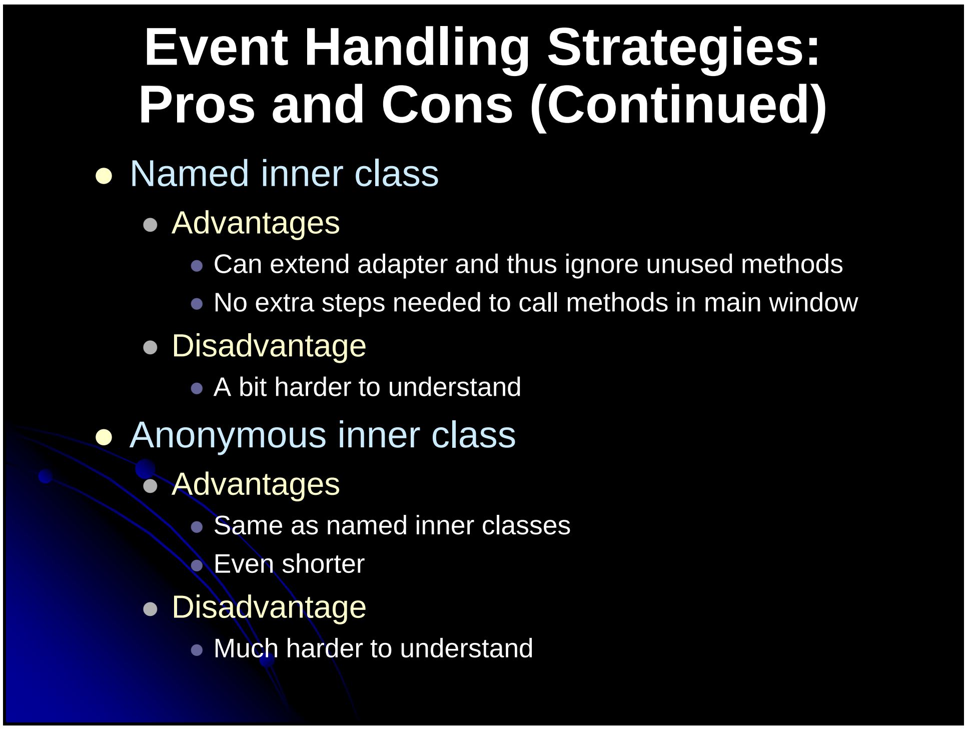
Case 4: Anonymous Inner Classes

```
public class CircleDrawer4 extends Applet {
  public void init() {
    setForeground(Color.blue);
    addMouseListener  ( new MouseAdapter() {
         private int radius = 25;

         public void mousePressed(MouseEvent event) {
           Graphics g = getGraphics();
           g.fillOval(event.getX()-radius,
                    event.getY()-radius, 2*radius, 2*radius);
         }
    }                     );
  }
}
```



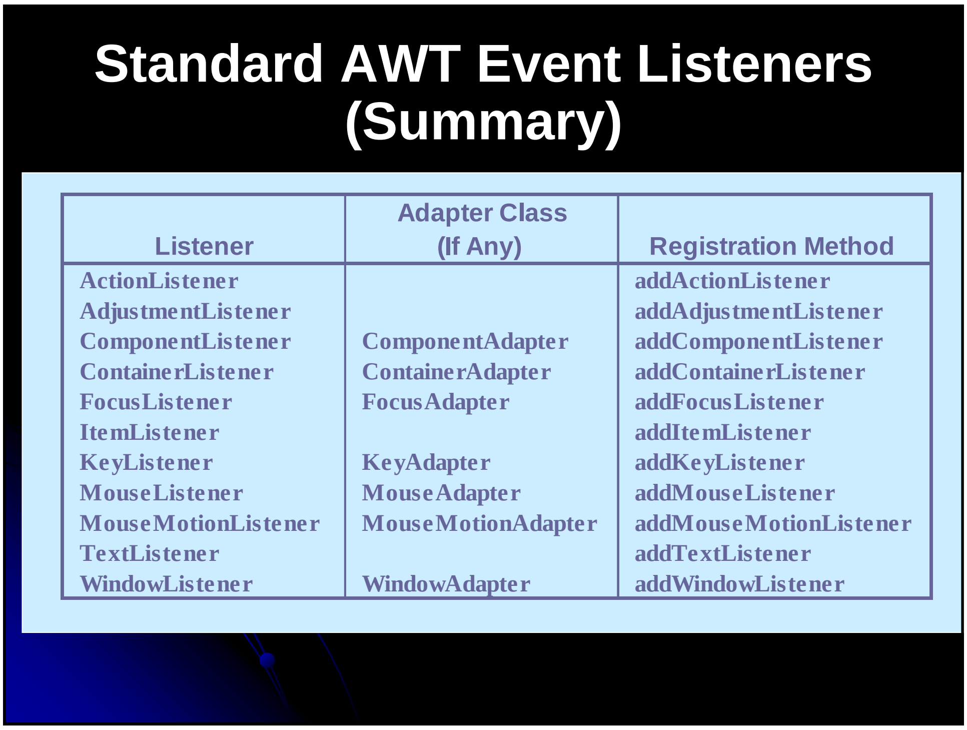
Event Handling Strategies: Pros and Cons

- Separate Listener
  - Advantages
    - Can extend adapter and thus ignore unused methods
    - Separate class easier to manage
  - Disadvantage
    - Need extra step to call methods in main window
- Main window that implements interface
  - Advantage
    - No extra steps needed to call methods in main window
  - Disadvantage
    - Must implement methods you might not care about



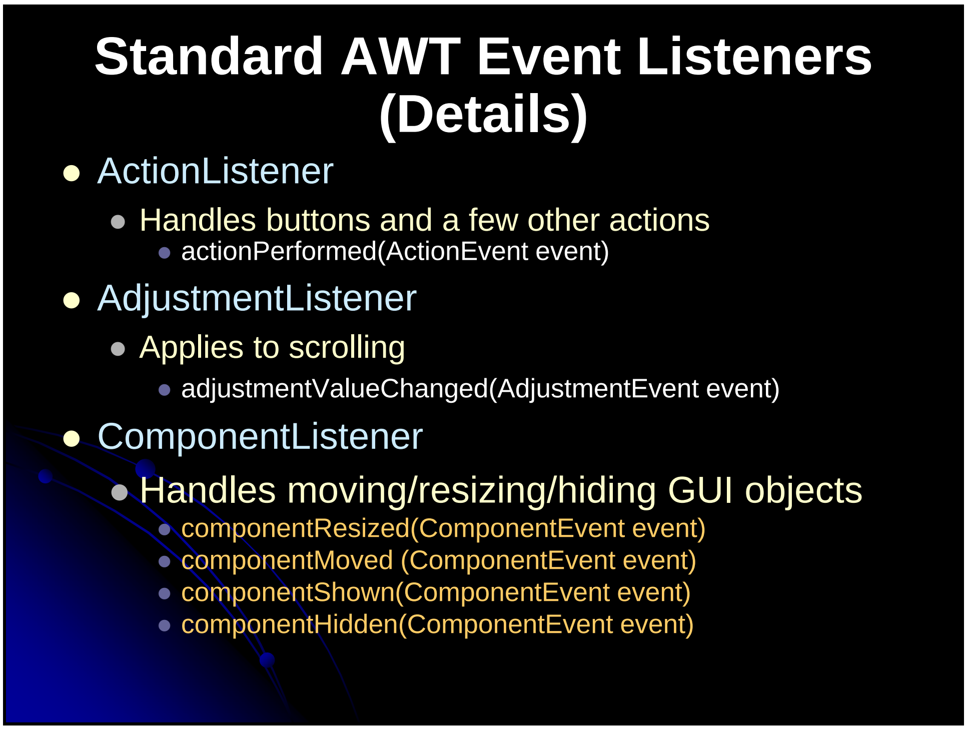
Event Handling Strategies: Pros and Cons (Continued)

- Named inner class
  - Advantages
    - Can extend adapter and thus ignore unused methods
    - No extra steps needed to call methods in main window
  - Disadvantage
    - A bit harder to understand
- Anonymous inner class
  - Advantages
    - Same as named inner classes
    - Even shorter
  - Disadvantage
    - Much harder to understand



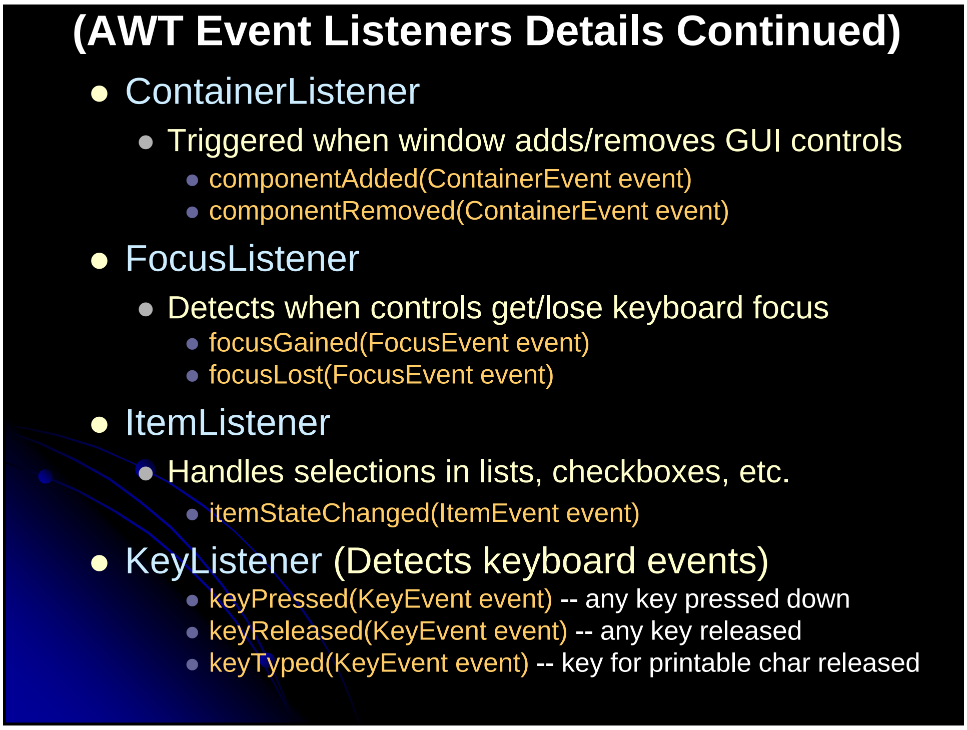
Standard AWT Event Listeners (Summary)

Listener	Adapter Class (If Any)	Registration Method
ActionListener		addActionListener
AdjustmentListener		addAdjustmentListener
ComponentListener	ComponentAdapter	addComponentListener
ContainerListener	ContainerAdapter	addContainerListener
FocusListener	FocusAdapter	addFocusListener
ItemListener		addItemListener
KeyListener	KeyAdapter	addKeyListener
MouseListener	MouseAdapter	addMouseListener
MouseMotionListener	MouseMotionAdapter	addMouseMotionListener
TextListener		addTextListener
WindowListener	WindowAdapter	addWindowListener



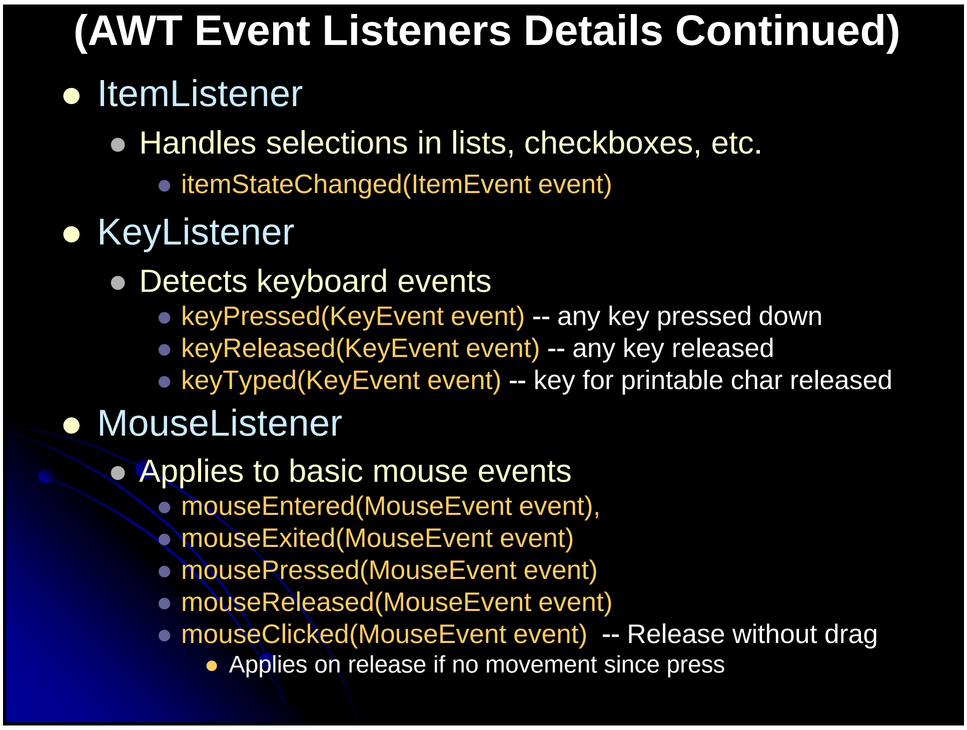
Standard AWT Event Listeners (Details)

- ActionListener
  - Handles buttons and a few other actions
    - actionPerformed(ActionEvent event)
- AdjustmentListener
  - Applies to scrolling
    - adjustmentValueChanged(AdjustmentEvent event)
- ComponentListener
  - Handles moving/resizing/hiding GUI objects
    - componentResized(ComponentEvent event)
    - componentMoved (ComponentEvent event)
    - componentShown(ComponentEvent event)
    - componentHidden(ComponentEvent event)



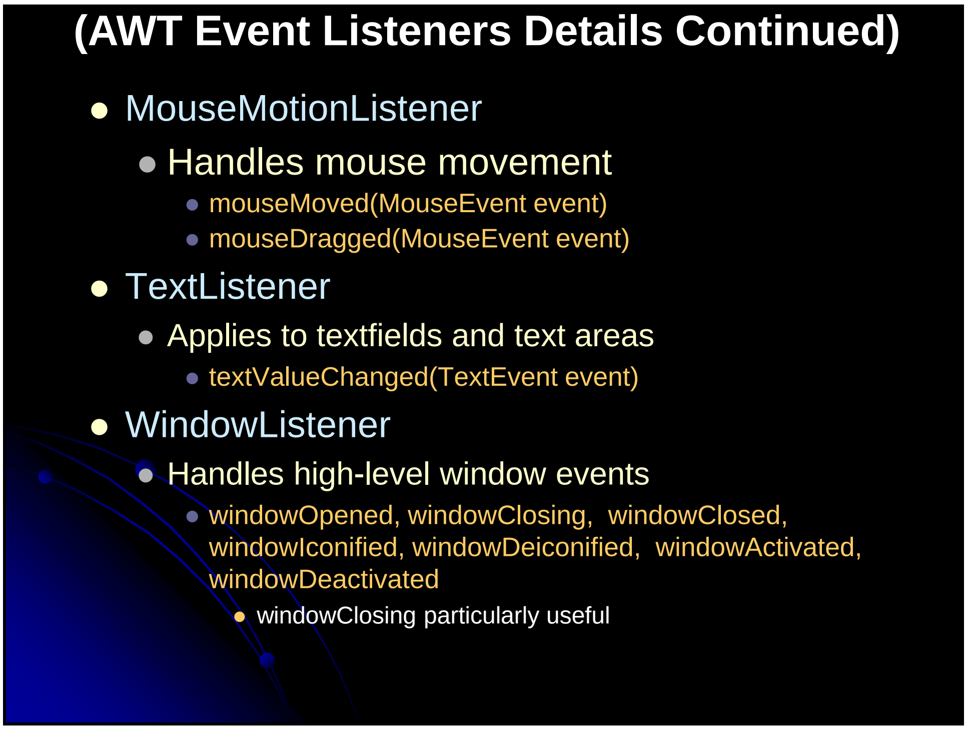
(AWT Event Listeners Details Continued)

- ContainerListener
  - Triggered when window adds/removes GUI controls
    - componentAdded(ContainerEvent event)
    - componentRemoved(ContainerEvent event)
- FocusListener
  - Detects when controls get/lose keyboard focus
    - focusGained(FocusEvent event)
    - focusLost(FocusEvent event)
- ItemListener
  - Handles selections in lists, checkboxes, etc.
    - itemStateChanged(ItemEvent event)
- KeyListener (Detects keyboard events)
    - keyPressed(KeyEvent event) -- any key pressed down
    - keyReleased(KeyEvent event) -- any key released
    - keyTyped(KeyEvent event) -- key for printable char released



(AWT Event Listeners Details Continued)

- ItemListener
  - Handles selections in lists, checkboxes, etc.
    - itemStateChanged(ItemEvent event)
- KeyListener
  - Detects keyboard events
    - keyPressed(KeyEvent event) -- any key pressed down
    - keyReleased(KeyEvent event) -- any key released
    - keyTyped(KeyEvent event) -- key for printable char released
- MouseListener
  - Applies to basic mouse events
    - mouseEntered(MouseEvent event),
    - mouseExited(MouseEvent event)
    - mousePressed(MouseEvent event)
    - mouseReleased(MouseEvent event)
    - mouseClicked(MouseEvent event) -- Release without drag
      - Applies on release if no movement since press



(AWT Event Listeners Details Continued)

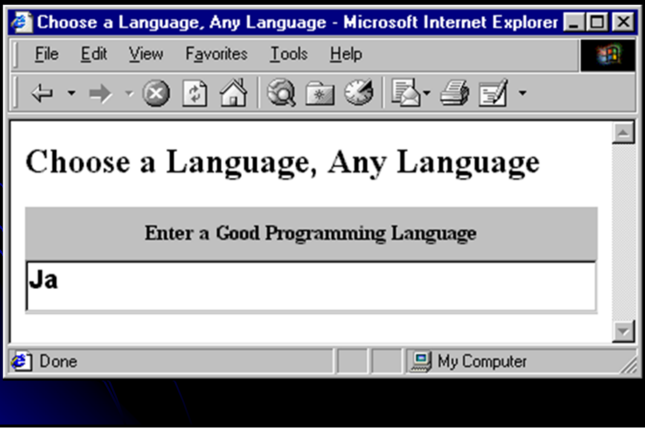
- MouseMotionListener
  - Handles mouse movement
    - mouseMoved(MouseEvent event)
    - mouseDragged(MouseEvent event)
- TextListener
  - Applies to textfields and text areas
    - textValueChanged(TextEvent event)
- WindowListener
  - Handles high-level window events
    - windowOpened, windowClosing, windowClosed, windowIconified, windowDeiconified, windowActivated, windowDeactivated
      - windowClosing particularly useful

Mouse Events: Details

- MouseListener and MouseMotionListener share event types
- Location of clicks
  - event.getX() and event.getY()
- Double clicks
  - Determined by OS, not by programmer
  - Call event.getClickCount()
- Distinguishing mouse buttons
  - Call event.getModifiers() and compare to MouseEvent.Button2_MASK for a middle click and MouseEvent.Button3_MASK for right click.
  - Can also trap Shift-click, Alt-click, etc.

Simple Example: Spelling-Correcting Textfield

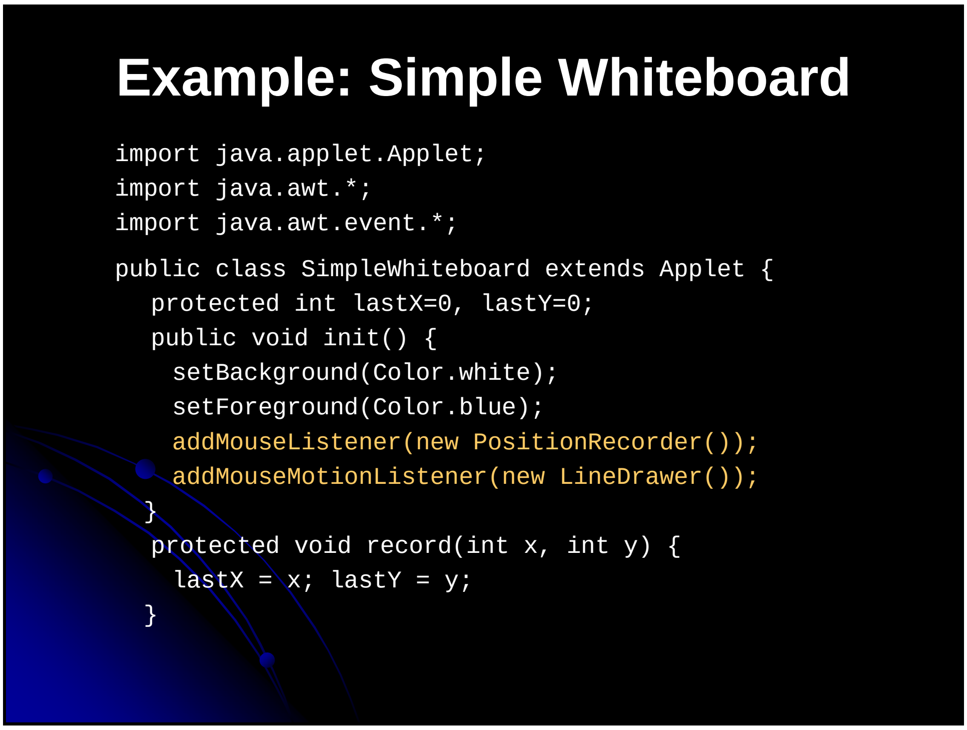
- KeyListener corrects spelling during typing
- ActionListener completes word on ENTER
- FocusListener gives subliminal hints


Choose a Language, Any Language - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Done
My Computer

Choose a Language, Any Language

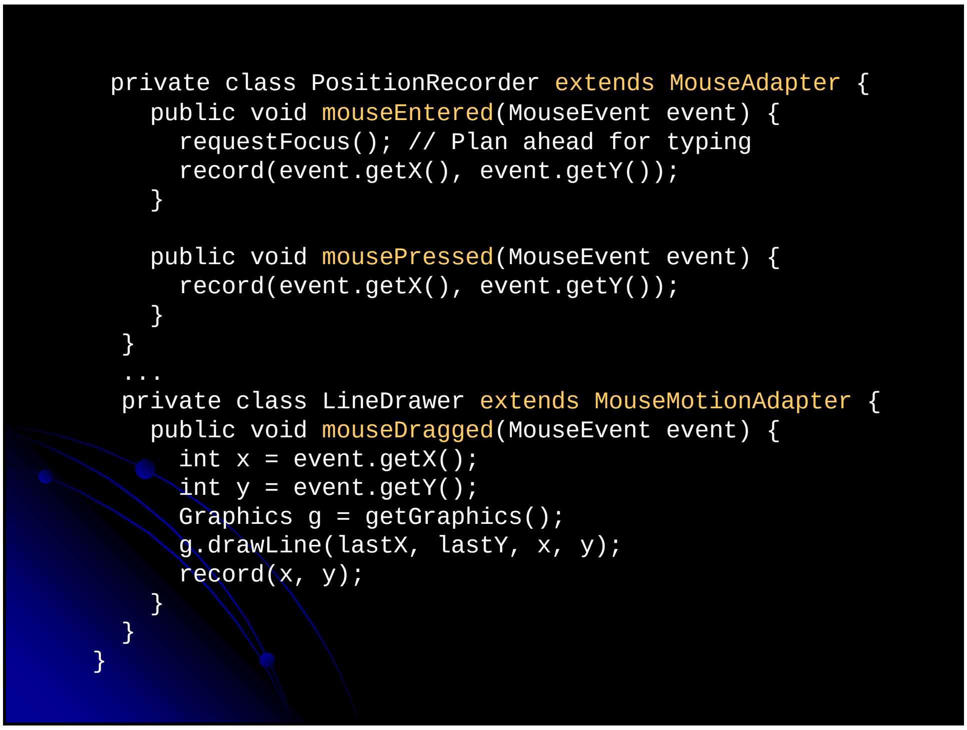
Enter a Good Programming Language

Ja



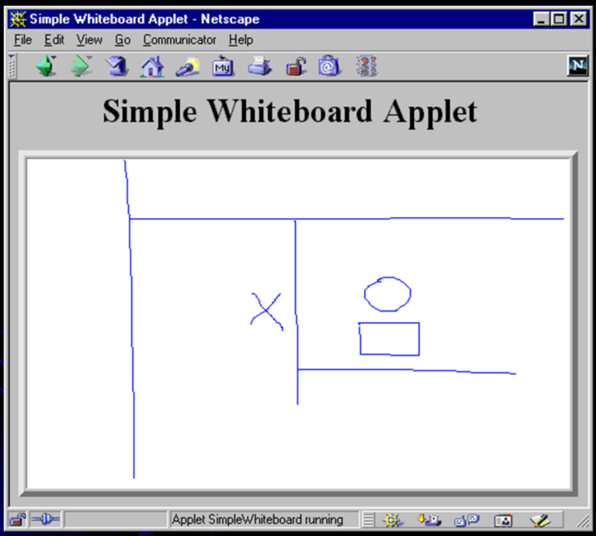
Example: Simple Whiteboard

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class SimpleWhiteboard extends Applet {
  protected int lastX=0, lastY=0;
  public void init() {
    setBackground(Color.white);
    setForeground(Color.blue);
    addMouseListener(new PositionRecorder());
    addMouseMotionListener(new LineDrawer());
  }
  protected void record(int x, int y) {
    lastX = x; lastY = y;
  }
```



```
  private class PositionRecorder extends MouseAdapter {
    public void mouseEntered(MouseEvent event) {
      requestFocus(); // Plan ahead for typing
      record(event.getX(), event.getY());
    }

    public void mousePressed(MouseEvent event) {
      record(event.getX(), event.getY());
    }
  }
  ...
  private class LineDrawer extends MouseMotionAdapter {
    public void mouseDragged(MouseEvent event) {
      int x = event.getX();
      int y = event.getY();
      Graphics g = getGraphics();
      g.drawLine(lastX, lastY, x, y);
      record(x, y);
    }
  }
}
```

Simple Whiteboard (Results)


Simple Whiteboard Applet - Netscape
File Edit View Go Communicator Help

Simple Whiteboard Applet

Applet SimpleWhiteboard running

Implemented as an applet

TestMouseClick

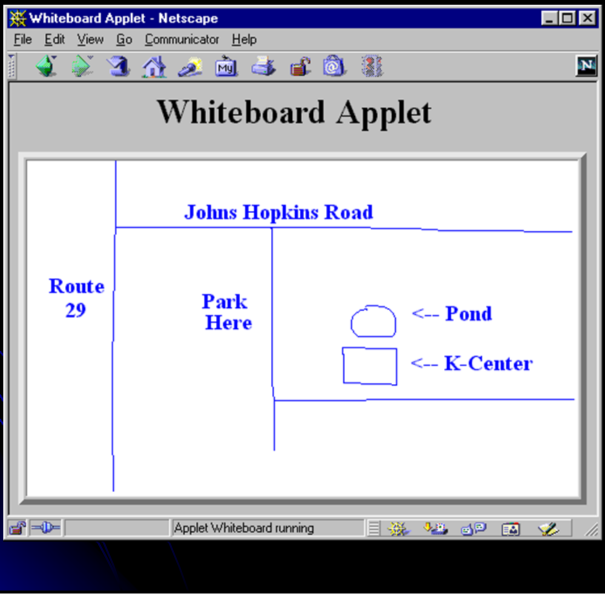
Run

Whiteboard: Adding Keyboard Events

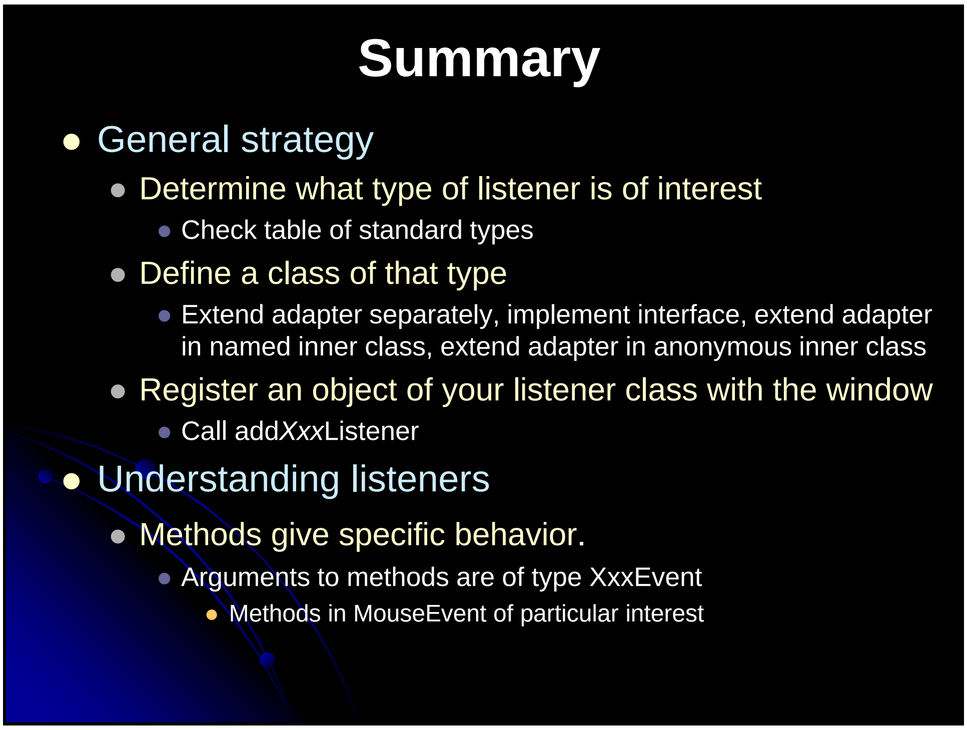
```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class Whiteboard extends SimpleWhiteboard {
  protected FontMetrics fm;

  public void init() {
    super.init();
    Font font = new Font("Serif", Font.BOLD, 20);
    setFont(font);
    fm = getFontMetrics(font);
    addKeyListener(new CharDrawer());
  }
```

```
 ...
 private class CharDrawer extends KeyAdapter {
  // When user types a printable character,
  // draw it and shift position rightwards.

  public void keyTyped(KeyEvent event) {
   String s = String.valueOf(event.getKeyChar());
   getGraphics().drawString(s, lastX, lastY);
   record(lastX + fm.stringWidth(s), lastY);
  }
 }
}
```

Whiteboard (Results)

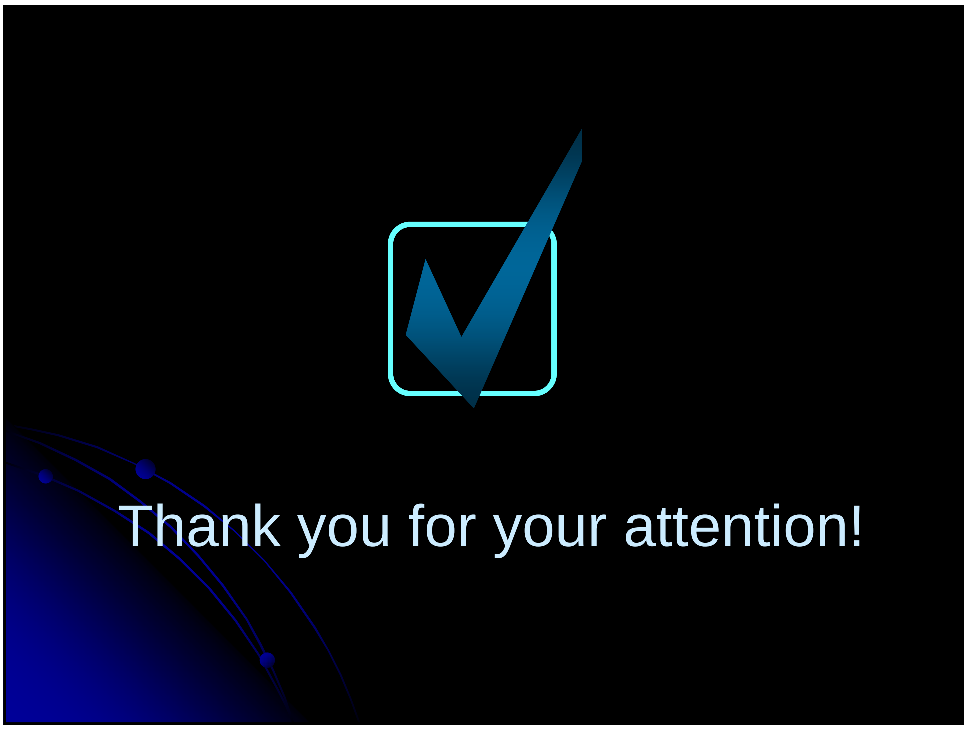


Whiteboard Applet - Netscape
File Edit View Go Communicator Help
Whiteboard Applet
Johns Hopkins Road
Route 29
Park Here
<-- Pond
<-- K-Center
Applet Whiteboard running



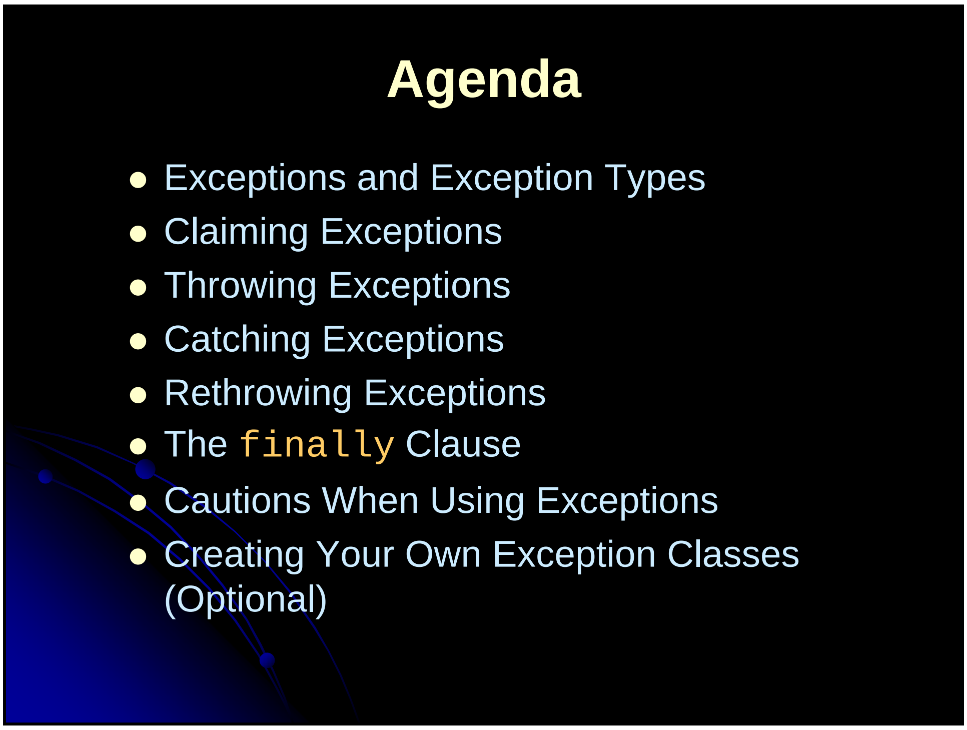
Summary

- General strategy
  - Determine what type of listener is of interest
    - Check table of standard types
  - Define a class of that type
    - Extend adapter separately, implement interface, extend adapter in named inner class, extend adapter in anonymous inner class
  - Register an object of your listener class with the window
    - Call add*Xxx*Listener
- Understanding listeners
  - Methods give specific behavior.
    - Arguments to methods are of type XxxEvent
      - Methods in MouseEvent of particular interest



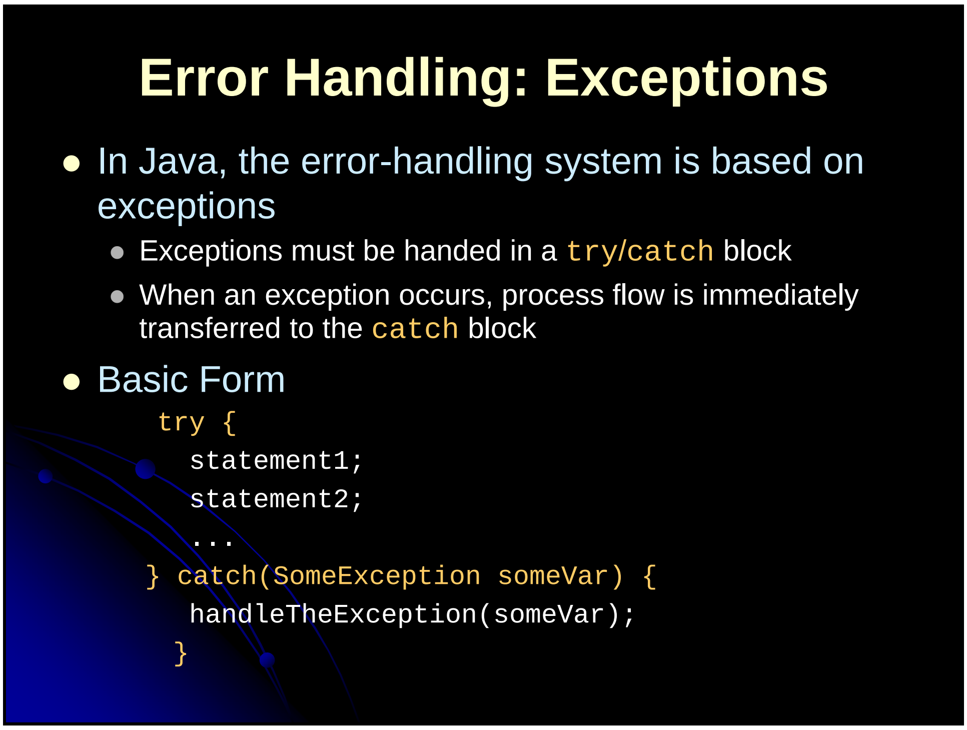
Thank you for your attention!

Exception



Agenda

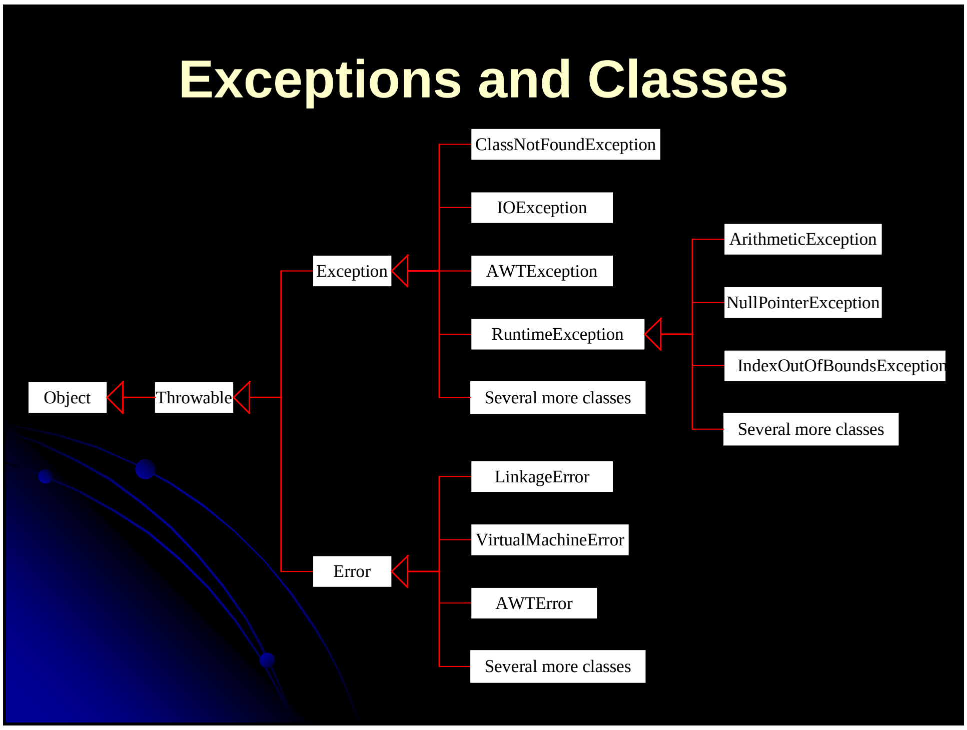
- Exceptions and Exception Types
- Claiming Exceptions
- Throwing Exceptions
- Catching Exceptions
- Rethrowing Exceptions
- The `finally` Clause
- Cautions When Using Exceptions
- Creating Your Own Exception Classes (Optional)



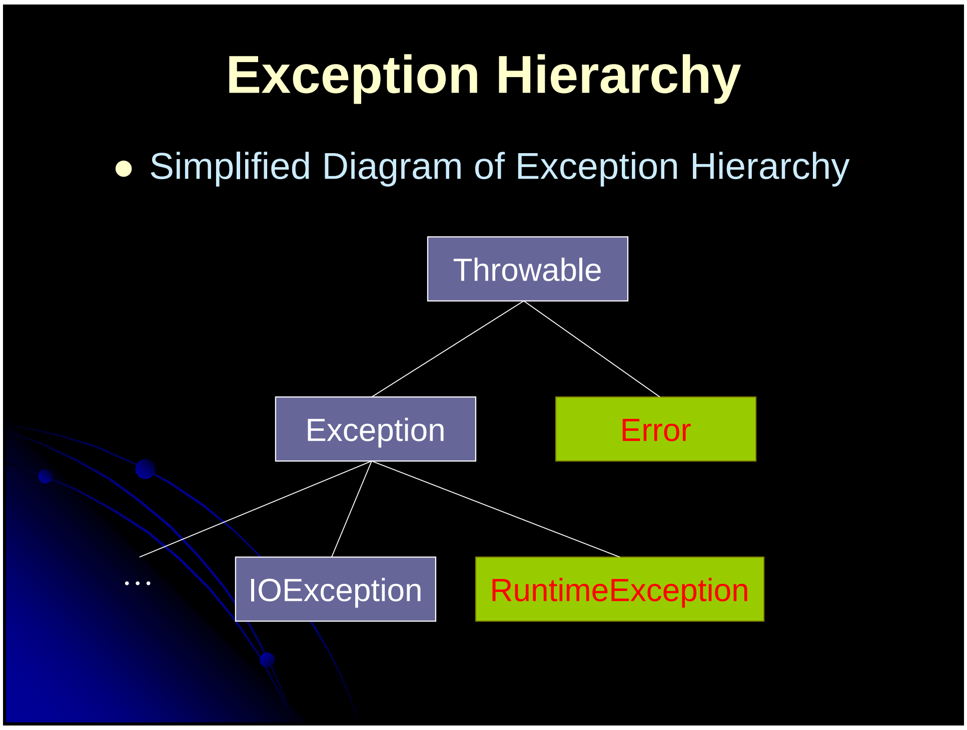
Error Handling: Exceptions

- In Java, the error-handling system is based on exceptions
  - Exceptions must be handed in a `try/catch` block
  - When an exception occurs, process flow is immediately transferred to the `catch` block
- Basic Form

```
    try {
      statement1;
      statement2;
      ...
    } catch(SomeException someVar) {
      handleTheException(someVar);
    }
```



Exceptions and Classes



Exception Hierarchy

- Simplified Diagram of Exception Hierarchy

Throwable

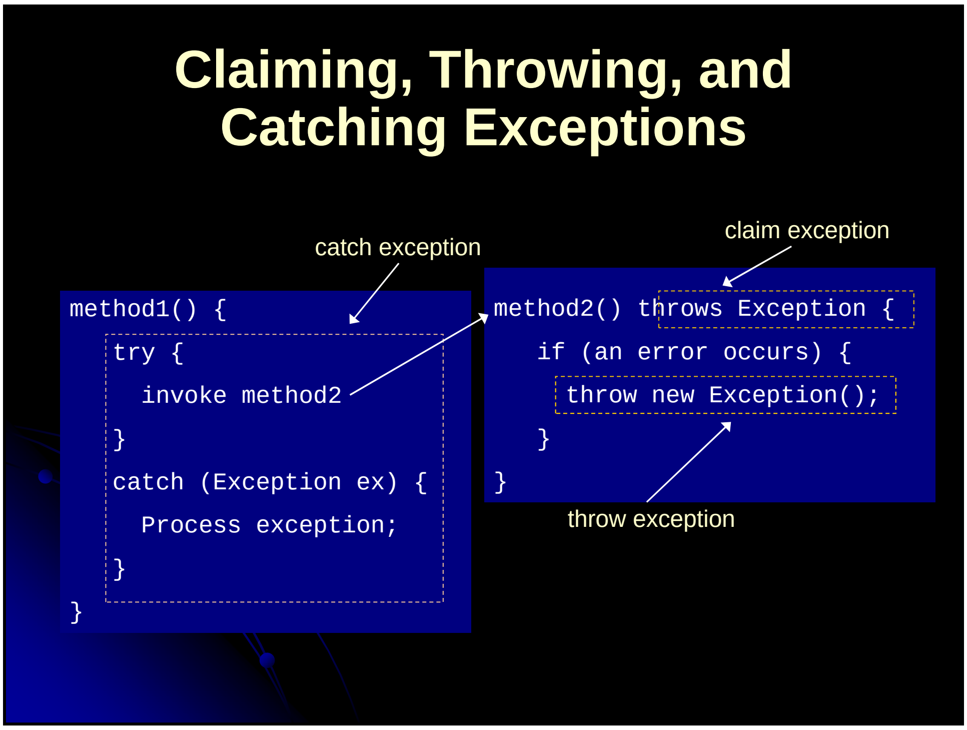
Exception

Error

…

IOException

RuntimeException



Claiming, Throwing, and Catching Exceptions

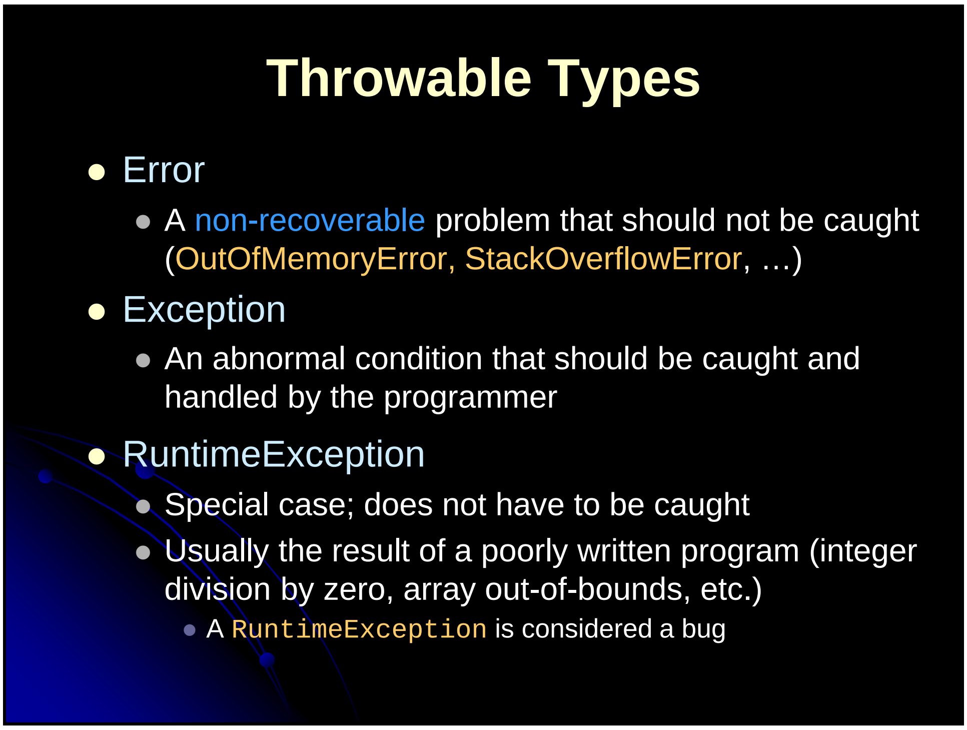
catch exception

claim exception

```
method1() {
  try {
    invoke method2
  }
  catch (Exception ex) {
    Process exception;
  }
}
```

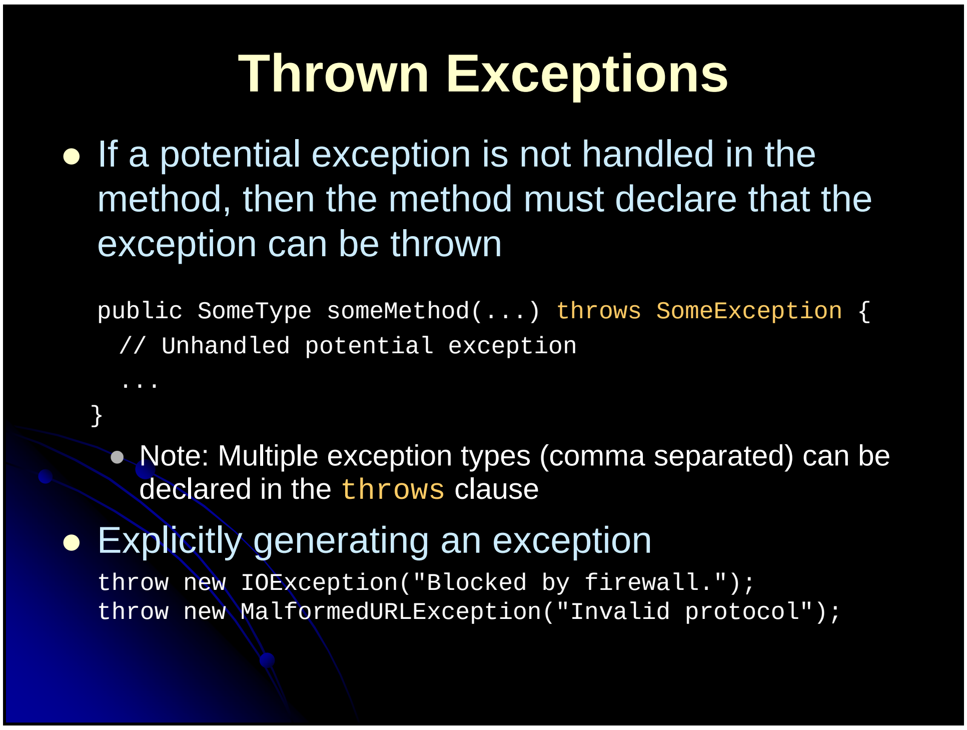
```
method2() throws Exception {
  if (an error occurs) {
    throw new Exception();
  }
}
```

throw exception



Throwable Types

- Error
  - A non-recoverable problem that should not be caught (OutOfMemoryError, StackOverflowError, …)
- Exception
  - An abnormal condition that should be caught and handled by the programmer
- RuntimeException
  - Special case; does not have to be caught
  - Usually the result of a poorly written program (integer division by zero, array out-of-bounds, etc.)
    - A `RuntimeException` is considered a bug



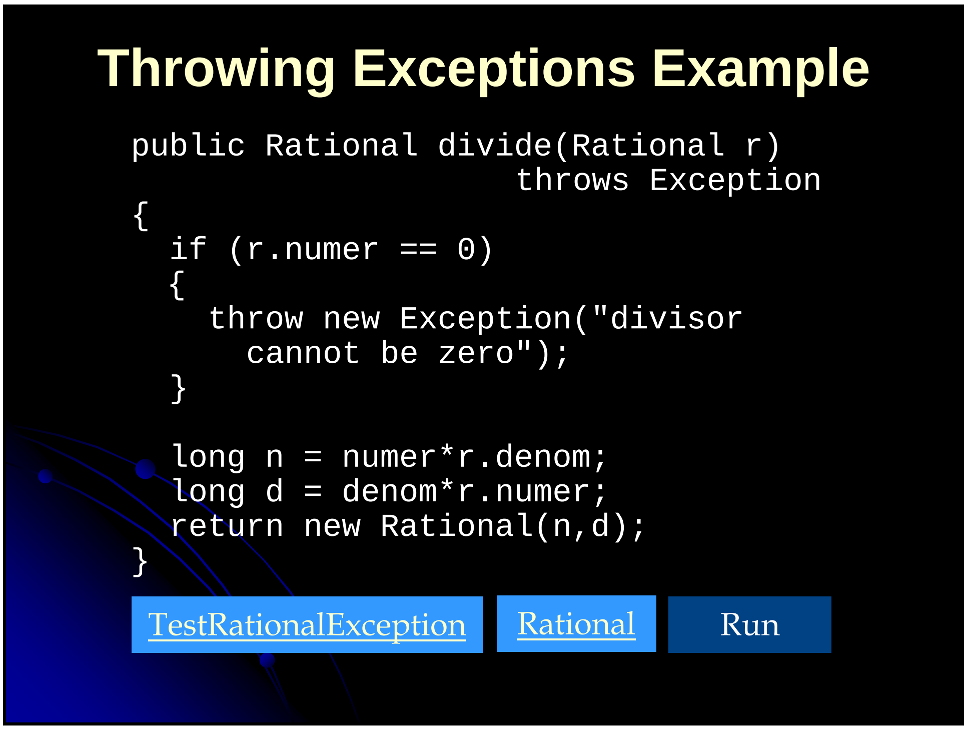
Thrown Exceptions

- If a potential exception is not handled in the method, then the method must declare that the exception can be thrown

```
public SomeType someMethod(...) throws SomeException {
  // Unhandled potential exception
  ...
}
```

  - Note: Multiple exception types (comma separated) can be declared in the `throws` clause
- Explicitly generating an exception

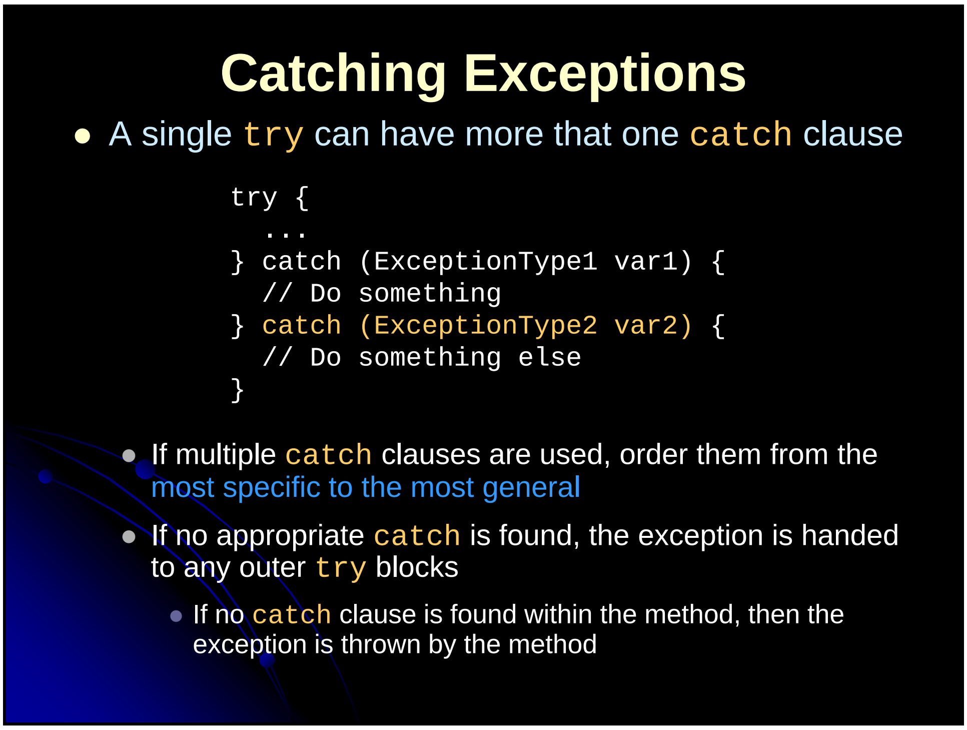
```
throw new IOException("Blocked by firewall.");
throw new MalformedURLException("Invalid protocol");
```



Throwing Exceptions Example

```
public Rational divide(Rational r)
                    throws Exception
{
  if (r.numer == 0)
  {
    throw new Exception("divisor
      cannot be zero");
  }

  long n = numer*r.denom;
  long d = denom*r.numer;
  return new Rational(n,d);
}
```

TestRationalException Rational Run

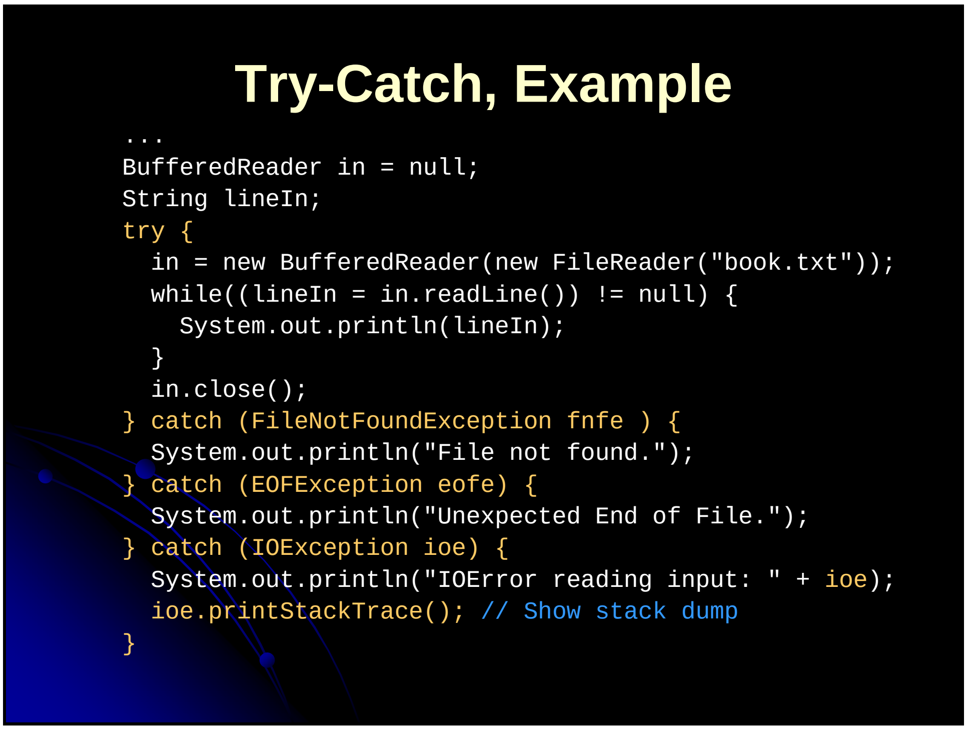


Catching Exceptions

- A single try can have more that one catch clause

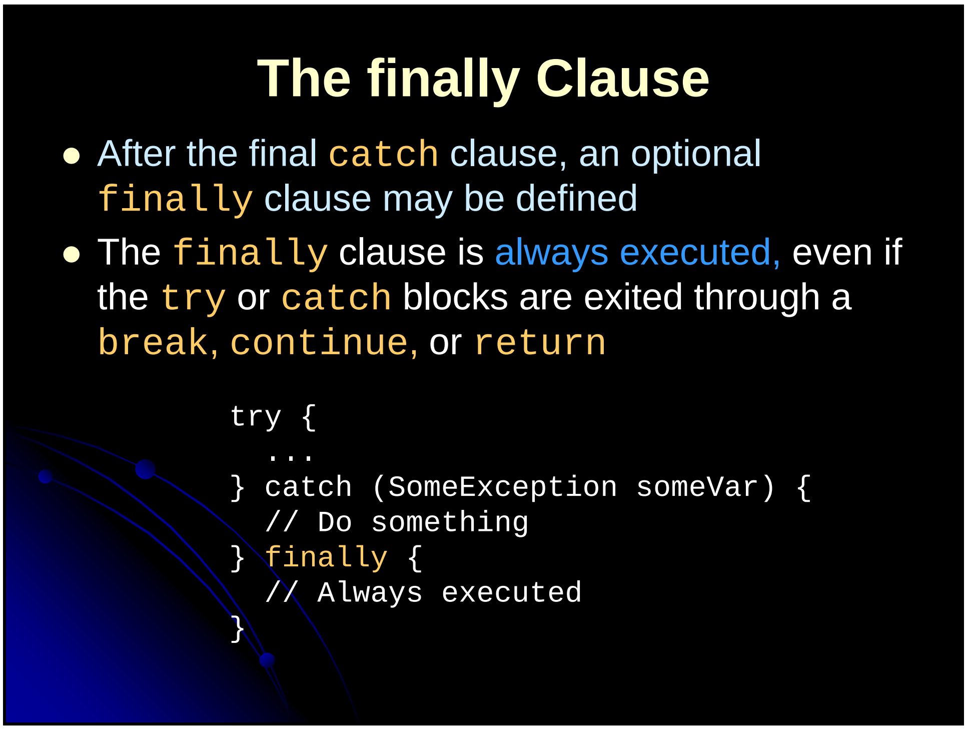
```
try {
  ...
} catch (ExceptionType1 var1) {
  // Do something
} catch (ExceptionType2 var2) {
  // Do something else
}
```

- If multiple catch clauses are used, order them from the most specific to the most general
- If no appropriate catch is found, the exception is handed to any outer try blocks
  - If no catch clause is found within the method, then the exception is thrown by the method



Try-Catch, Example

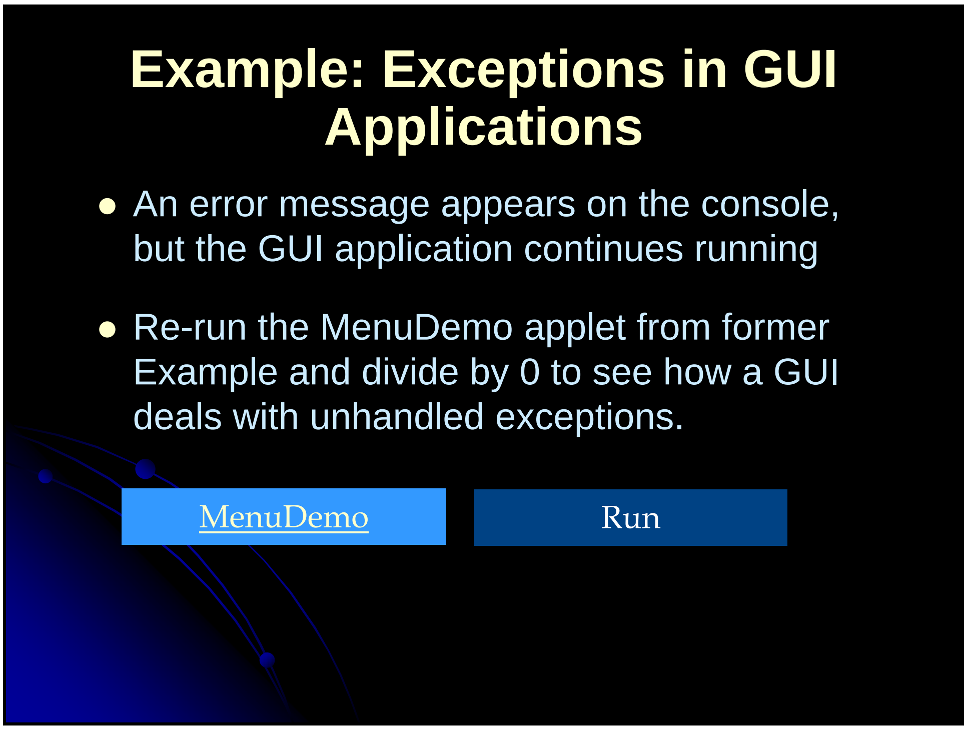
```
...
BufferedReader in = null;
String lineIn;
try {
  in = new BufferedReader(new FileReader("book.txt"));
  while((lineIn = in.readLine()) != null) {
    System.out.println(lineIn);
  }
  in.close();
} catch (FileNotFoundException fnfe ) {
  System.out.println("File not found.");
} catch (EOFException eofe) {
  System.out.println("Unexpected End of File.");
} catch (IOException ioe) {
  System.out.println("IOError reading input: " + ioe);
  ioe.printStackTrace(); // Show stack dump
}
```



The finally Clause

- After the final `catch` clause, an optional `finally` clause may be defined
- The `finally` clause is always executed, even if the `try` or `catch` blocks are exited through a `break`, `continue`, or `return`

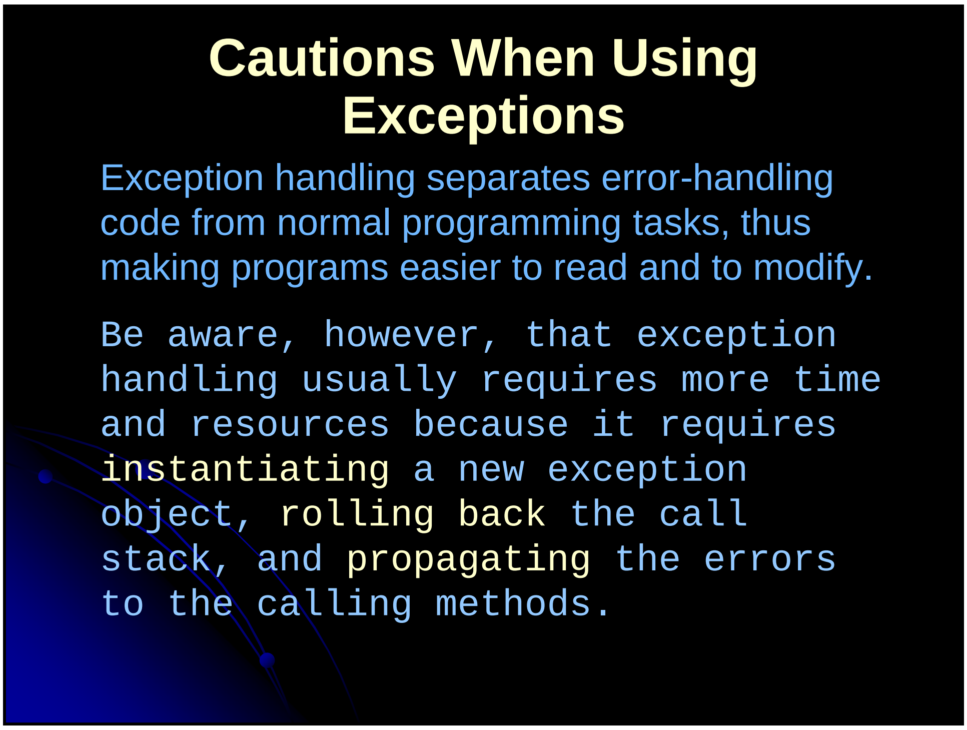
```
try {
  ...
} catch (SomeException someVar) {
  // Do something
} finally {
  // Always executed
}
```



Example: Exceptions in GUI Applications

- An error message appears on the console, but the GUI application continues running
- Re-run the MenuDemo applet from former Example and divide by 0 to see how a GUI deals with unhandled exceptions.

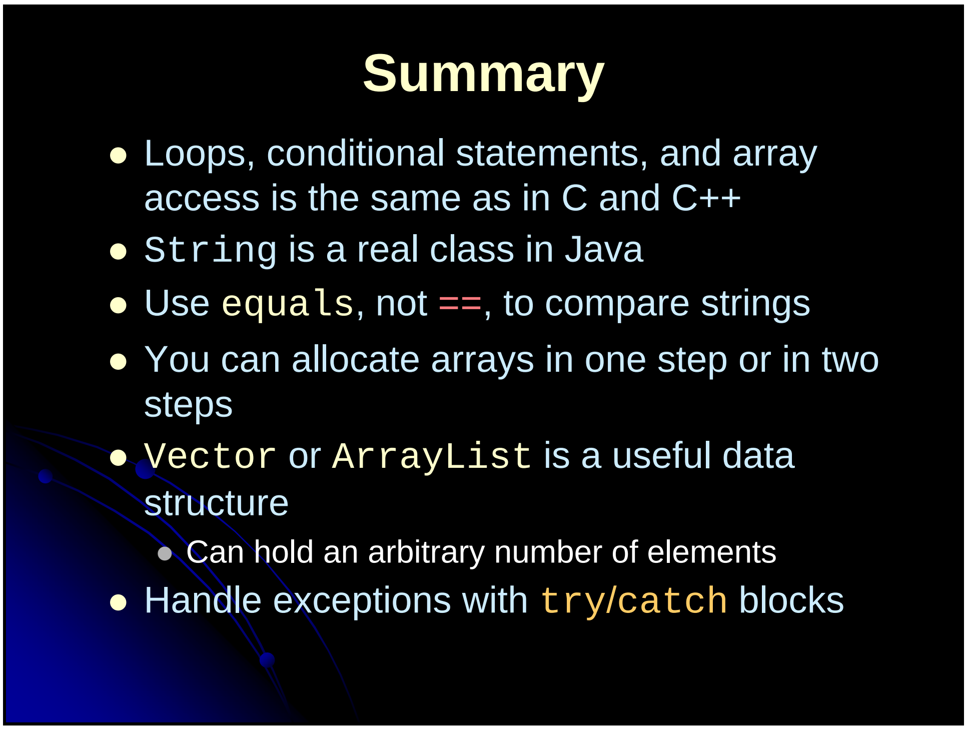
MenuDemo
Run



Cautions When Using Exceptions

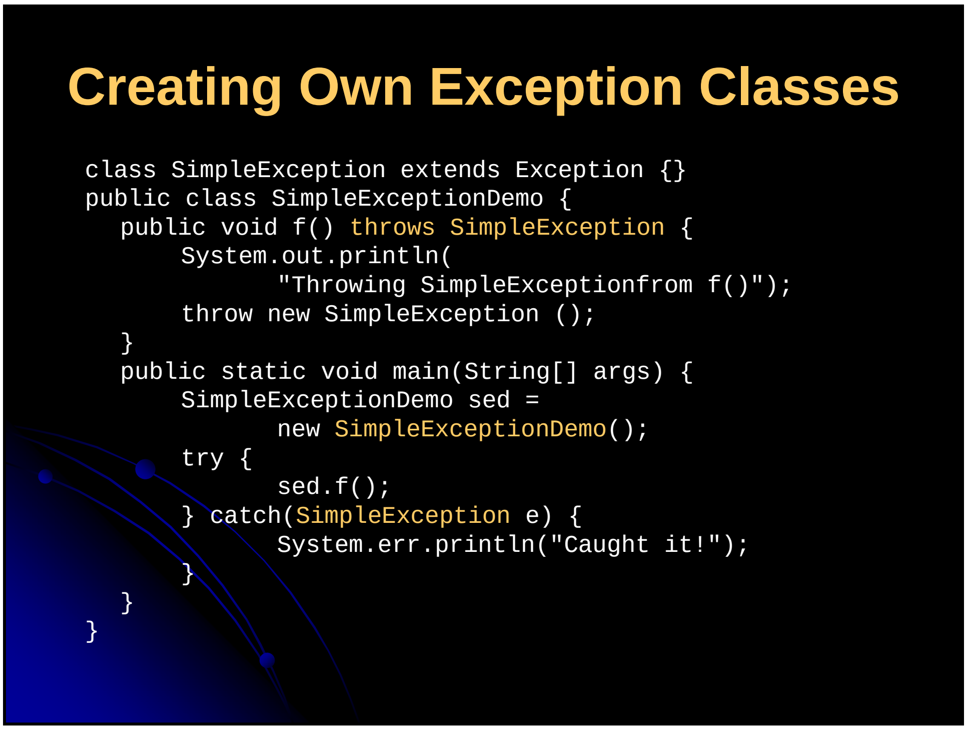
Exception handling separates error-handling code from normal programming tasks, thus making programs easier to read and to modify.

Be aware, however, that exception handling usually requires more time and resources because it requires instantiating a new exception object, rolling back the call stack, and propagating the errors to the calling methods.



Summary

- Loops, conditional statements, and array access is the same as in C and C++
- `String` is a real class in Java
- Use `equals`, not `==`, to compare strings
- You can allocate arrays in one step or in two steps
- `Vector` or `ArrayList` is a useful data structure
  - Can hold an arbitrary number of elements
- Handle exceptions with `try/catch` blocks



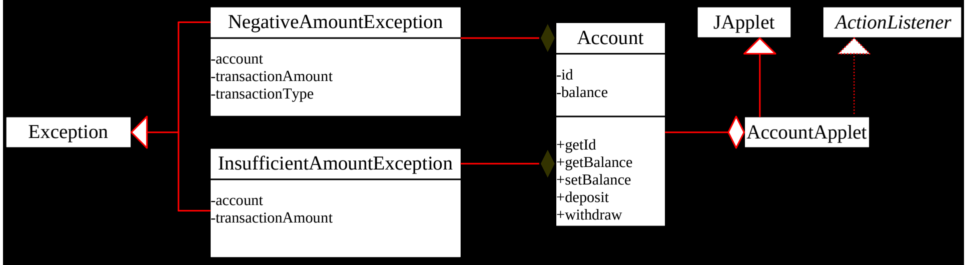
Creating Own Exception Classes

```
class SimpleException extends Exception {}
public class SimpleExceptionDemo {
  public void f() throws SimpleException {
      System.out.println(
             "Throwing SimpleExceptionfrom f()");
      throw new SimpleException ();
  }
  public static void main(String[] args) {
      SimpleExceptionDemo sed =
             new SimpleExceptionDemo();
      try {
             sed.f();
      } catch(SimpleException e) {
             System.err.println("Caught it!");
      }
  }
}
```

Example (Optional): Creating Your Own Exception Classes

- Objective: This program creates a Java applet for handling account transactions. The applet displays the account id and balance, and lets the user deposit to or withdraw from the account. For each transaction, a message is displayed to indicate the status of the transaction: successful or failed. In case of failure, the failure reason is reported.

Example, cont.



NegativeAmountException
-account
-transactionAmount
-transactionType

InsufficientAmountException
-account
-transactionAmount

Exception

Account
-id
-balance
+getId
+getBalance
+setBalance
+deposit
+withdraw

JApplet

ActionListener

AccountApplet

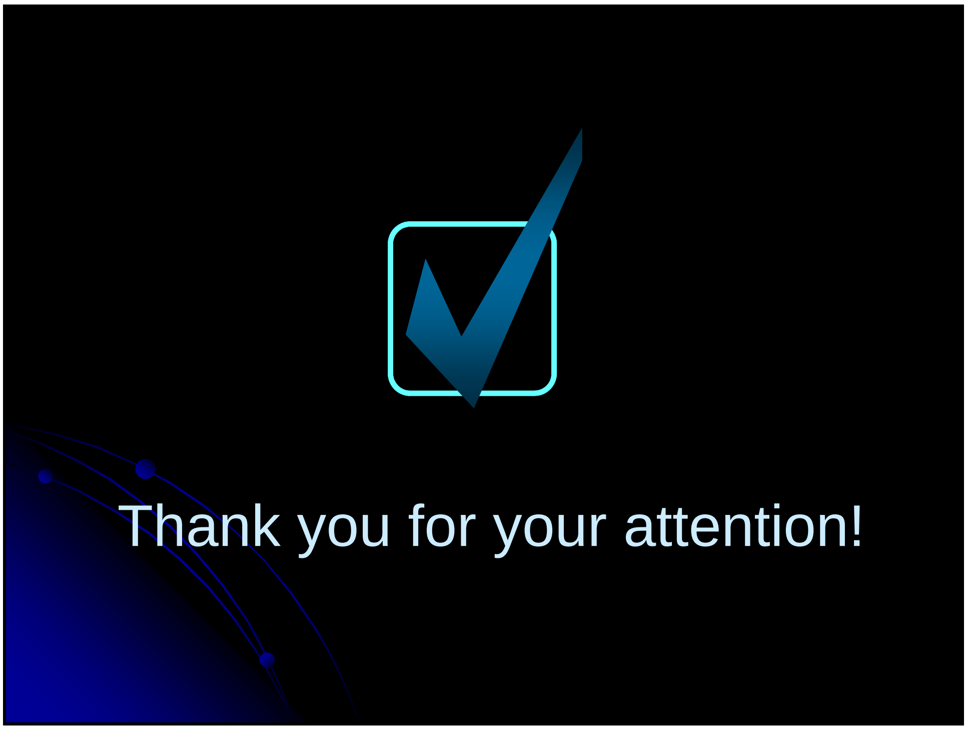
InsufficientFundException

Account

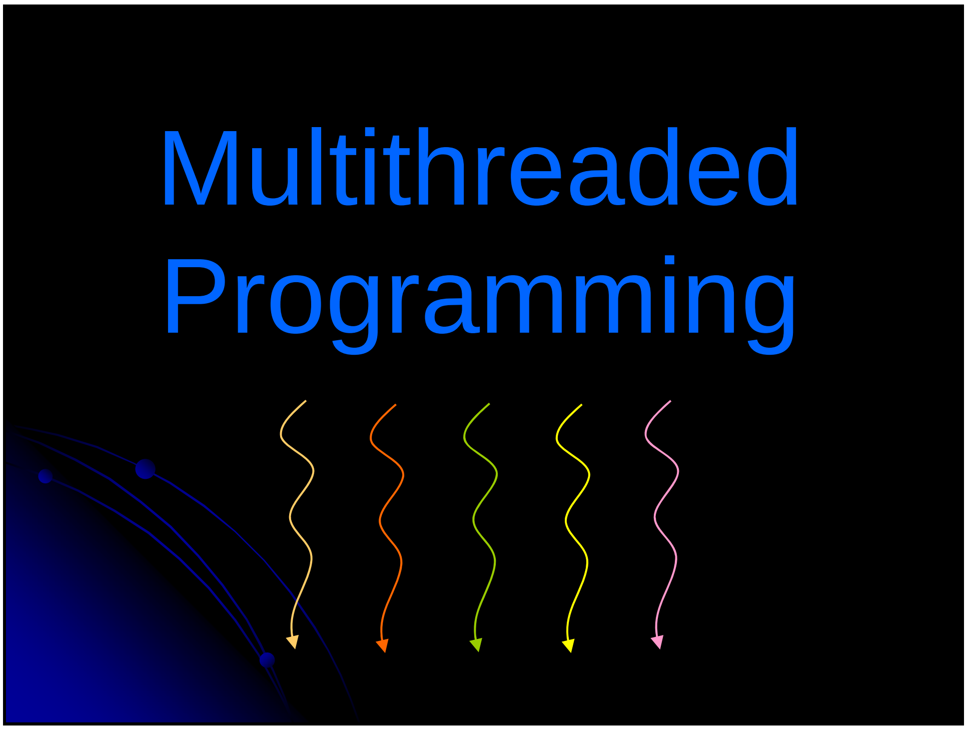
NegativeAmountException

AccountApplet

Run



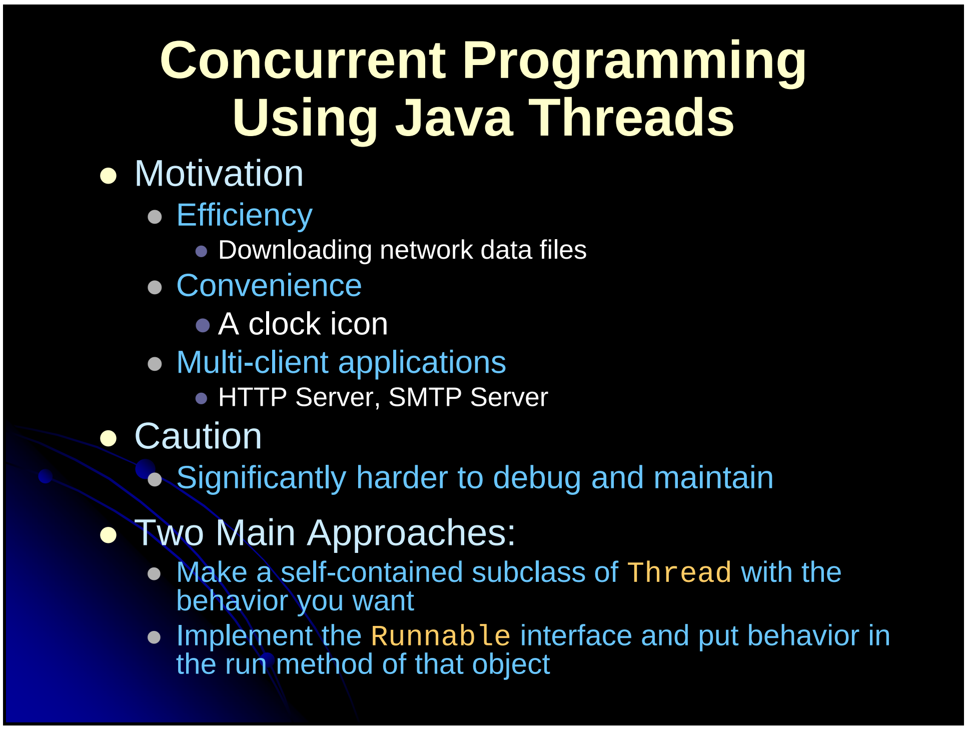
Thank you for your attention!



Multithreaded Programming

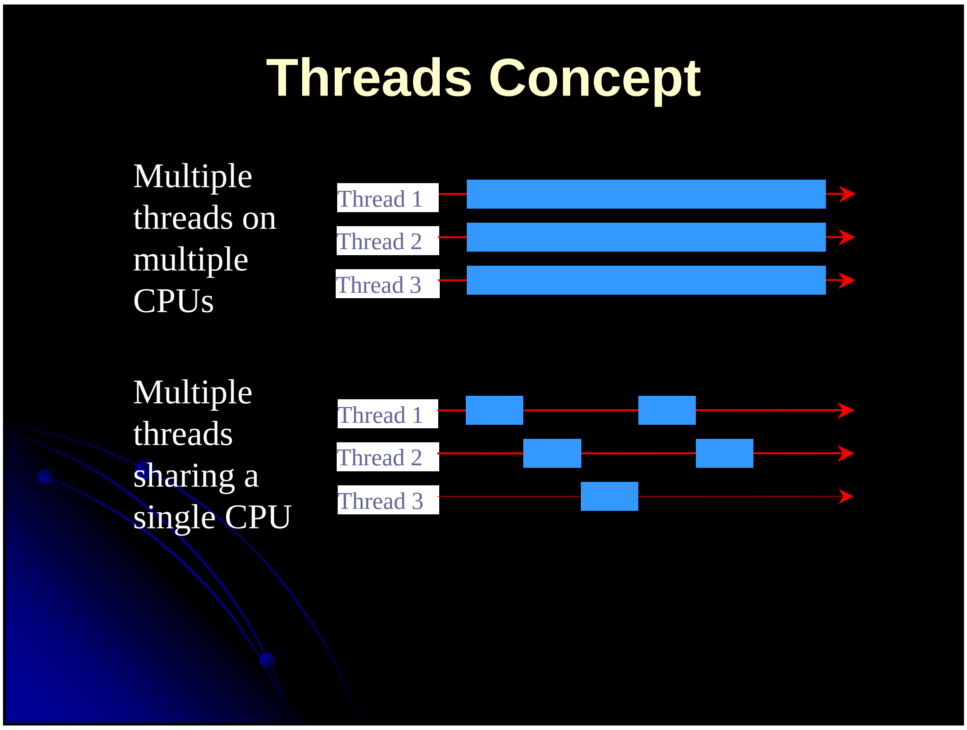
Agenda

- Why threads?
- Approaches for starting threads
  - Separate class approach
  - Callback approach
- Solving common thread problems
- Synchronizing access to shared resources
- Thread life cycle
- Stopping threads



Concurrent Programming Using Java Threads

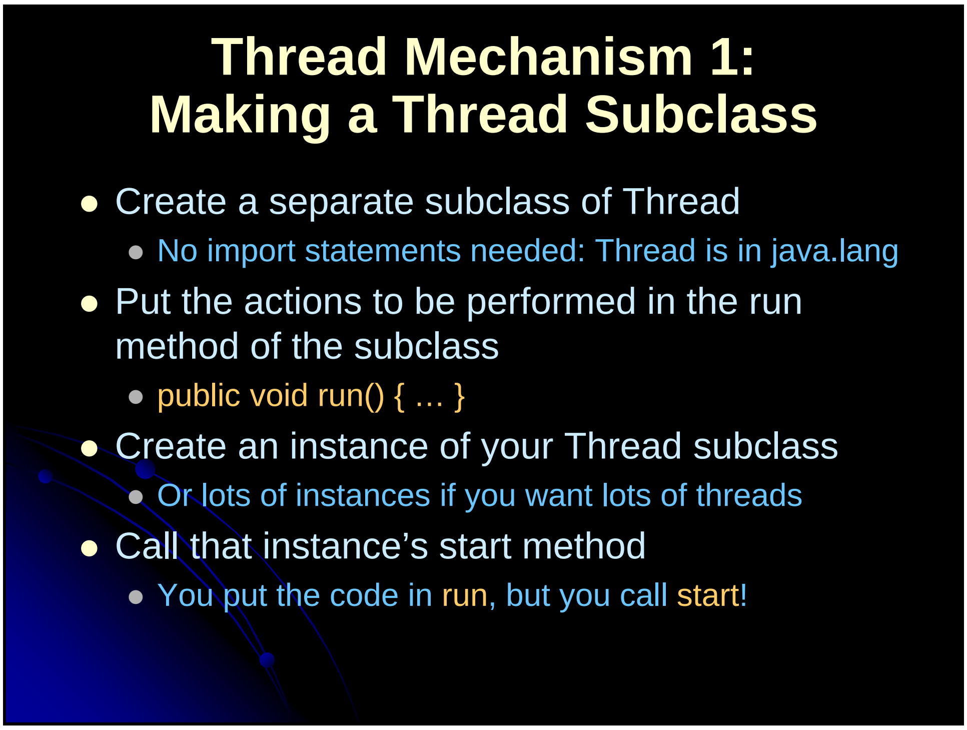
- Motivation
  - Efficiency
    - Downloading network data files
  - Convenience
    - A clock icon
  - Multi-client applications
    - HTTP Server, SMTP Server
- Caution
  - Significantly harder to debug and maintain
- Two Main Approaches:
  - Make a self-contained subclass of `Thread` with the behavior you want
  - Implement the `Runnable` interface and put behavior in the run method of that object



Threads Concept

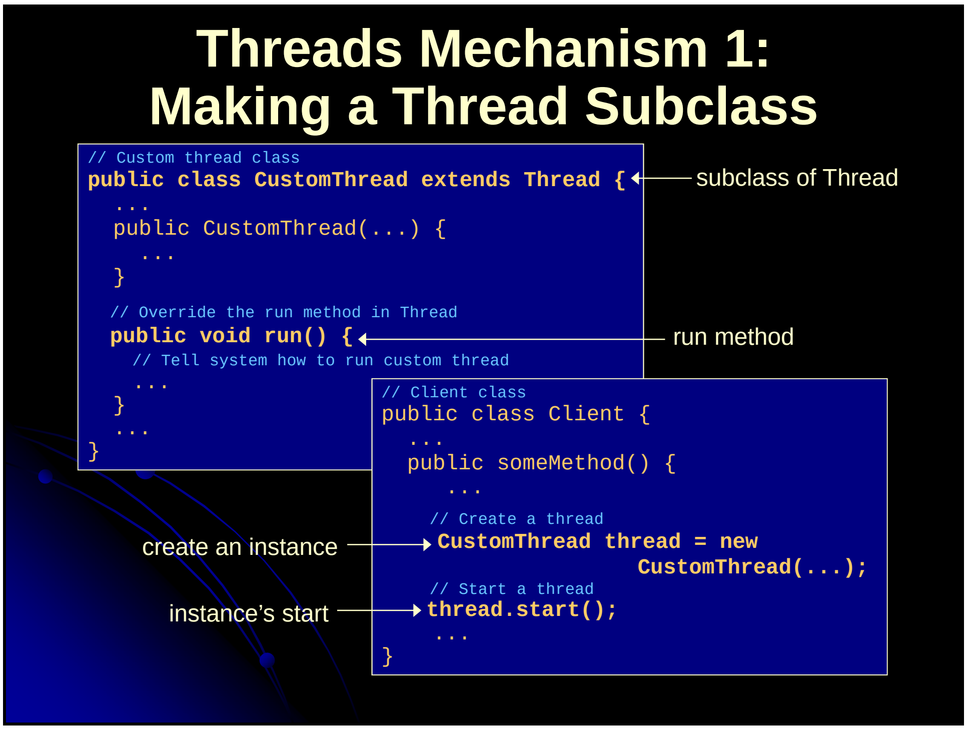
Multiple threads on multiple CPUs

Multiple threads sharing a single CPU



Thread Mechanism 1: Making a Thread Subclass

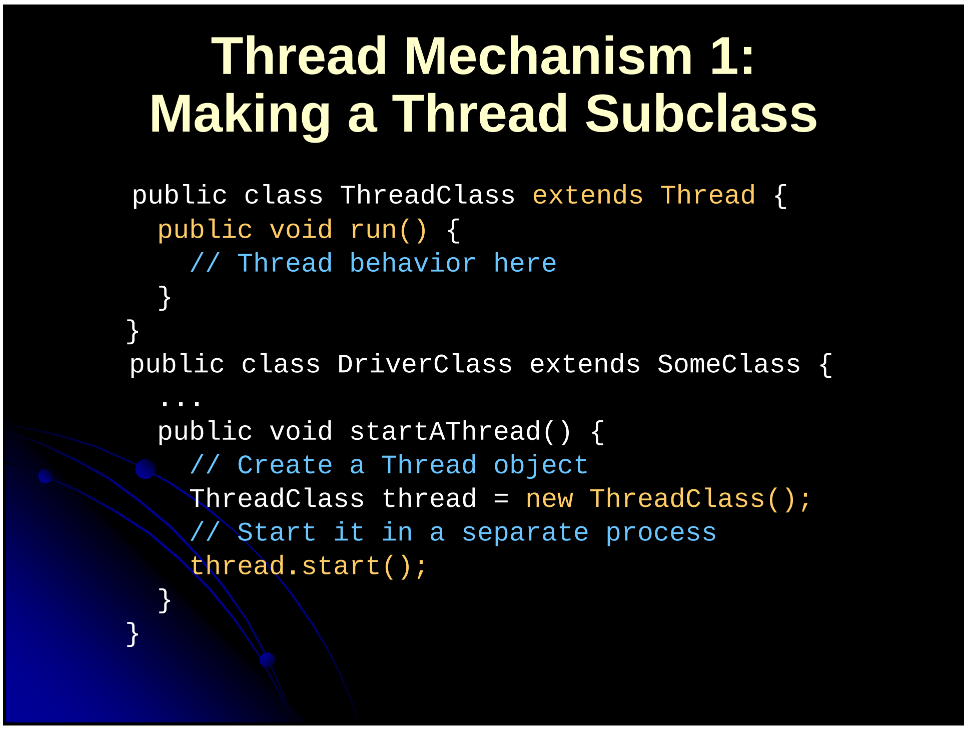
- Create a separate subclass of Thread
  - No import statements needed: Thread is in java.lang
- Put the actions to be performed in the run method of the subclass
  - public void run() { … }
- Create an instance of your Thread subclass
  - Or lots of instances if you want lots of threads
- Call that instance's start method
  - You put the code in run, but you call start!



Threads Mechanism 1: Making a Thread Subclass

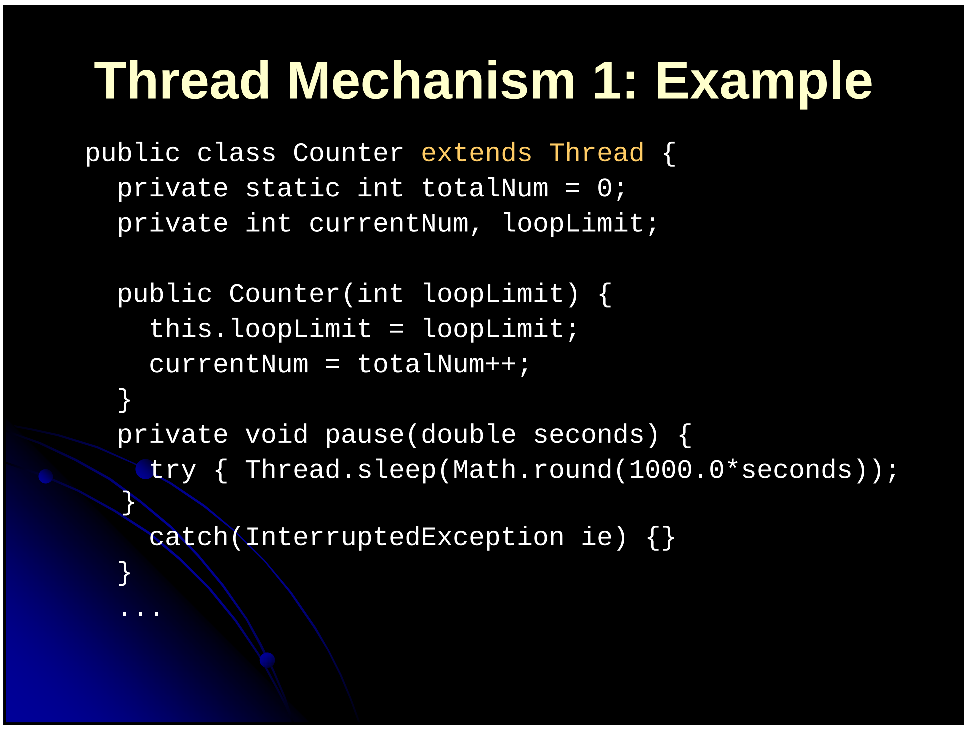
```
// Custom thread class
public class CustomThread extends Thread {
  ...
  public CustomThread(...) {
    ...
  }
  // Override the run method in Thread
  public void run() {
    // Tell system how to run custom thread
    ...
  }
  ...
}
```

```
// Client class
public class Client {
  ...
  public someMethod() {
    ...
    // Create a thread
    CustomThread thread = new
                    CustomThread(...);
    // Start a thread
    thread.start();
    ...
}
```



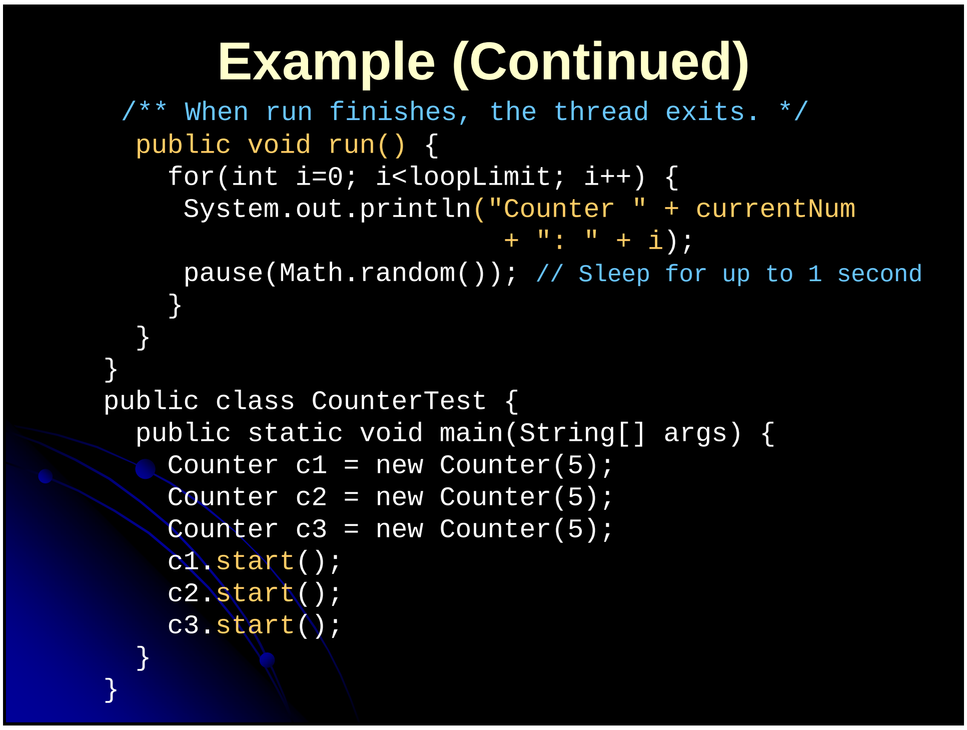
Thread Mechanism 1: Making a Thread Subclass

```
public class ThreadClass extends Thread {
  public void run() {
    // Thread behavior here
  }
}
public class DriverClass extends SomeClass {
  ...
  public void startAThread() {
    // Create a Thread object
    ThreadClass thread = new ThreadClass();
    // Start it in a separate process
    thread.start();
  }
}
```



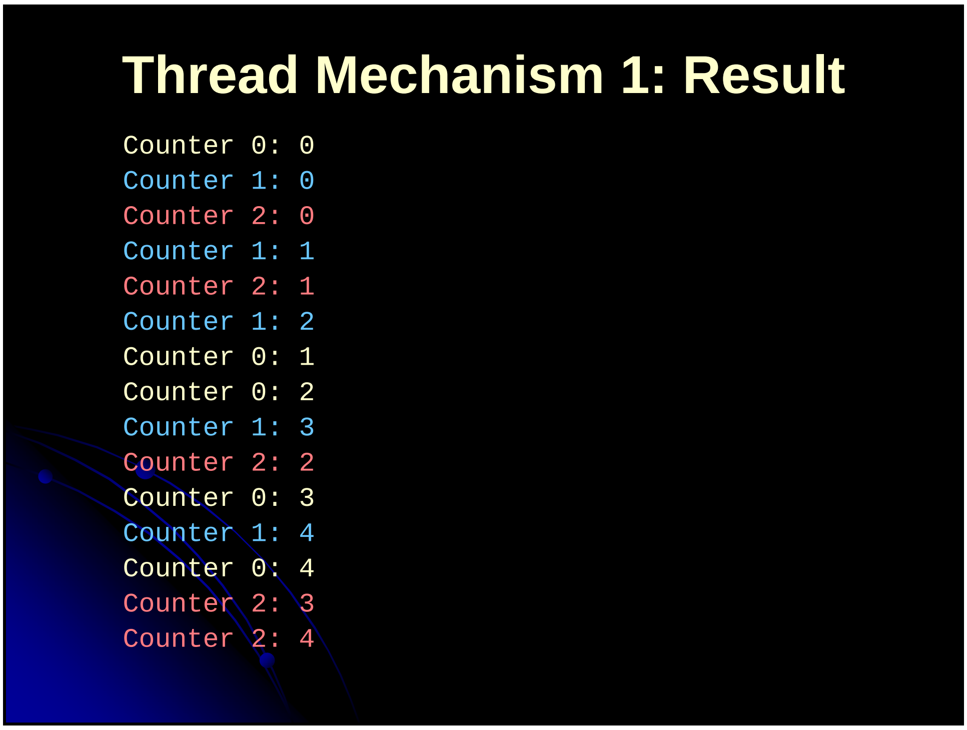
Thread Mechanism 1: Example

```
public class Counter extends Thread {
  private static int totalNum = 0;
  private int currentNum, loopLimit;

  public Counter(int loopLimit) {
    this.loopLimit = loopLimit;
    currentNum = totalNum++;
  }
  private void pause(double seconds) {
    try { Thread.sleep(Math.round(1000.0*seconds));
  }
    catch(InterruptedException ie) {}
  }
  ...
```



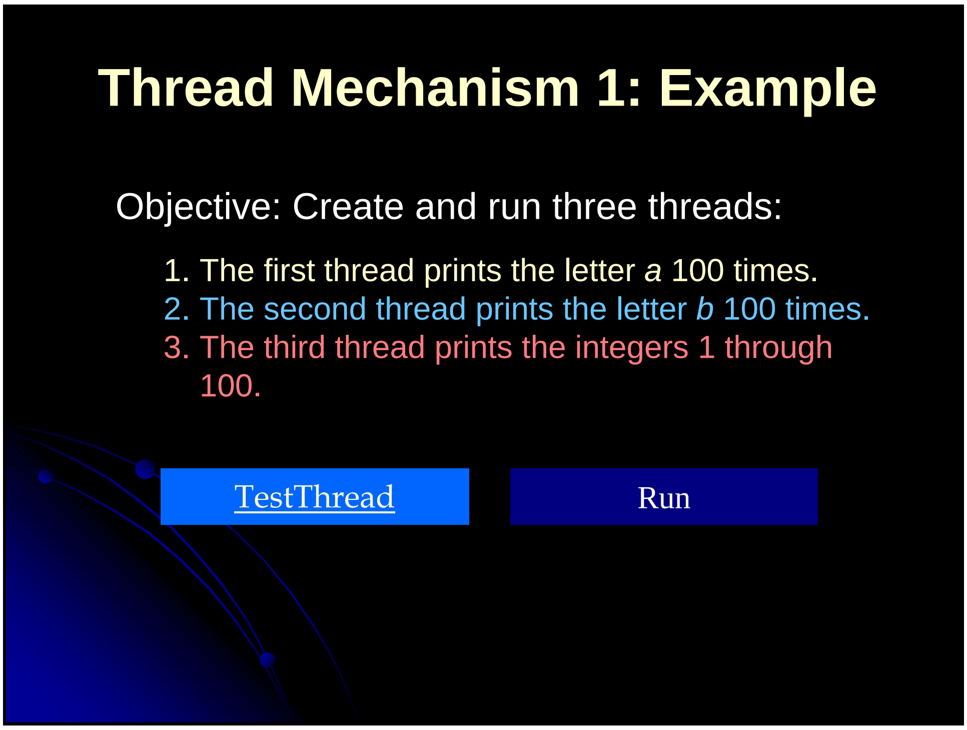
Example (Continued)

```
 /** When run finishes, the thread exits. */
  public void run() {
    for(int i=0; i<loopLimit; i++) {
     System.out.println("Counter " + currentNum
                         + ": " + i);
     pause(Math.random()); // Sleep for up to 1 second
    }
  }
}
public class CounterTest {
  public static void main(String[] args) {
    Counter c1 = new Counter(5);
    Counter c2 = new Counter(5);
    Counter c3 = new Counter(5);
    c1.start();
    c2.start();
    c3.start();
  }
}
```



Thread Mechanism 1: Result

```
Counter 0: 0
Counter 1: 0
Counter 2: 0
Counter 1: 1
Counter 2: 1
Counter 1: 2
Counter 0: 1
Counter 0: 2
Counter 1: 3
Counter 2: 2
Counter 0: 3
Counter 1: 4
Counter 0: 4
Counter 2: 3
Counter 2: 4
```



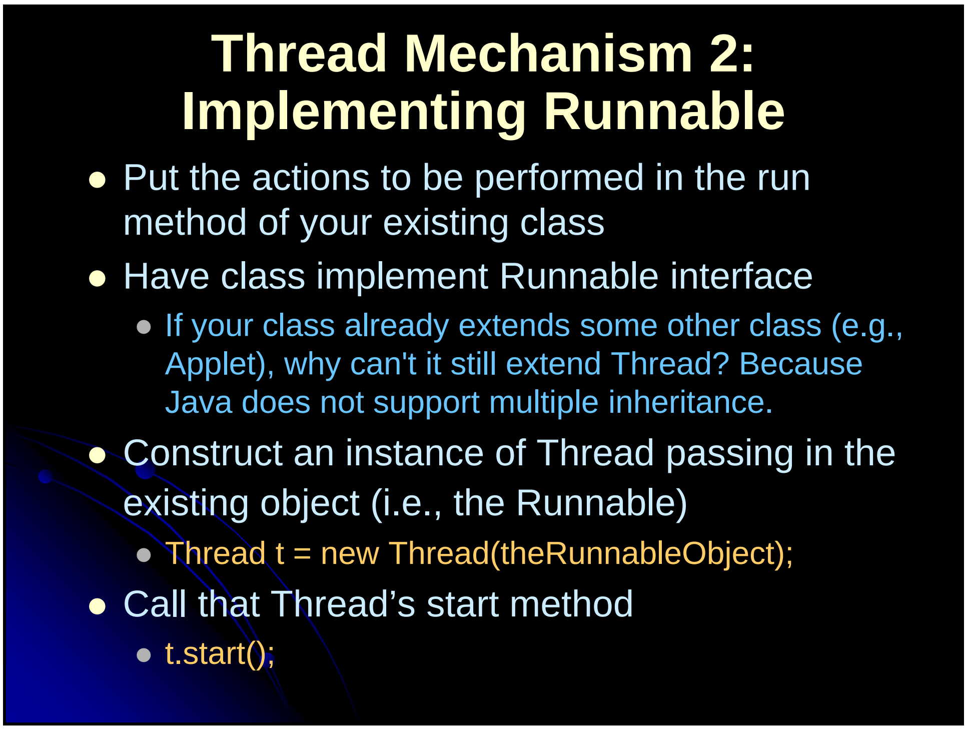
Thread Mechanism 1: Example

Objective: Create and run three threads:

1. The first thread prints the letter *a* 100 times.
2. The second thread prints the letter *b* 100 times.
3. The third thread prints the integers 1 through 100.

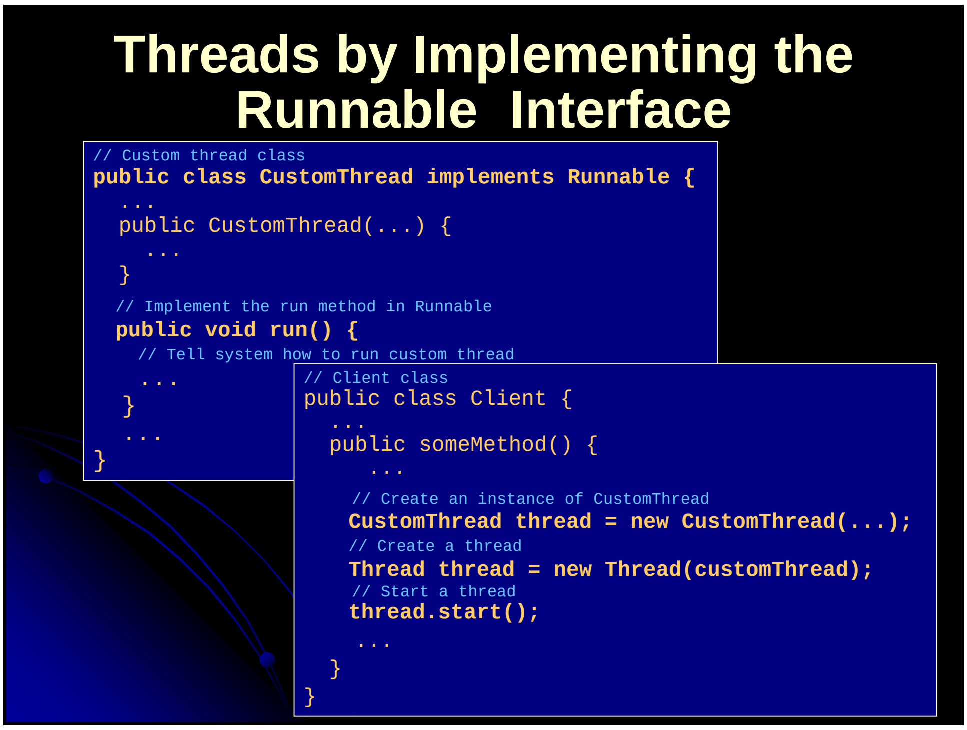
TestThread

Run



Thread Mechanism 2: Implementing Runnable

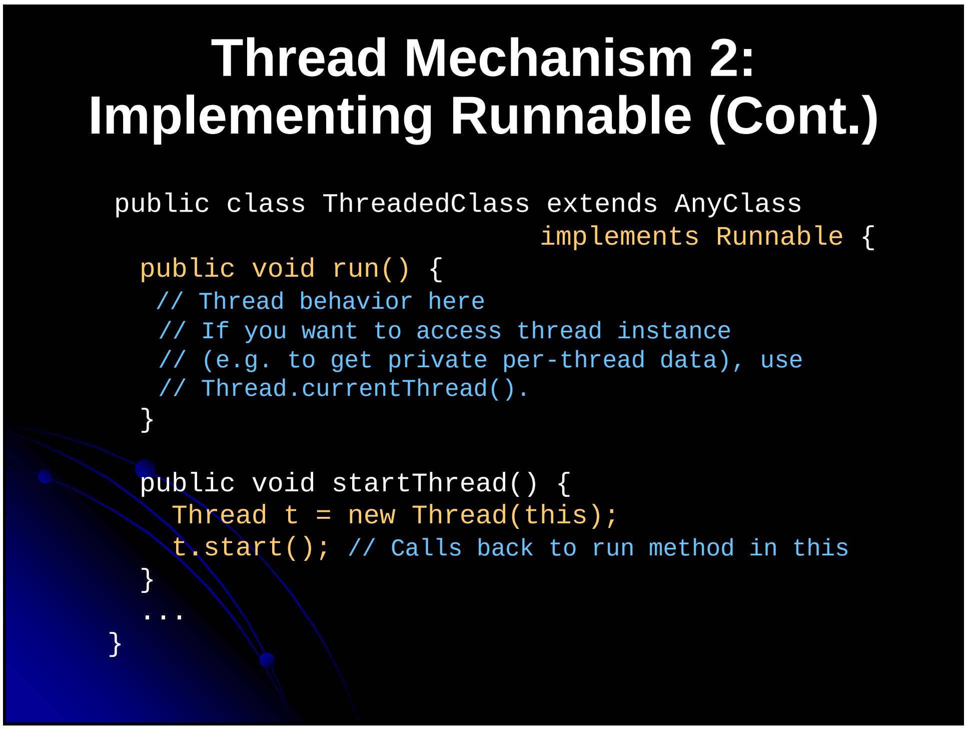
- Put the actions to be performed in the run method of your existing class
- Have class implement Runnable interface
  - If your class already extends some other class (e.g., Applet), why can't it still extend Thread? Because Java does not support multiple inheritance.
- Construct an instance of Thread passing in the existing object (i.e., the Runnable)
  - Thread t = new Thread(theRunnableObject);
- Call that Thread’s start method
  - t.start();



Threads by Implementing the Runnable Interface

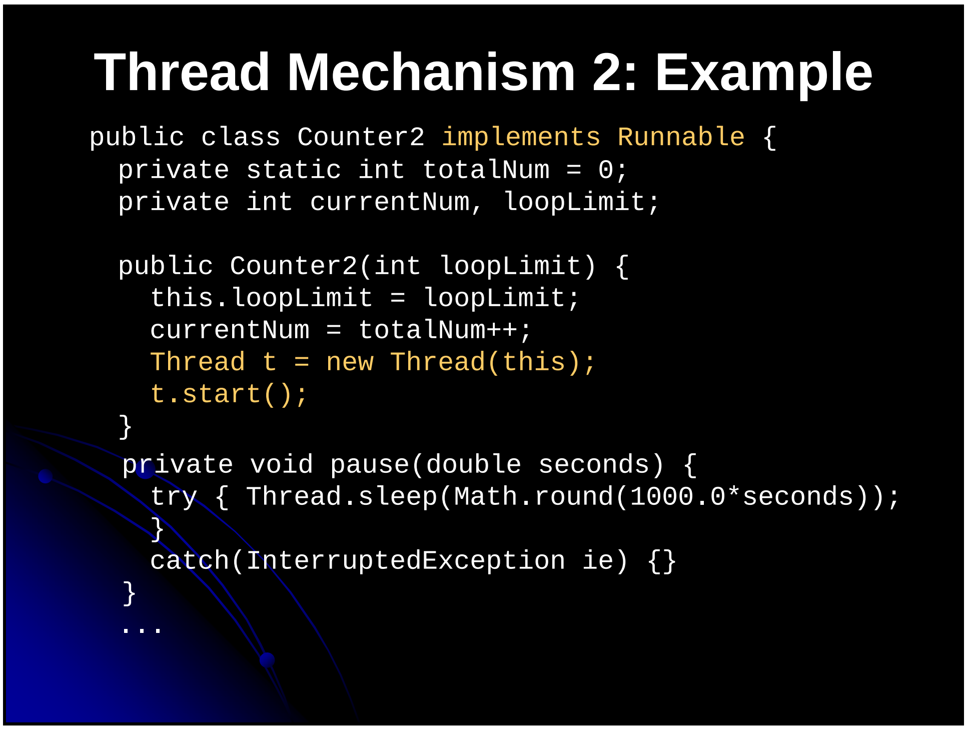
```
// Custom thread class
public class CustomThread implements Runnable {
  ...
  public CustomThread(...) {
    ...
  }
  // Implement the run method in Runnable
  public void run() {
    // Tell system how to run custom thread
    ...
  }
  ...
}
```

```
// Client class
public class Client {
  ...
  public someMethod() {
    ...
    // Create an instance of CustomThread
    CustomThread thread = new CustomThread(...);
    // Create a thread
    Thread thread = new Thread(customThread);
    // Start a thread
    thread.start();
    ...
  }
}
```



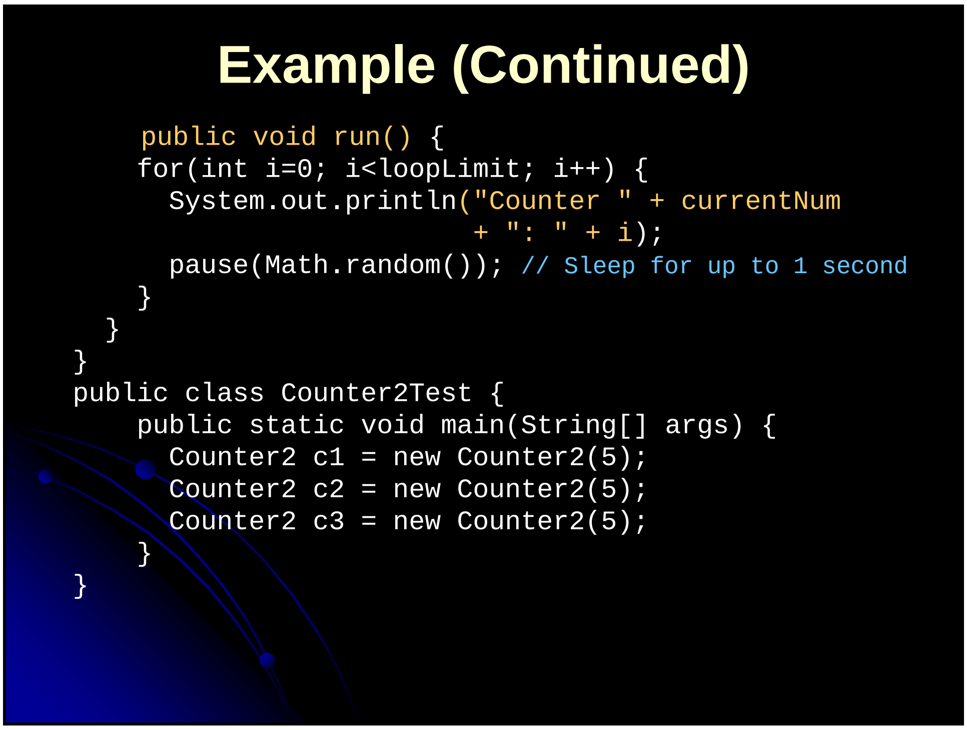
Thread Mechanism 2: Implementing Runnable (Cont.)

```
public class ThreadedClass extends AnyClass
                                  implements Runnable {
  public void run() {
    // Thread behavior here
    // If you want to access thread instance
    // (e.g. to get private per-thread data), use
    // Thread.currentThread().
  }

  public void startThread() {
    Thread t = new Thread(this);
    t.start(); // Calls back to run method in this
  }
  ...
}
```



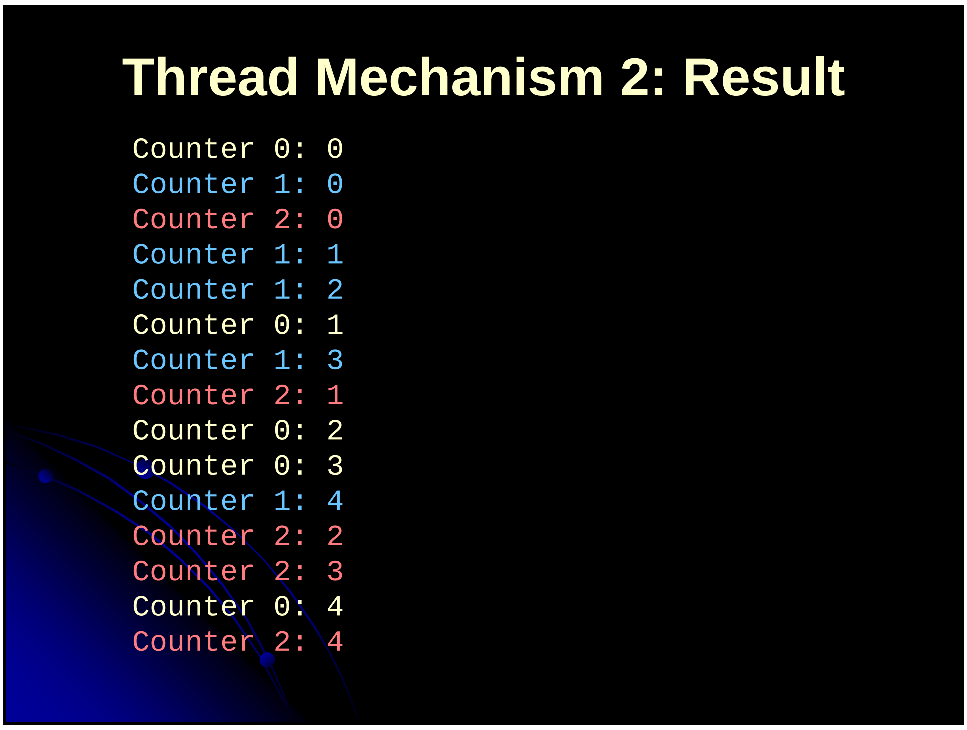
Thread Mechanism 2: Example

```
public class Counter2 implements Runnable {
  private static int totalNum = 0;
  private int currentNum, loopLimit;

  public Counter2(int loopLimit) {
    this.loopLimit = loopLimit;
    currentNum = totalNum++;
    Thread t = new Thread(this);
    t.start();
  }
  private void pause(double seconds) {
    try { Thread.sleep(Math.round(1000.0*seconds));
    }
    catch(InterruptedException ie) {}
  }
  ...
```



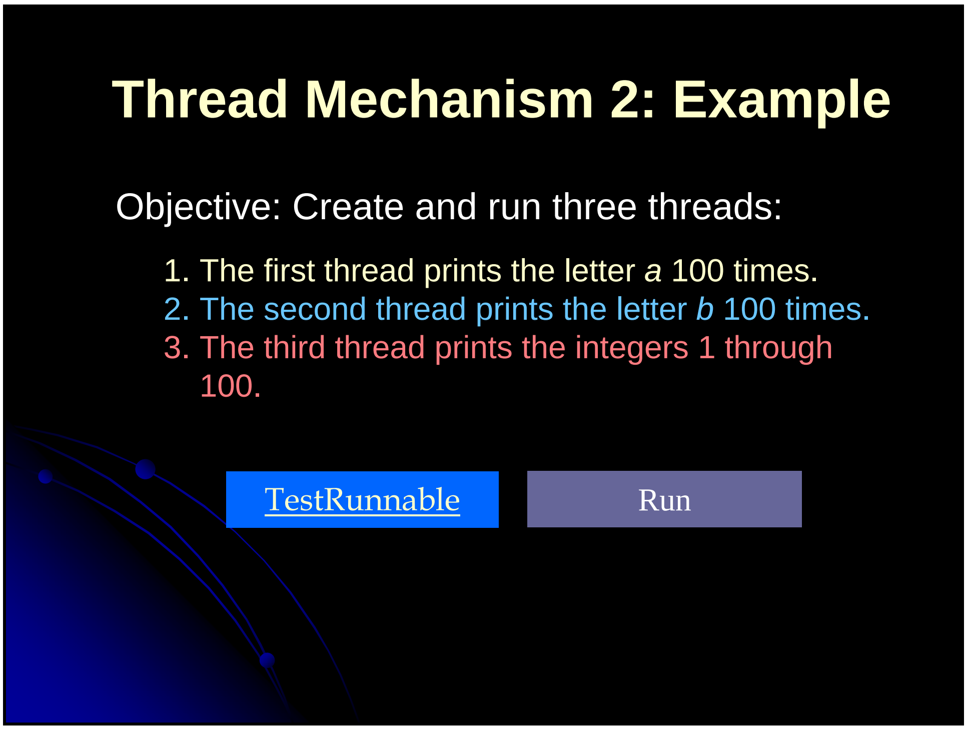
Example (Continued)

```
    public void run() {
    for(int i=0; i<loopLimit; i++) {
      System.out.println("Counter " + currentNum
                         + ": " + i);
      pause(Math.random()); // Sleep for up to 1 second
    }
  }
}
public class Counter2Test {
    public static void main(String[] args) {
      Counter2 c1 = new Counter2(5);
      Counter2 c2 = new Counter2(5);
      Counter2 c3 = new Counter2(5);
    }
}
```



Thread Mechanism 2: Result

```
Counter 0: 0
Counter 1: 0
Counter 2: 0
Counter 1: 1
Counter 1: 2
Counter 0: 1
Counter 1: 3
Counter 2: 1
Counter 0: 2
Counter 0: 3
Counter 1: 4
Counter 2: 2
Counter 2: 3
Counter 0: 4
Counter 2: 4
```



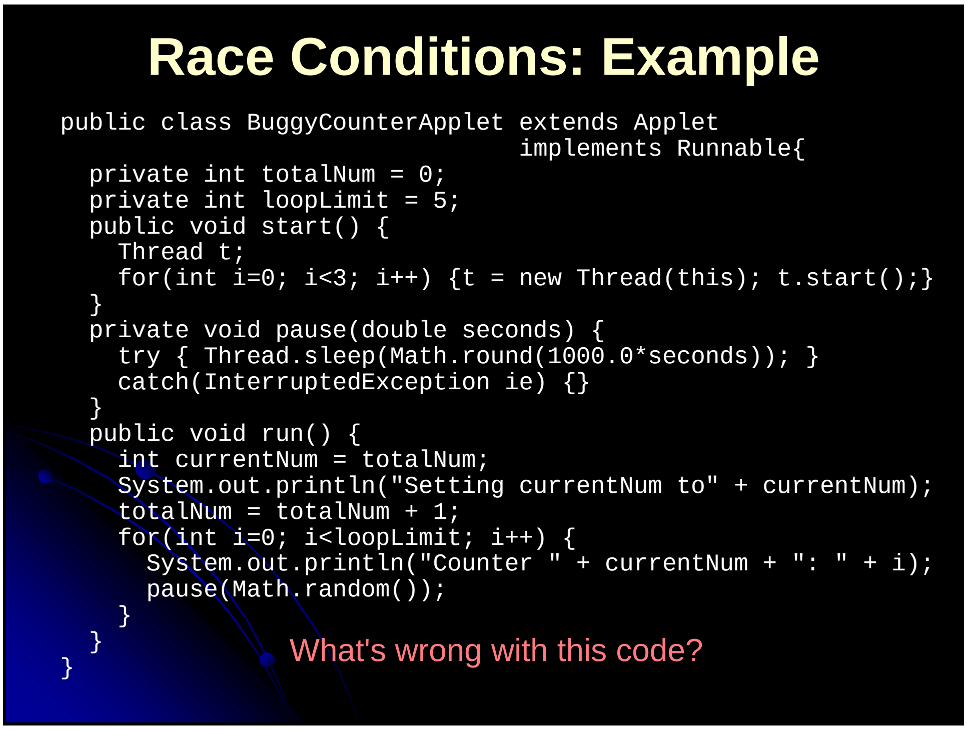
Thread Mechanism 2: Example

Objective: Create and run three threads:

1. The first thread prints the letter *a* 100 times.
2. The second thread prints the letter *b* 100 times.
3. The third thread prints the integers 1 through 100.

TestRunnable

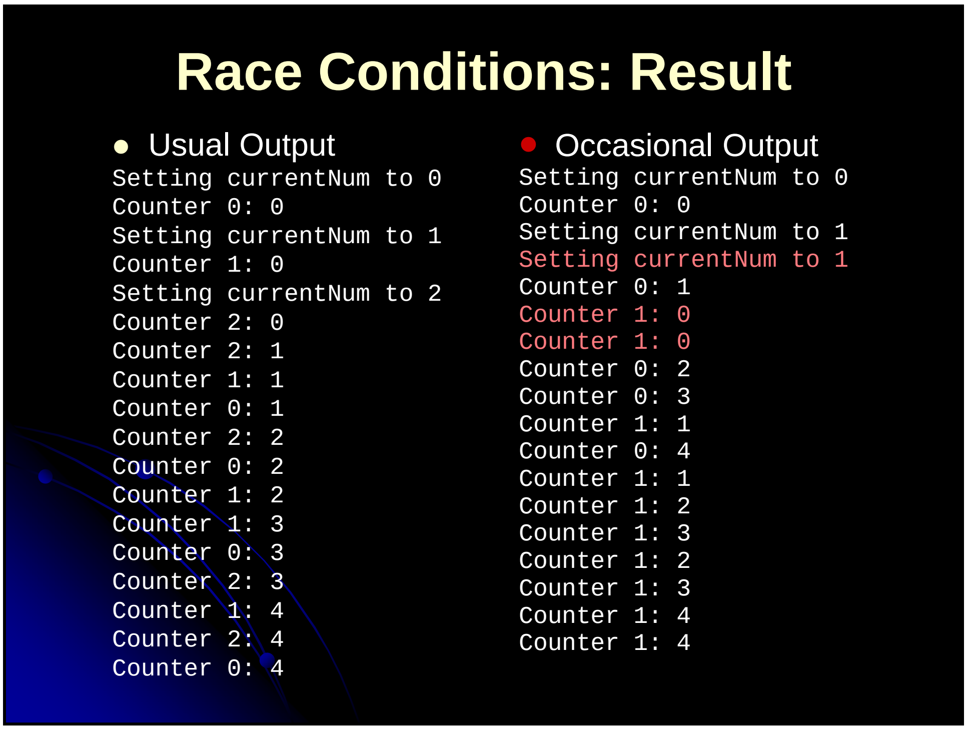
Run



Race Conditions: Example

```
public class BuggyCounterApplet extends Applet
                                implements Runnable{
  private int totalNum = 0;
  private int loopLimit = 5;
  public void start() {
    Thread t;
    for(int i=0; i<3; i++) {t = new Thread(this); t.start();}
  }
  private void pause(double seconds) {
    try { Thread.sleep(Math.round(1000.0*seconds)); }
    catch(InterruptedException ie) {}
  }
  public void run() {
    int currentNum = totalNum;
    System.out.println("Setting currentNum to" + currentNum);
    totalNum = totalNum + 1;
    for(int i=0; i<loopLimit; i++) {
      System.out.println("Counter " + currentNum + ": " + i);
      pause(Math.random());
    }
  }
}
```

What's wrong with this code?



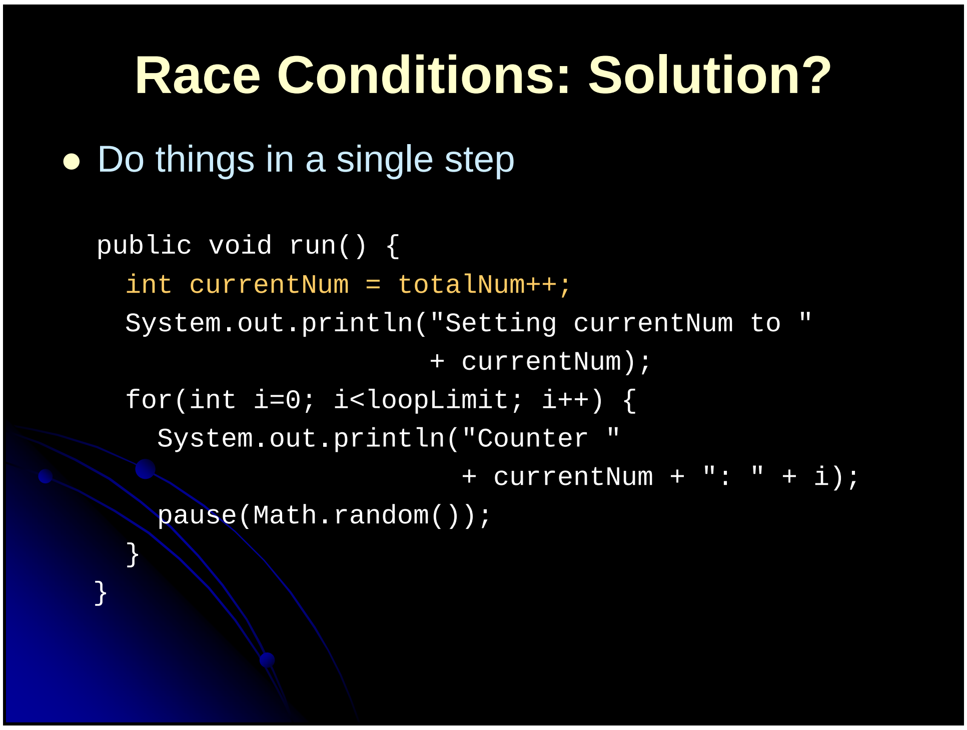
Race Conditions: Result

- Usual Output

```
Setting currentNum to 0
Counter 0: 0
Setting currentNum to 1
Counter 1: 0
Setting currentNum to 2
Counter 2: 0
Counter 2: 1
Counter 1: 1
Counter 0: 1
Counter 2: 2
Counter 0: 2
Counter 1: 2
Counter 1: 3
Counter 0: 3
Counter 2: 3
Counter 1: 4
Counter 2: 4
Counter 0: 4
```

- Occasional Output

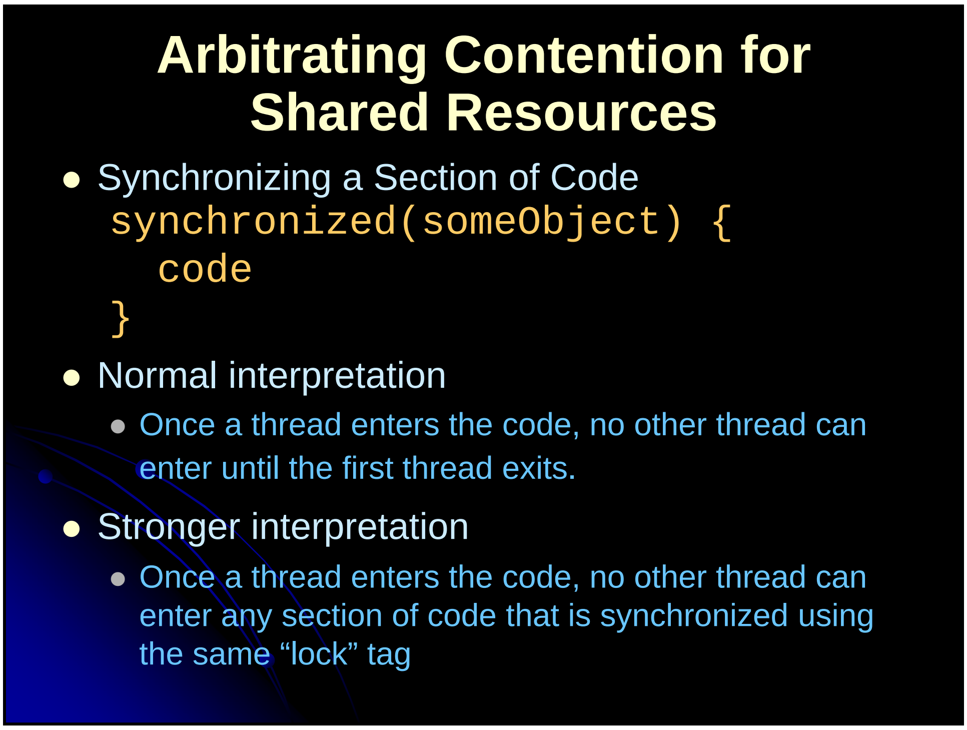
```
Setting currentNum to 0
Counter 0: 0
Setting currentNum to 1
Setting currentNum to 1
Counter 0: 1
Counter 1: 0
Counter 1: 0
Counter 0: 2
Counter 0: 3
Counter 1: 1
Counter 0: 4
Counter 1: 1
Counter 1: 2
Counter 1: 3
Counter 1: 2
Counter 1: 3
Counter 1: 4
Counter 1: 4
```



Race Conditions: Solution?

- Do things in a single step

```
public void run() {
  int currentNum = totalNum++;
  System.out.println("Setting currentNum to "
                     + currentNum);
  for(int i=0; i<loopLimit; i++) {
    System.out.println("Counter "
                       + currentNum + ": " + i);
    pause(Math.random());
  }
}
```

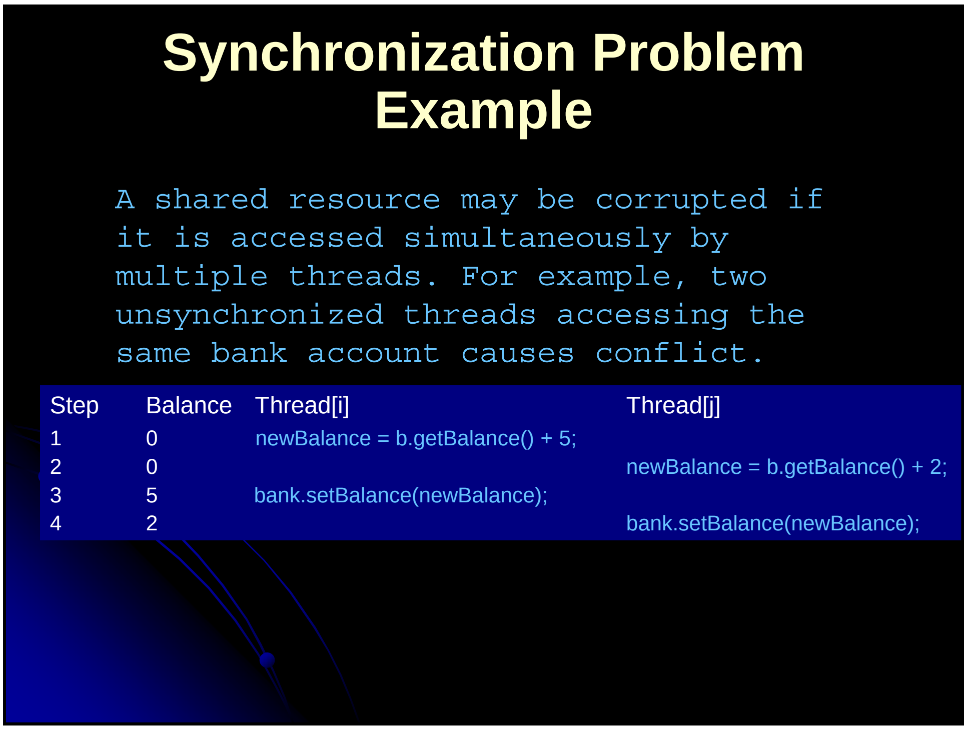


Arbitrating Contention for Shared Resources

- Synchronizing a Section of Code

```
synchronized(someObject) {
   code
}
```

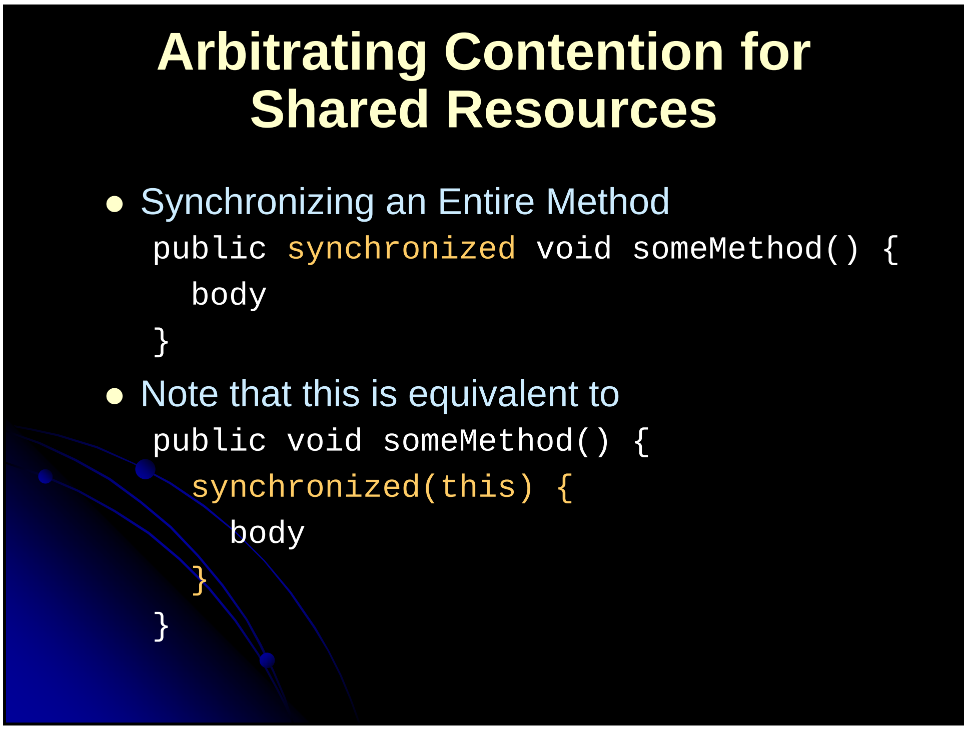
- Normal interpretation
  - Once a thread enters the code, no other thread can enter until the first thread exits.
- Stronger interpretation
  - Once a thread enters the code, no other thread can enter any section of code that is synchronized using the same “lock” tag



Synchronization Problem Example

A shared resource may be corrupted if it is accessed simultaneously by multiple threads. For example, two unsynchronized threads accessing the same bank account causes conflict.

Step	Balance	Thread[i]	Thread[j]
1	0	newBalance = b.getBalance() + 5;	
2	0		newBalance = b.getBalance() + 2;
3	5	bank.setBalance(newBalance);	
4	2		bank.setBalance(newBalance);



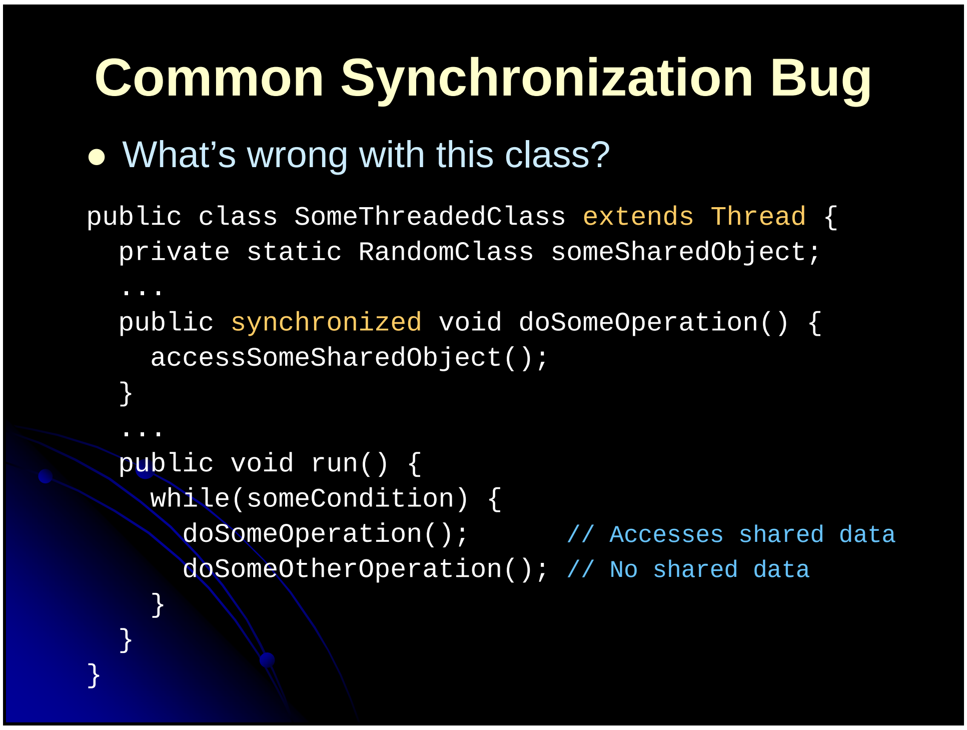
Arbitrating Contention for Shared Resources

- Synchronizing an Entire Method

```
public synchronized void someMethod() {
  body
}
```

- Note that this is equivalent to

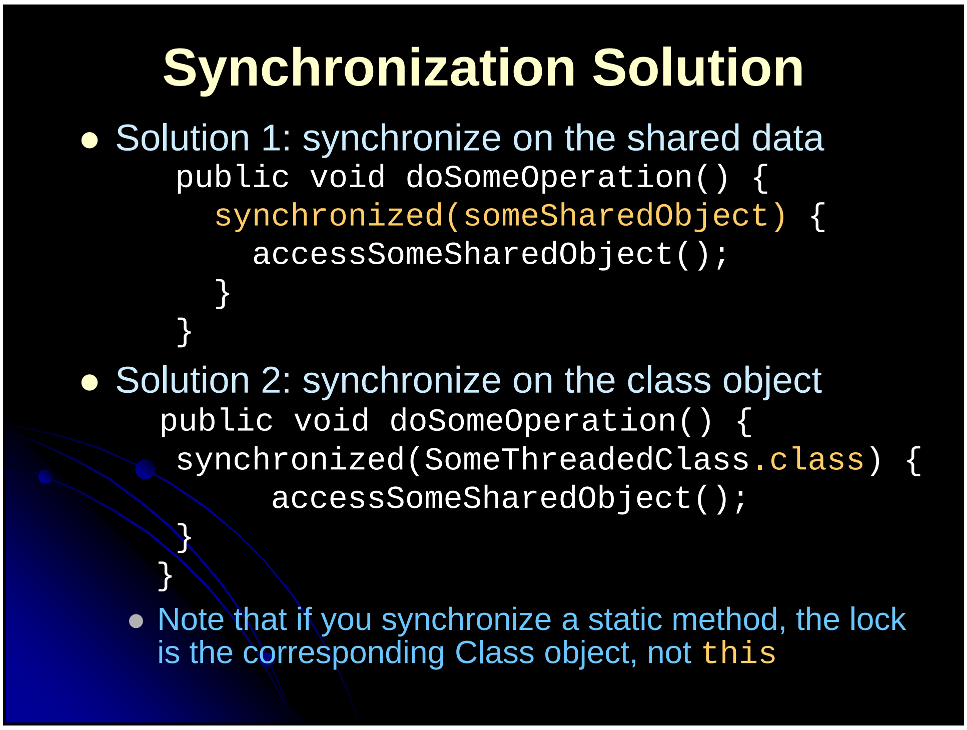
```
public void someMethod() {
  synchronized(this) {
    body
  }
}
```



Common Synchronization Bug

- What's wrong with this class?

```
public class SomeThreadedClass extends Thread {
  private static RandomClass someSharedObject;
  ...
  public synchronized void doSomeOperation() {
    accessSomeSharedObject();
  }
  ...
  public void run() {
    while(someCondition) {
      doSomeOperation();      // Accesses shared data
      doSomeOtherOperation(); // No shared data
    }
  }
}
```



Synchronization Solution

- Solution 1: synchronize on the shared data

```
public void doSomeOperation() {
  synchronized(someSharedObject) {
    accessSomeSharedObject();
  }
}
```

- Solution 2: synchronize on the class object

```
public void doSomeOperation() {
 synchronized(SomeThreadedClass.class) {
      accessSomeSharedObject();
 }
}
```

  - Note that if you synchronize a static method, the lock is the corresponding Class object, not `this`

Synchronization Solution (Continued)

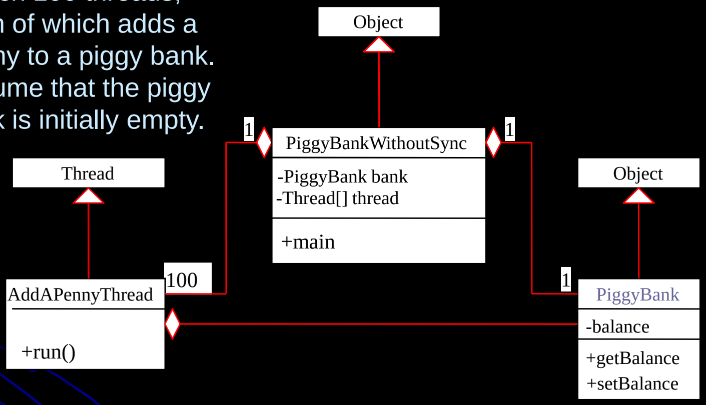
- Solution 3: synchronize on arbitrary object

```
public class SomeThreadedClass extends
  Thread {
   private static Object lockObject
     = new Object();
   ...
   public void doSomeOperation() {
     synchronized(lockObject) {
       accessSomeSharedObject();
     }
   }
   ...
```

  - Why doesn't this problem usually occur with Runnable?

Example

Objective: create and launch 100 threads, each of which adds a penny to a piggy bank. Assume that the piggy bank is initially empty.



PiggyBankWithoutSync

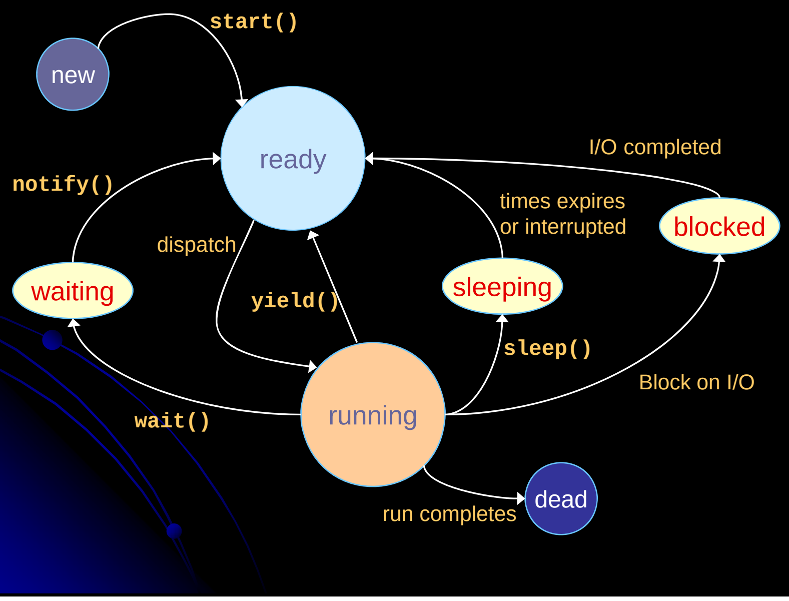
Run

With synchronize

PiggyBankWithSync

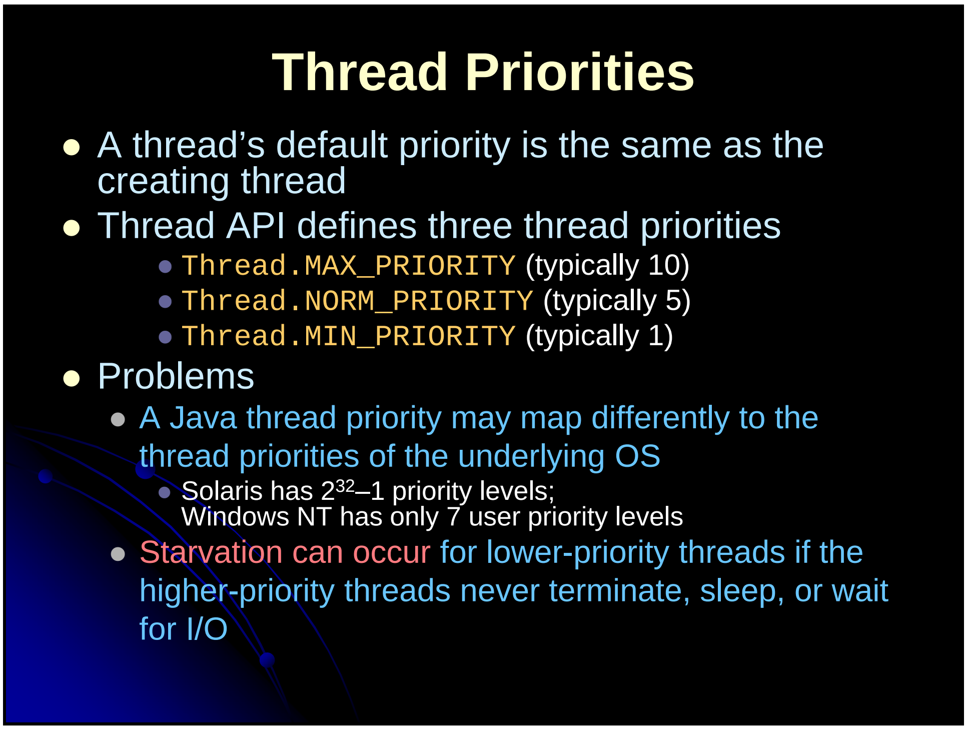
Run

Thread Lifecycle



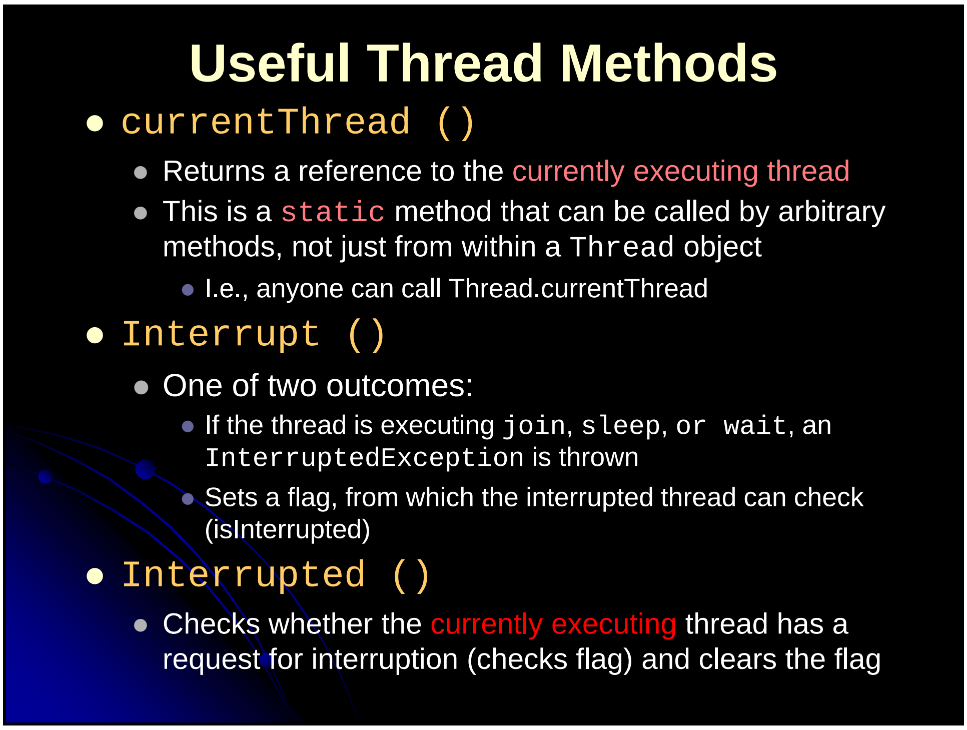
Useful Thread Constructors

- `Thread()`
  - Default version you get when you call constructor of your custom Thread subclass.
- `Thread(Runnable target)`
  - Creates a thread, that, once started, will execute the `run` method of the target
- `Thread(ThreadGroup group, Runnable target)`
  - Creates a thread and places it in the specified thread group
  - A `ThreadGroup` is a collection of threads that can be operated on as a set
- `Thread(String name)`
  - Creates a thread with the given name
  - Useful for debugging



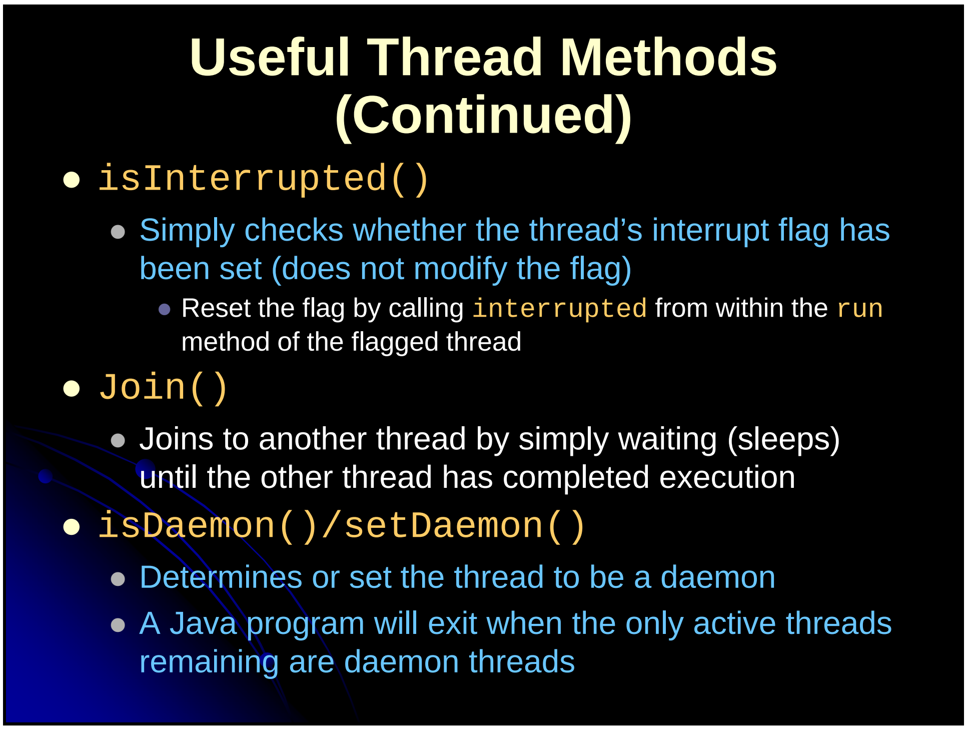
Thread Priorities

- A thread's default priority is the same as the creating thread
- Thread API defines three thread priorities
  - `Thread.MAX_PRIORITY` (typically 10)
  - `Thread.NORM_PRIORITY` (typically 5)
  - `Thread.MIN_PRIORITY` (typically 1)
- Problems
  - A Java thread priority may map differently to the thread priorities of the underlying OS
    - Solaris has 2^{32}–1 priority levels;
Windows NT has only 7 user priority levels
  - Starvation can occur for lower-priority threads if the higher-priority threads never terminate, sleep, or wait for I/O



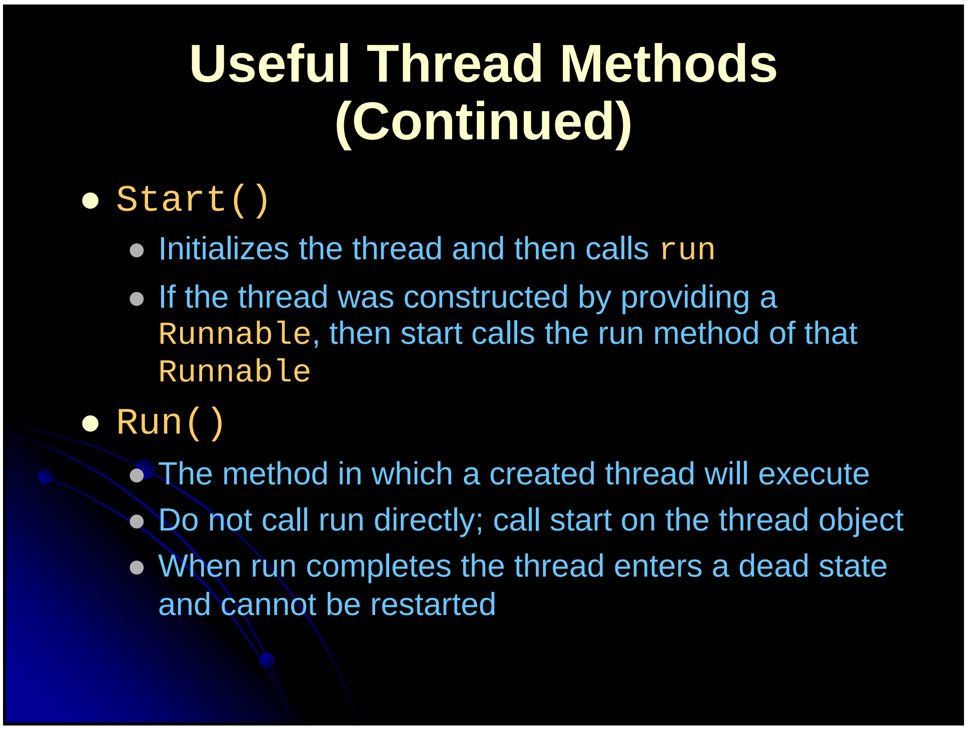
Useful Thread Methods

- `currentThread ()`
  - Returns a reference to the currently executing thread
  - This is a `static` method that can be called by arbitrary methods, not just from within a `Thread` object
    - I.e., anyone can call Thread.currentThread
- `Interrupt ()`
  - One of two outcomes:
    - If the thread is executing `join`, `sleep`, `or wait`, an `InterruptedException` is thrown
    - Sets a flag, from which the interrupted thread can check (isInterrupted)
- `Interrupted ()`
  - Checks whether the currently executing thread has a request for interruption (checks flag) and clears the flag



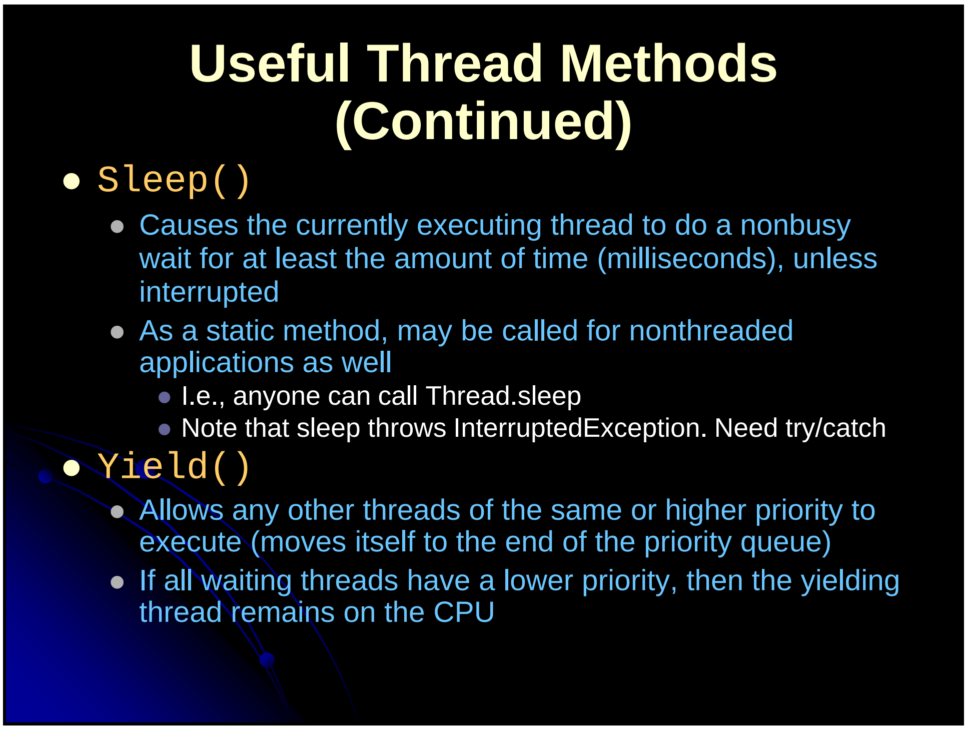
Useful Thread Methods (Continued)

- `isInterrupted()`
  - Simply checks whether the thread’s interrupt flag has been set (does not modify the flag)
    - Reset the flag by calling `interrupted` from within the `run` method of the flagged thread
- `Join()`
  - Joins to another thread by simply waiting (sleeps) until the other thread has completed execution
- `isDaemon()/setDaemon()`
  - Determines or set the thread to be a daemon
  - A Java program will exit when the only active threads remaining are daemon threads



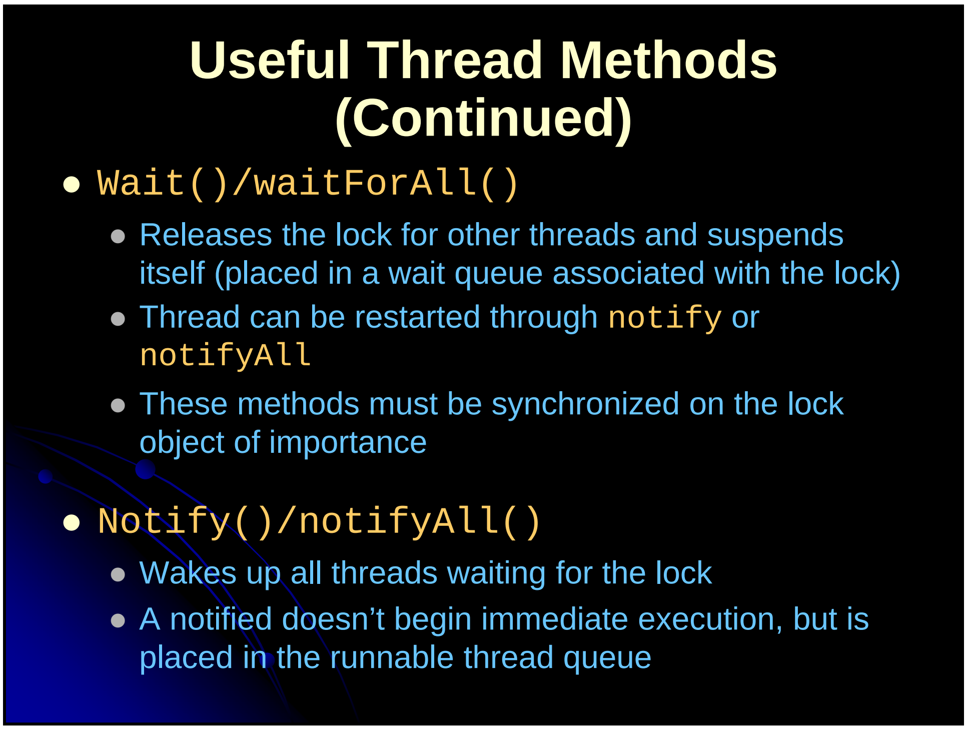
Useful Thread Methods (Continued)

- `Start()`
  - Initializes the thread and then calls `run`
  - If the thread was constructed by providing a `Runnable`, then start calls the run method of that `Runnable`
- `Run()`
  - The method in which a created thread will execute
  - Do not call run directly; call start on the thread object
  - When run completes the thread enters a dead state and cannot be restarted



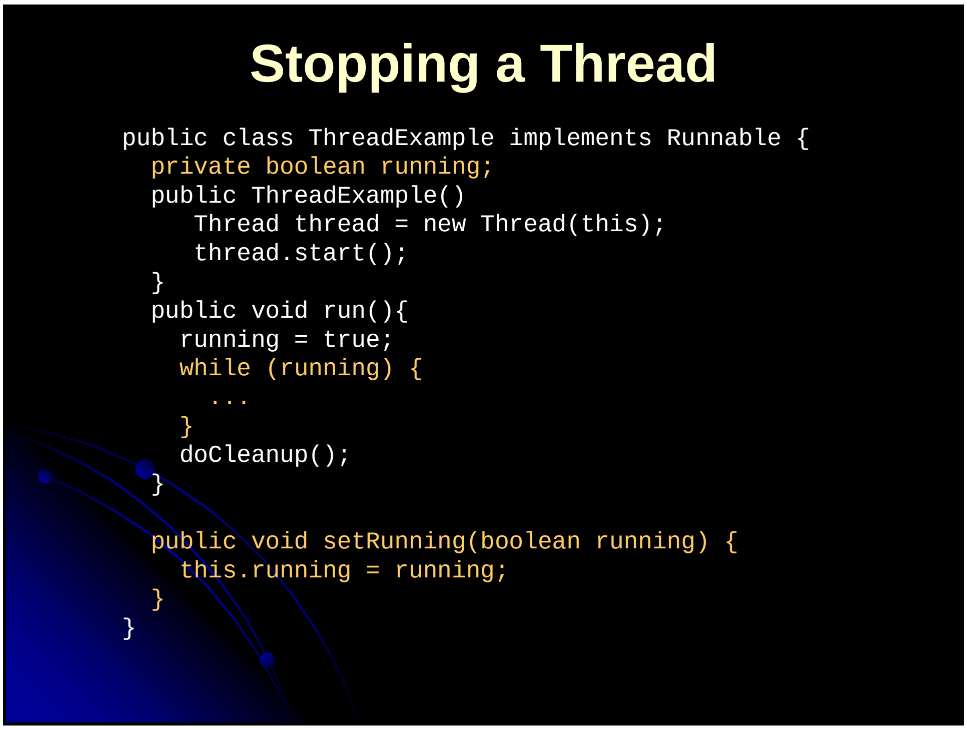
Useful Thread Methods (Continued)

- `Sleep()`
  - Causes the currently executing thread to do a nonbusy wait for at least the amount of time (milliseconds), unless interrupted
  - As a static method, may be called for nonthreaded applications as well
    - I.e., anyone can call Thread.sleep
    - Note that sleep throws InterruptedException. Need try/catch
- `Yield()`
  - Allows any other threads of the same or higher priority to execute (moves itself to the end of the priority queue)
  - If all waiting threads have a lower priority, then the yielding thread remains on the CPU



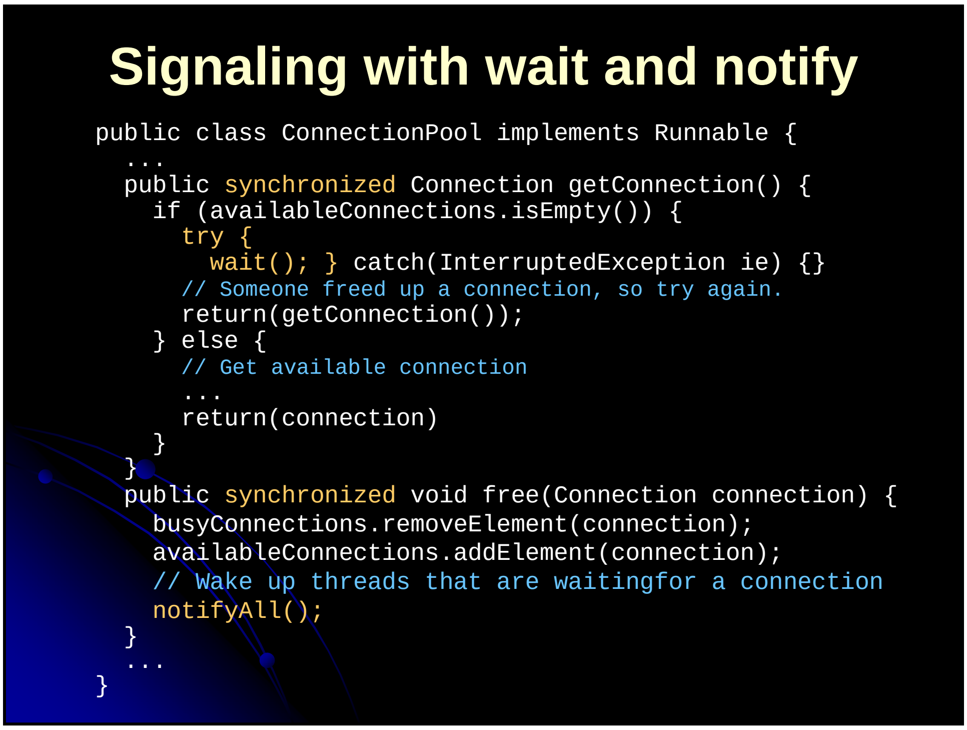
Useful Thread Methods (Continued)

- `Wait()/waitForAll()`
  - Releases the lock for other threads and suspends itself (placed in a wait queue associated with the lock)
  - Thread can be restarted through `notify` or `notifyAll`
  - These methods must be synchronized on the lock object of importance
- `Notify()/notifyAll()`
  - Wakes up all threads waiting for the lock
  - A notified doesn’t begin immediate execution, but is placed in the runnable thread queue



Stopping a Thread

```
public class ThreadExample implements Runnable {
  private boolean running;
  public ThreadExample()
     Thread thread = new Thread(this);
     thread.start();
  }
  public void run(){
    running = true;
    while (running) {
      ...
    }
    doCleanup();
  }

  public void setRunning(boolean running) {
    this.running = running;
  }
}
```



Signaling with wait and notify

```
public class ConnectionPool implements Runnable {
  ...
  public synchronized Connection getConnection() {
    if (availableConnections.isEmpty()) {
      try {
        wait(); } catch(InterruptedException ie) {}
      // Someone freed up a connection, so try again.
      return(getConnection());
    } else {
      // Get available connection
      ...
      return(connection)
    }
  }
  public synchronized void free(Connection connection) {
    busyConnections.removeElement(connection);
    availableConnections.addElement(connection);
    // Wake up threads that are waitingfor a connection
    notifyAll();
  }
  ...
}
```

Summary

- Achieve multithreaded behavior by
  - Inheriting directly from `Thread` (separate class approach)
  - Implementing the `Runnable` interface (callback approach)
- In either case, put your code in the `run` method. Call `start` on the Thread object.
- Avoid race conditions by placing the shared resource in a synchronized block
- You can’t restart a dead thread
- Stop threads by setting a flag that the thread's run method checks

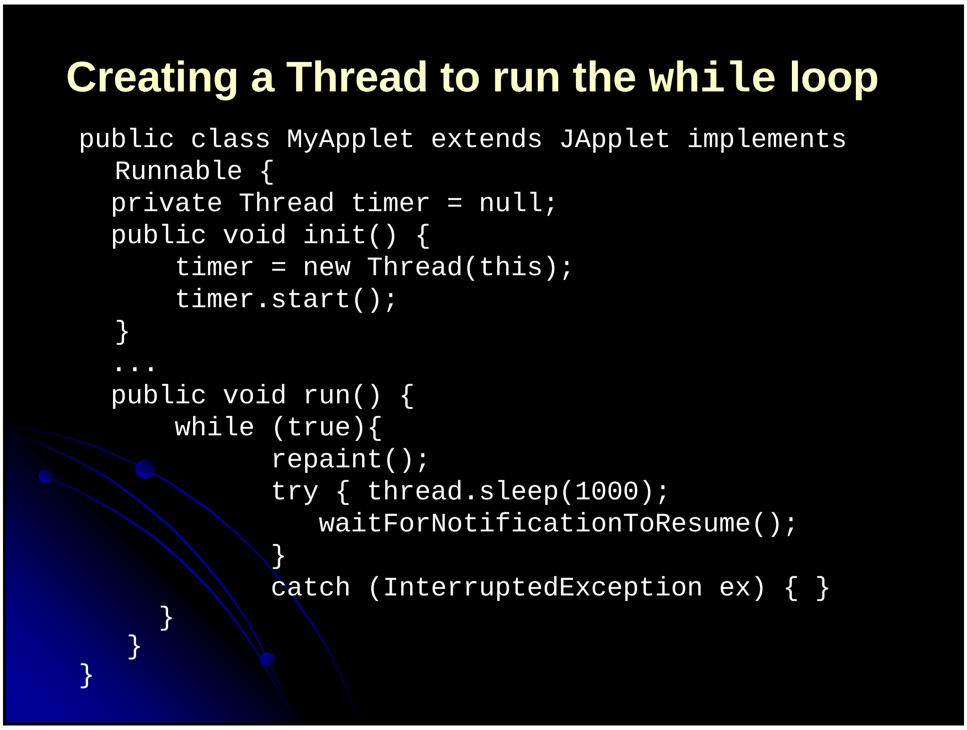
Creating Threads for Applets

In Example "Displaying a Clock" in P11 (Graphics), you drew a clock to show the current time in an applet. The clock does not tick after it is displayed. What can you do to let the clock display a new current time every second? The key to making the clock tick is to repaint it every second with a new current time. You can use the code given below to override the `start()` method in `CurrentTimeApplet`:

What is wrong in this code?

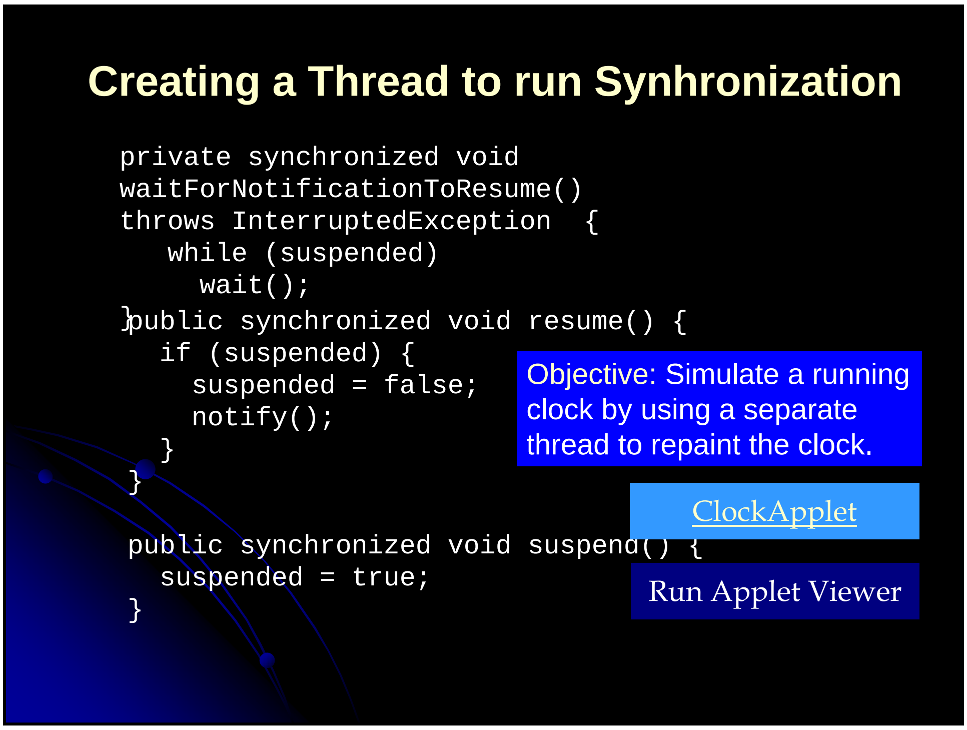
As long as the while loop is running, the browser cannot serve any other event that might be occurring.

```
public void start() {
  while (true) {
    stillClock.repaint();
    try {
      Thread.sleep(1000);
    }
    catch(InterruptedException ex){}
  }
```



Creating a Thread to run the while loop

```
public class MyApplet extends JApplet implements
  Runnable {
  private Thread timer = null;
  public void init() {
      timer = new Thread(this);
      timer.start();
  }
  ...
  public void run() {
      while (true){
            repaint();
            try { thread.sleep(1000);
               waitForNotificationToResume();
            }
            catch (InterruptedException ex) { }
     }
   }
}
```



Creating a Thread to run Synhronization

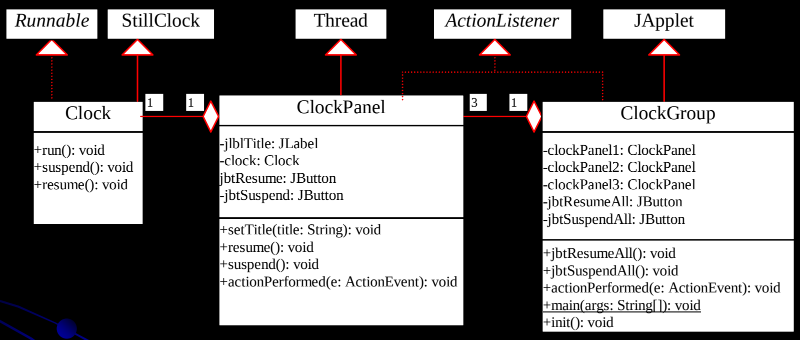
```
  private synchronized void
  waitForNotificationToResume()
  throws InterruptedException  {
     while (suspended)
       wait();
  }public synchronized void resume() {
     if (suspended) {
       suspended = false;
       notify();
     }
   }

   public synchronized void suspend() {
     suspended = true;
   }
```

Objective: Simulate a running clock by using a separate thread to repaint the clock.

ClockApplet

Run Applet Viewer

Controlling a Group of Clocks



Runnable

StillClock

Thread

ActionListener

JApplet

Clock

+run(): void
+suspend(): void
+resume(): void

1

1

ClockPanel

-jlblTitle: JLabel
-clock: Clock
jbtResume: JButton
-jbtSuspend: JButton

+setTitle(title: String): void
+resume(): void
+suspend(): void
+actionPerformed(e: ActionEvent): void

3

1

ClockGroup

-clockPanel1: ClockPanel
-clockPanel2: ClockPanel
-clockPanel3: ClockPanel
-jbtResumeAll: JButton
-jbtSuspendAll: JButton

+jbtResumeAll(): void
+jbtSuspendAll(): void
+actionPerformed(e: ActionEvent): void
+main(args: String[]): void
+init(): void

Clock

ClockGroup

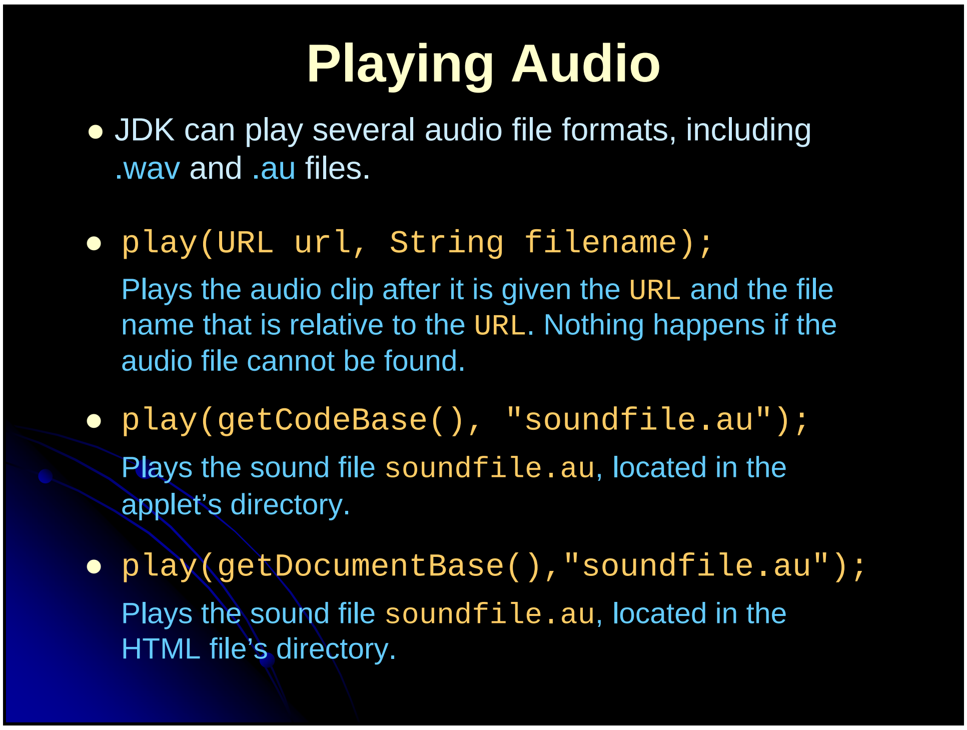
Run

Thank you for your attention!

Multimedia

Agenda

- Audio Files
- Playing Audio
- Running Audio on a Separate Thread
- Displaying Images
- Displaying a Sequence of Images
- Using `MediaTracker`



Playing Audio

- JDK can play several audio file formats, including .wav and .au files.

- `play(URL url, String filename);`

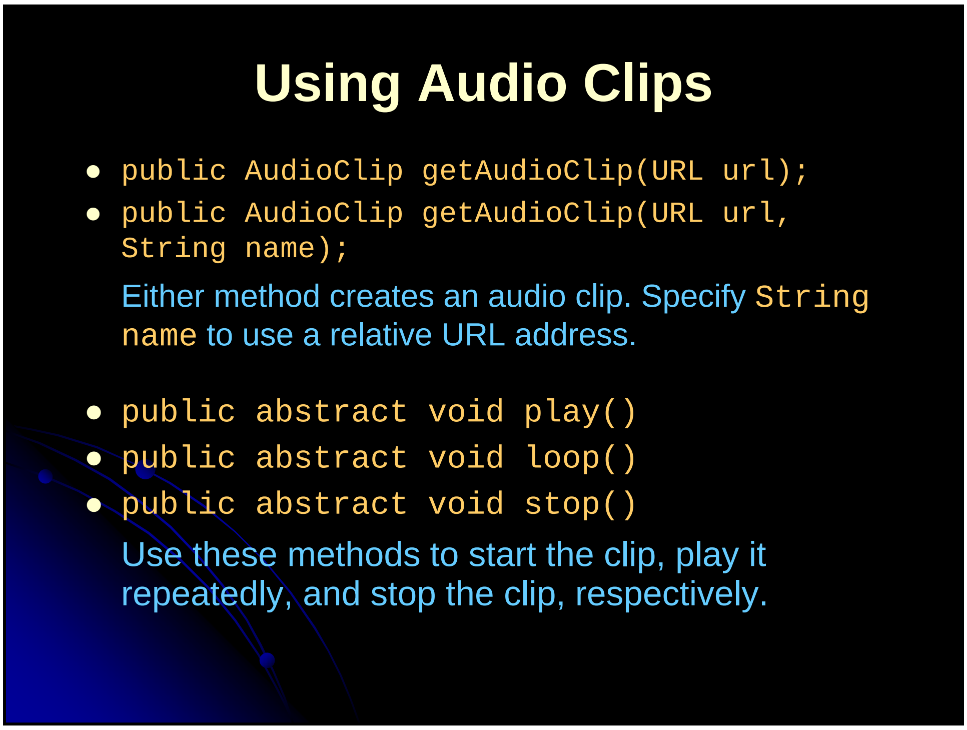
  Plays the audio clip after it is given the URL and the file name that is relative to the URL. Nothing happens if the audio file cannot be found.

- `play(getCodeBase(), "soundfile.au");`

  Plays the sound file `soundfile.au`, located in the applet's directory.

- `play(getDocumentBase(),"soundfile.au");`

  Plays the sound file `soundfile.au`, located in the HTML file's directory.



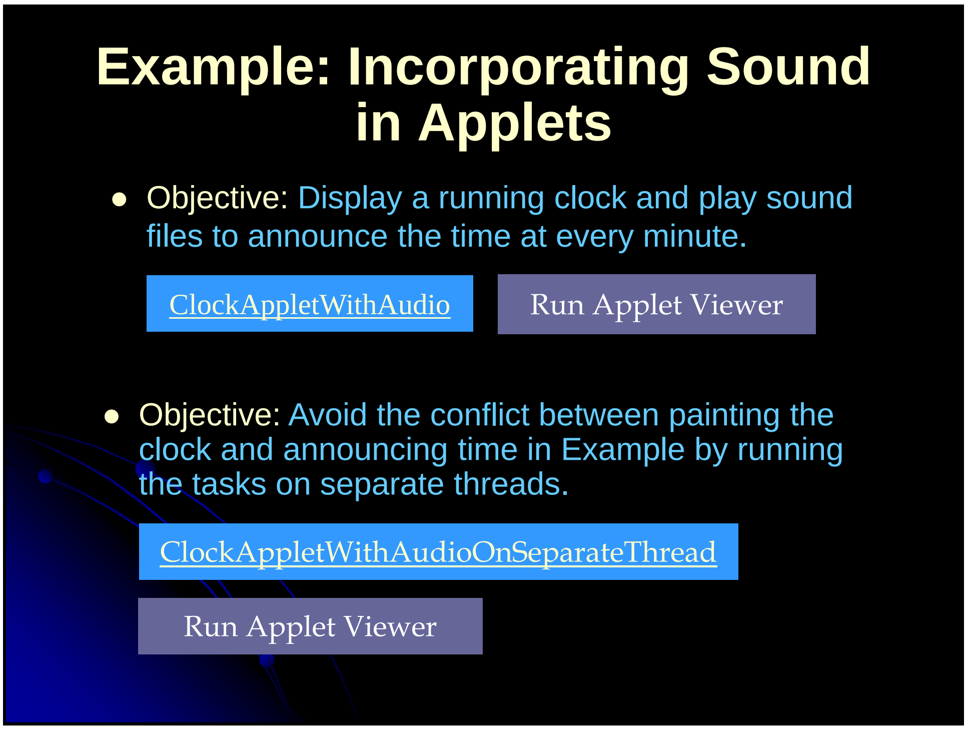
Using Audio Clips

- `public AudioClip getAudioClip(URL url);`
- `public AudioClip getAudioClip(URL url, String name);`

  Either method creates an audio clip. Specify `String name` to use a relative URL address.

- `public abstract void play()`
- `public abstract void loop()`
- `public abstract void stop()`

  Use these methods to start the clip, play it repeatedly, and stop the clip, respectively.



Example: Incorporating Sound in Applets

- Objective: Display a running clock and play sound files to announce the time at every minute.

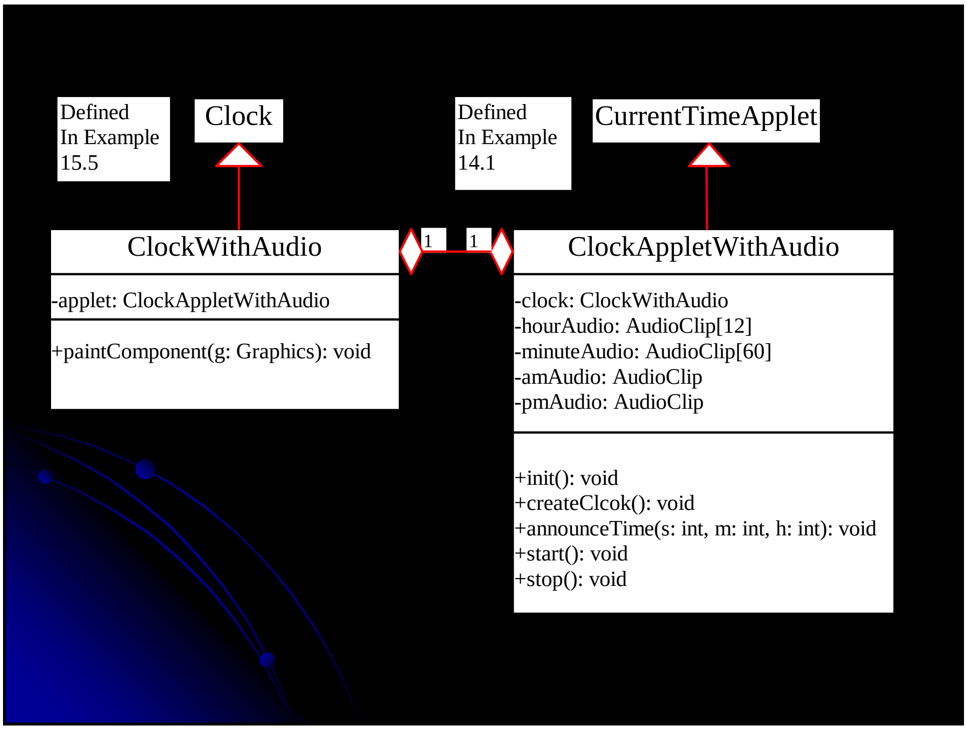
ClockAppletWithAudio

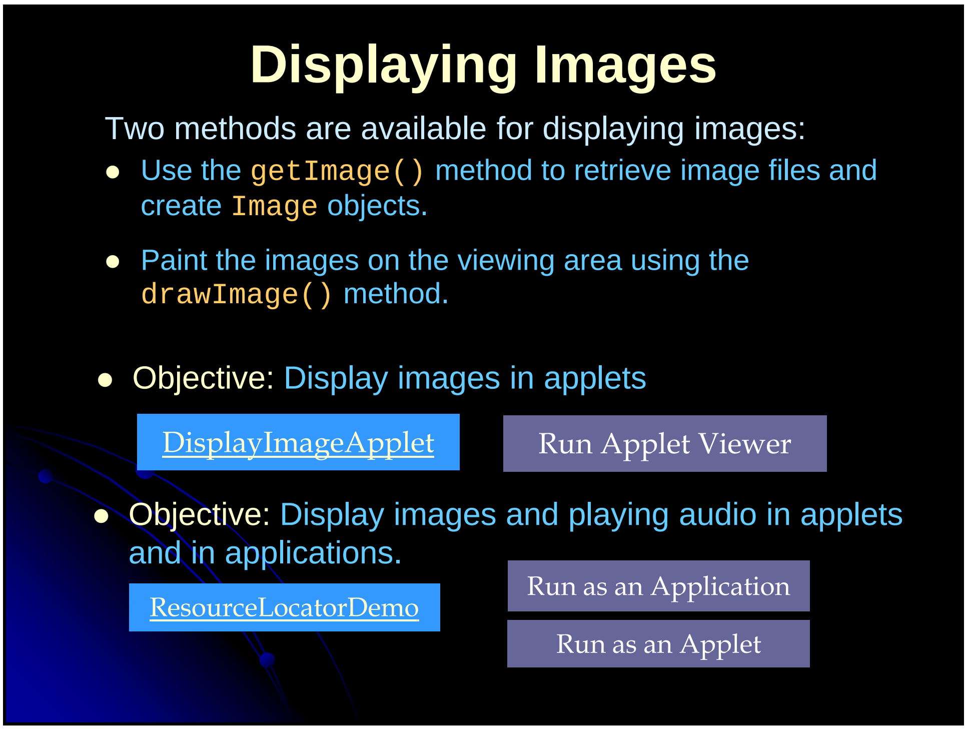
Run Applet Viewer

- Objective: Avoid the conflict between painting the clock and announcing time in Example by running the tasks on separate threads.

ClockAppletWithAudioOnSeparateThread

Run Applet Viewer





Displaying Images

Two methods are available for displaying images:

- Use the `getImage()` method to retrieve image files and create `Image` objects.
- Paint the images on the viewing area using the `drawImage()` method.

- Objective: Display images in applets

DisplayImageApplet

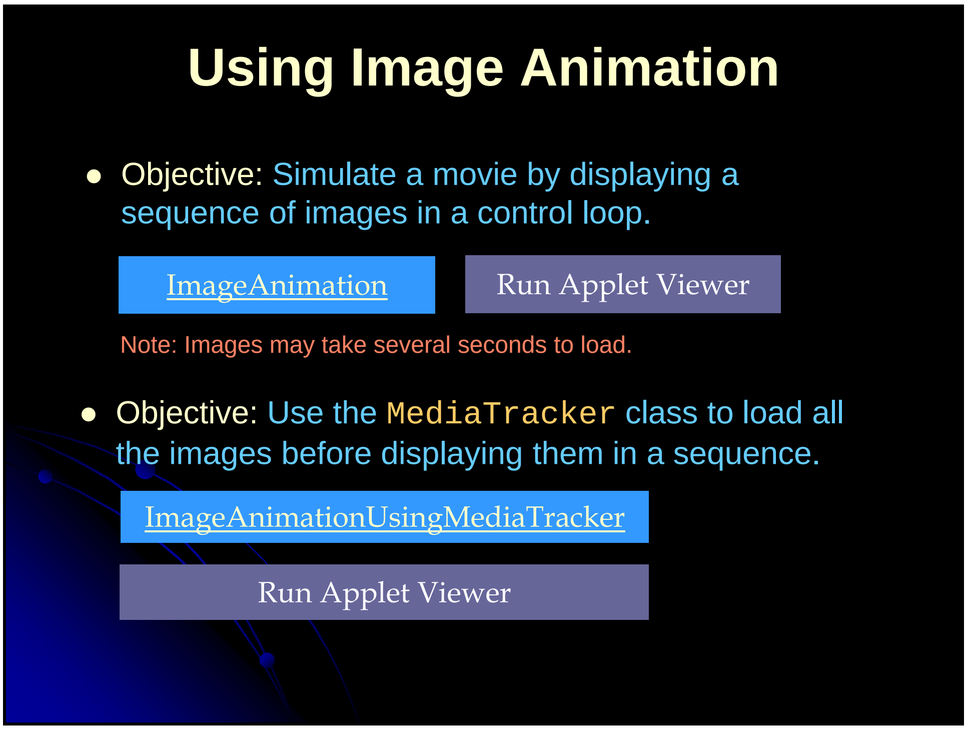
Run Applet Viewer

- Objective: Display images and playing audio in applets and in applications.

ResourceLocatorDemo

Run as an Application

Run as an Applet



Using Image Animation

- Objective: Simulate a movie by displaying a sequence of images in a control loop.

ImageAnimation

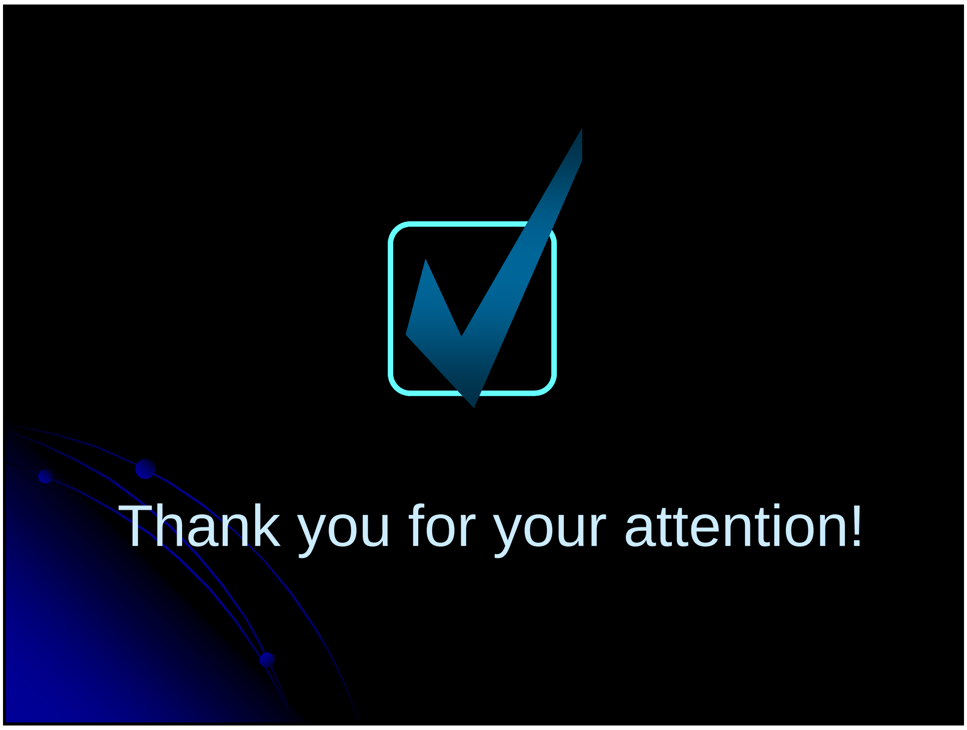
Run Applet Viewer

Note: Images may take several seconds to load.

- Objective: Use the `MediaTracker` class to load all the images before displaying them in a sequence.

ImageAnimationUsingMediaTracker

Run Applet Viewer



Thank you for your attention!