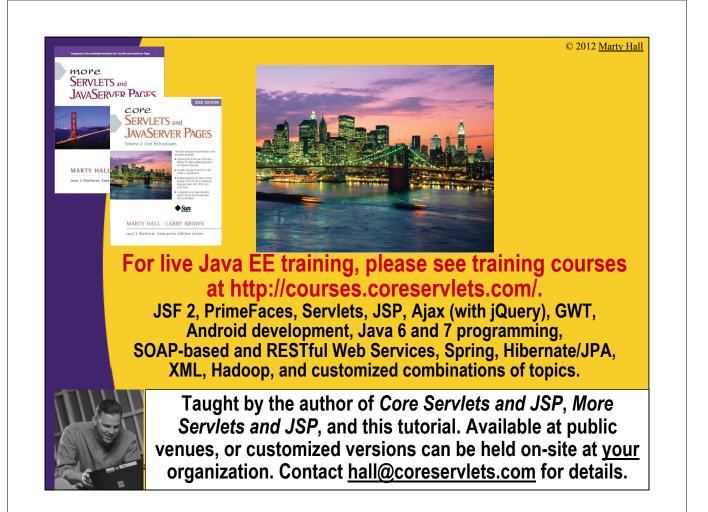


Applets and Basic Graphics

Originals of Slides and Source Code for Examples: http://courses.coreservlets.com/Course-Materials/java.html



Topics in This Section

- 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
- Using try/catch blocks
- The value of @Override
- Controlling image loading



© 2012 Marty Hall

Applet Basics



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

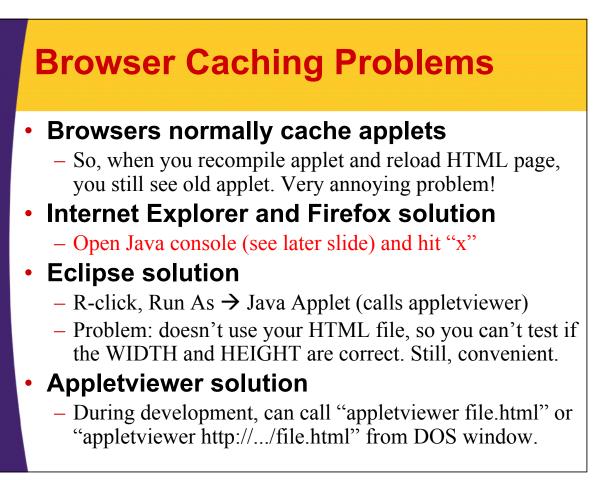
- Ordinary Java applications can invoke locally installed programs (Runtime.exec or ProcessBuilder.start) as well as link to local C/C++ modules ("native" methods)
- These actions are prohibited in applets because there is no way to determine whether the operations these local programs perform are safe

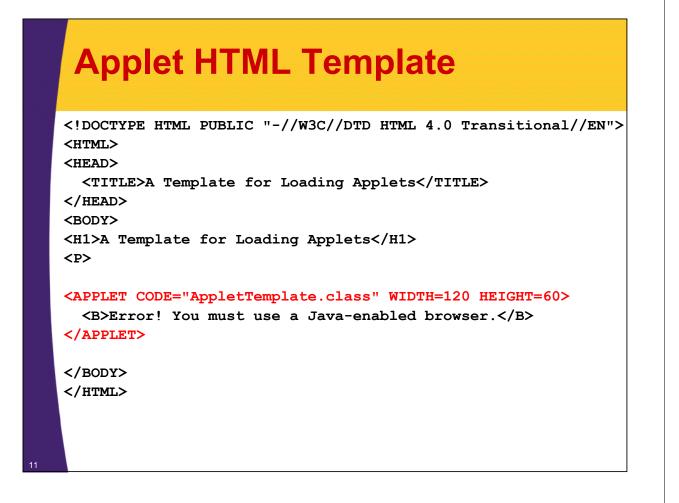
Discover private information about the user

- 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.
    @Override // I will explain @Override later
    public void init() {
        // Variable initializations, image loading, etc.
    }
    @Override // I will explain @Override later
    public void paint(Graphics g) {
        // Drawing operations.
    }
}
```





Applet Example 1: Drawing Diagonal Lines

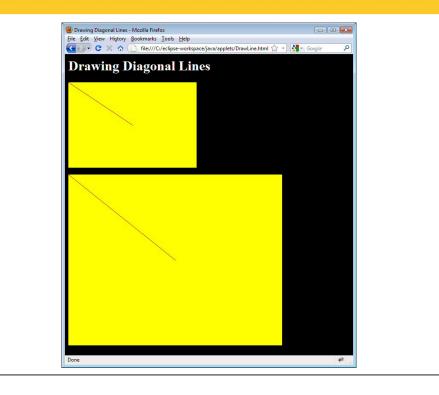
```
import java.applet.Applet;
import java.awt.*;
/** Draws a line from top-left halfway to bottom-right. */
public class DrawLine extends Applet {
 private int width; // Instance var declarations here
 private int height;
  @Override // I'll explain later what this optional tag does
 public void init() {
                                 // Initializations here
    setBackground(Color.YELLOW);
   setForeground(Color.RED);
   width = getWidth();
   height = getHeight();
  }
  Override // I'll explain later what this optional tag does
  public void paint(Graphics g) { // Drawing here
    g.drawLine(0, 0, width/2, height/2);
  }
```

Applet Example 1: HTML

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <HTML> <HEAD> <TITLE>Drawing Diagonal Lines</TITLE> </HEAD> <BODY BGCOLOR="BLACK" TEXT="WHITE"> <H1>Drawing Diagonal Lines</H1> <P> <APPLET CODE="DrawLine.class" WIDTH=300 HEIGHT=200> Sorry, this example requires Java. </APPLET> <P> <APPLET CODE="DrawLine.class" WIDTH=500 HEIGHT=400> Sorry, this example requires Java. </APPLET> </BODY></HTML>

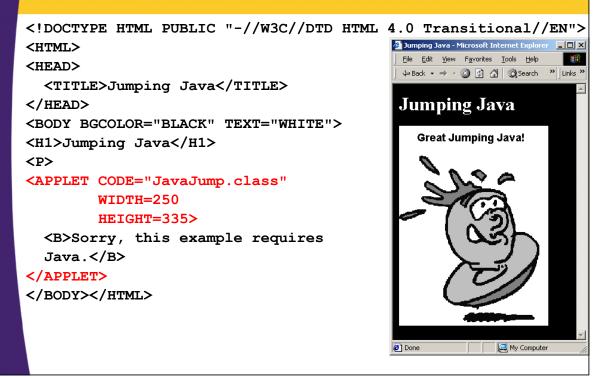
Applet Example 1: Result

13



Applet Example 2 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(getCodeBase(), "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 2: Result



15

Debugging Applets: The Java Console

Standard output (from System.out.println) is sent to the Java Console

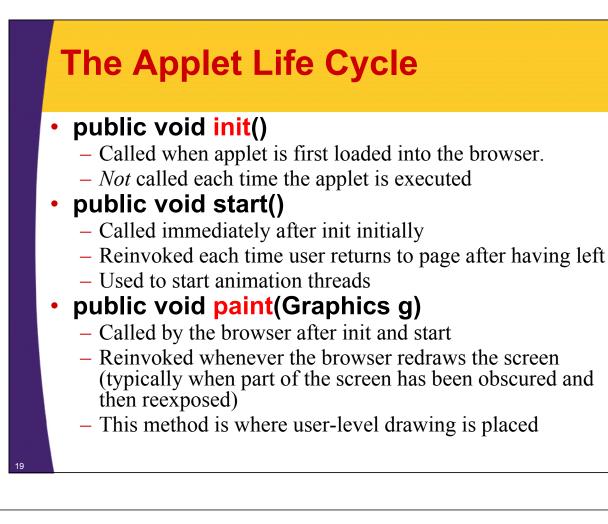
- IE6 and early releases of IE7: Tools \rightarrow Sun Java Console
- Firefox, IE8, later releases of IE7: R-click on Java icon in taskbar, then select Console
 - In either browser, to have it open automatically: go to Control Panel, click on Java, go to "Advanced", choose "Java console", and select "Show Console".
- Note that this is for *your* debugging: don't expect end user to ever notice this

c: clear console f: finalize object	window ts on finalization queue	
g: garbage colle h: display this h l: dump classloa m: print memory o: trigger loggin q: hide console r: reload policy	ect help message der list y usage ng configuration and deployment properti list stack der cache	Hitting "x" will clear cached applets. Very useful during development.
		 java/applets/images/Jumping-Java.gif

© 2012 Marty Hall



Applet Life Cycle and Methods



The Applet Life Cycle (Continued)

public void stop()

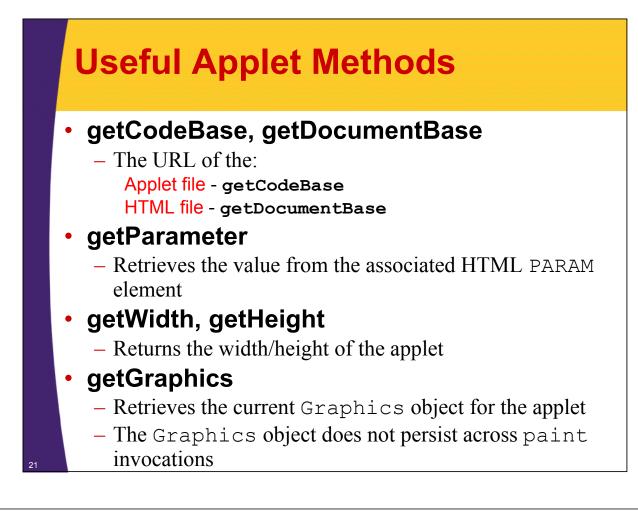
- Called when the user leaves the page
- Used to stop animation threads

public void destroy()

- Called when applet is killed by the browser

Nonstandard behavior in IE

 In some versions of Internet Explorer, unlike in Firefox, 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!).



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
- Java supports MIDI, .aiff, .wav, etc.

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)

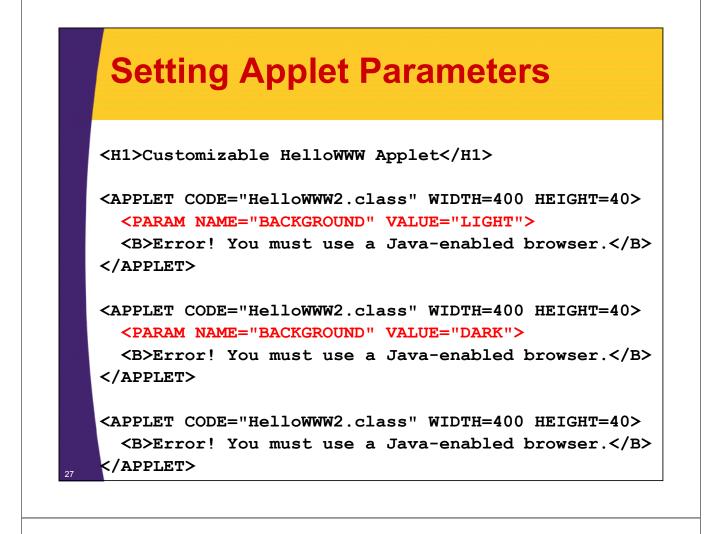
Other Attributes

- 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



© 2012 Marty Hall

Applet Parameters: Letting the HTML Author Supply Data

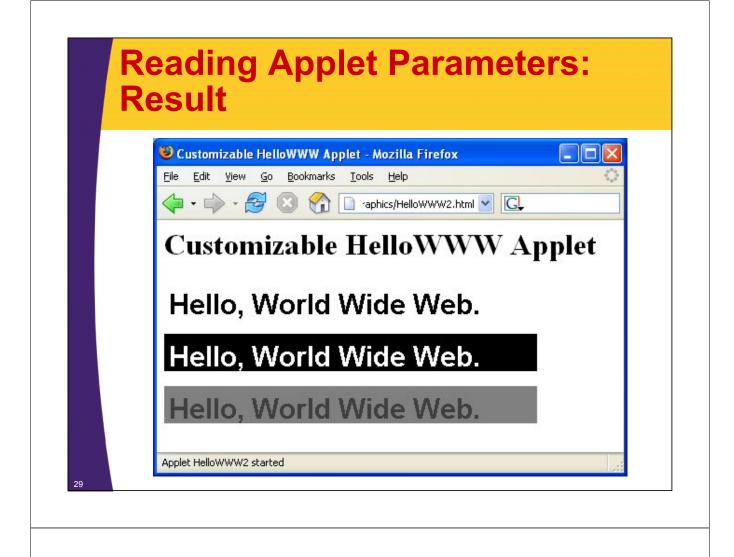




- Use getParameter(name) to retrieve the value of the PARAM element
 - If no match, return value is null. Always check for null!

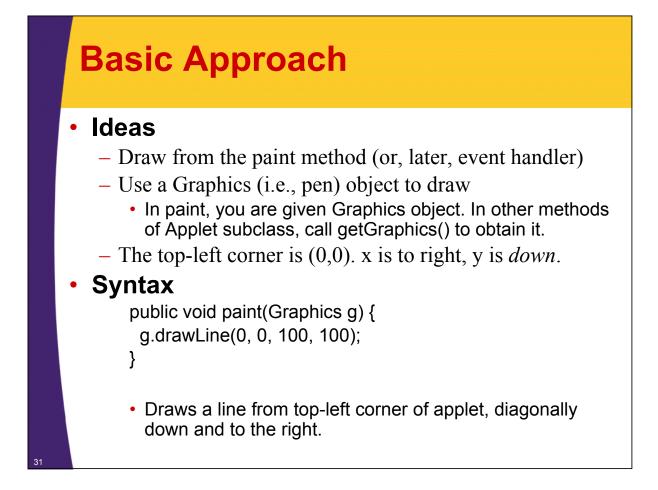
• The name argument is case sensitive

```
public void init() {
  Color background = Color.GRAY;
  Color foreground = Color.DARK_GRAY;
  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;
      foreground = Color.WHITE;
    }
  }
  ...
}
```





Drawing in Applets







- 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

drawlmage

- Draws an image
- Images can be in JPEG or GIF (including animated GIF) 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
- Standard Java font names: Serif (e.g., Times New Roman), SansSerif (e.g., Arial), Monospaced (e.g., Courier New), 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



Loading and Drawing Images

Customized Java EE Training: http://courses.coreservlets.com/ Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

Drawing Images

Register the Image (from init)

```
myImage = getImage(getCodeBase(), "file");
myImage = getImage(url);
```

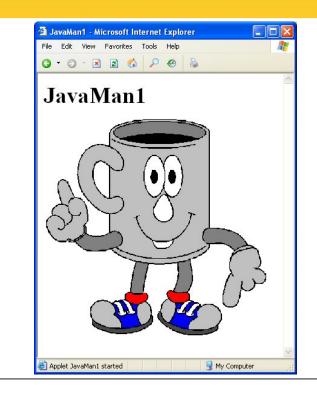
- You should declare the image as an instance variable so that paint can access it later
 - private Image myImage;
- Loading is done in a separate thread
- If URL is absolute, then try/catch block is required
- Draw the image (from paint)

```
g.drawImage(myImage, x, y, window);
g.drawImage(myImage, 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

Image Loading Result



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");
        }
              In real life, it is not useful to load images from absolute URLs in applets. That is because the browser security rules mandate that the applet can only connect 
back to the machine it was loaded from. But, since we are covering image loading anyhow, it is worth mentioning this because you use this approach when 
loading images in regular desktop applications (not browser-based applets). Besides, this gives me an excuse to cover try/catch blocks now.
    }
```

Aside: Exceptions and try/catch Blocks

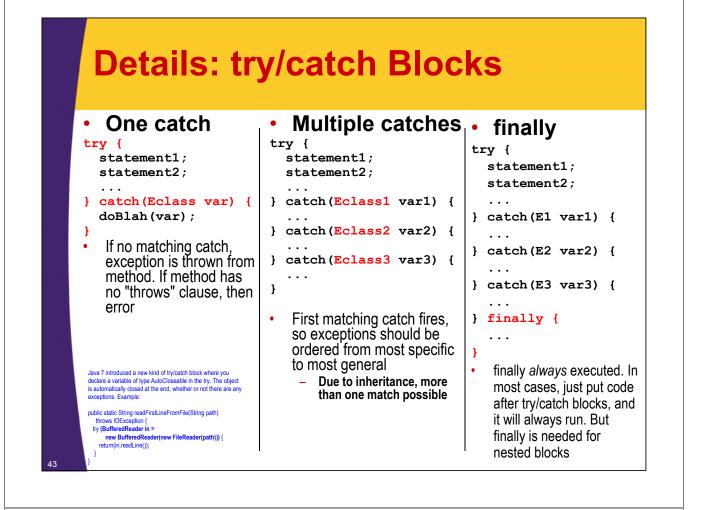
In Java, the error-handling system is based on exceptions

- Exceptions handled in a try/catch block
- When an exception occurs, process flow is immediately transferred to the catch block
- Most exceptions must be handled (unlike C++). If API says method or constructor throws exception, code won't compile unless you use try/catch or say the method throws exception

Basic Form

}

```
try {
   statement1;
   statement2;
   ...
} catch(SomeExceptionClass someVar) {
   handleTheException(someVar);
```

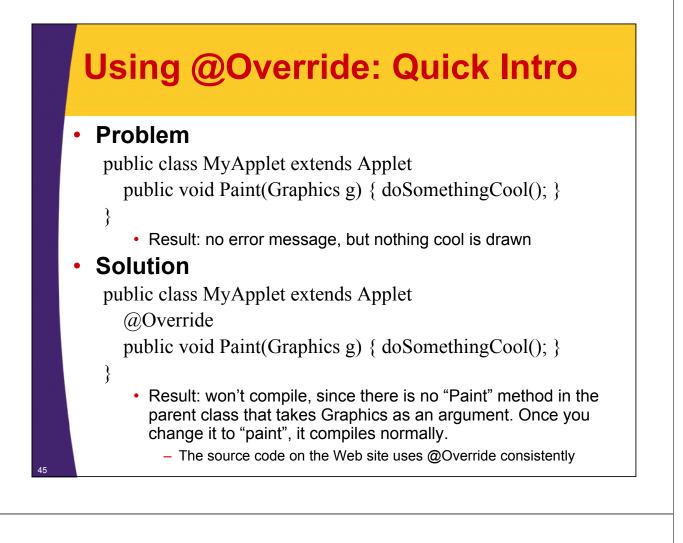




Advanced Topics

Customized Java EE Training: http://courses.coreservlets.com/ Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Hadoop, Android.

Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

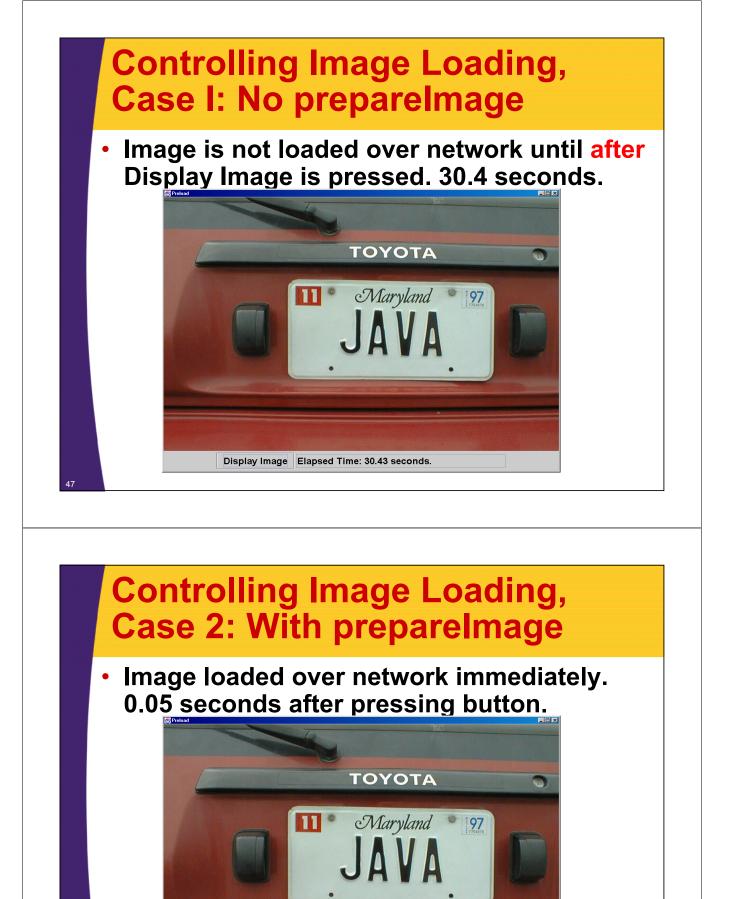


Controlling Image Loading

Use prepareImage to start loading image

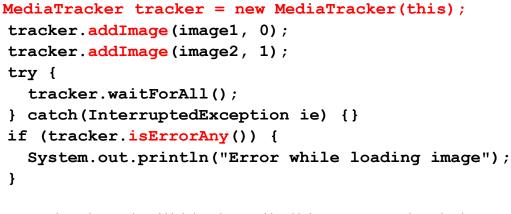
prepareImage(image, window)
prepareImage(image, width, height, window)

- 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

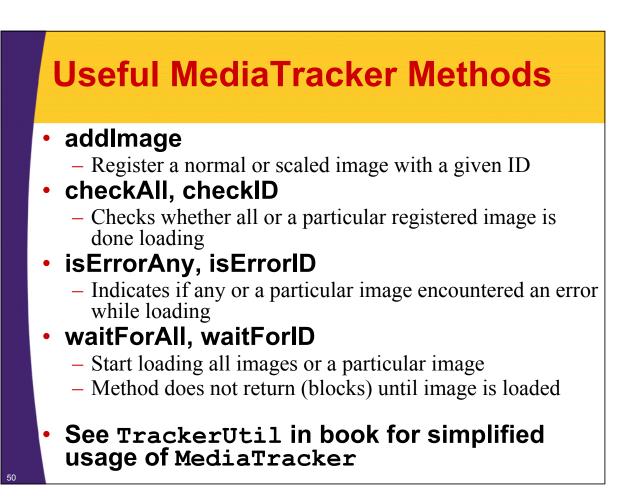


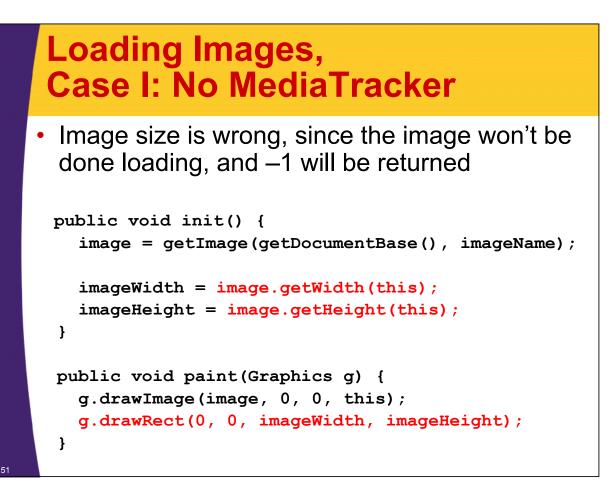
Controlling Image Loading: MediaTracker

Registering images with a MediaTracker to control image loading



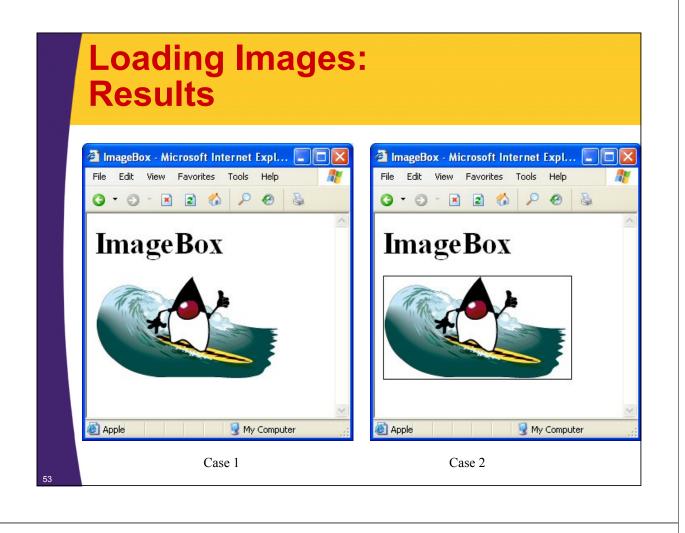
Applet thread will block until all images are loadedEach image is loaded in parallel on a separate thread

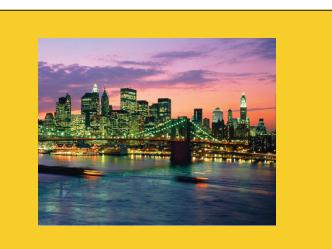




Loading Images, Case 2: With MediaTracker

```
• 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);
```





Wrap-Up

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" image
- g.drawImage(image, x, y, this) to draw image



© 2012 Marty Hall

Questions?

JSF 2, PrimeFaces, Java 7, Ajax, jQuery, Hadoop, RESTful Web Services, Android, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training.