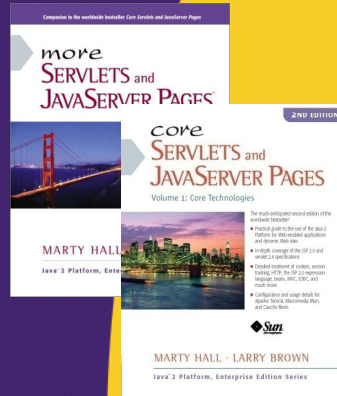




Applets and Basic Graphics

Originals of Slides and Source Code for Examples:
<http://courses.coreservlets.com/Course-Materials/java5.html>

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



**For live Java EE training, please see training courses
at <http://courses.coreservlets.com/>.**

Servlets, JSP, Struts, JSF 1.x, JSF 2.0, Ajax (with jQuery, Dojo, Prototype, Ext-JS, Google Closure, etc.), GWT 2.0 (with GXT), Java 5, Java 6, SOAP-based and RESTful Web Services, Spring, Hibernate/JPA, and customized combinations of topics.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization. Contact hall@coreservlets.com for details.

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

5

© 2010 Marty Hall



Applet Basics

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

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

6

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

7

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

8

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.
    }
}
```

9

Browser Caching Problems

- **Browsers normally cache applets**
 - So, when you recompile applet and reload HTML page, you still see old applet. Very annoying problem!
- **Internet Explorer and Firefox solution**
 - Open Java console (see later slide) and hit “x”
- **Eclipse solution**
 - R-click, Run As → Java Applet (calls appletviewer)
 - Problem: doesn’t use your HTML file, so you can’t test if the WIDTH and HEIGHT are correct. Still, convenient.
- **Appletviewer solution**
 - During development, can call “appletviewer file.html” or “appletviewer http://.../file.html” from DOS window.

10

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>
```

11

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

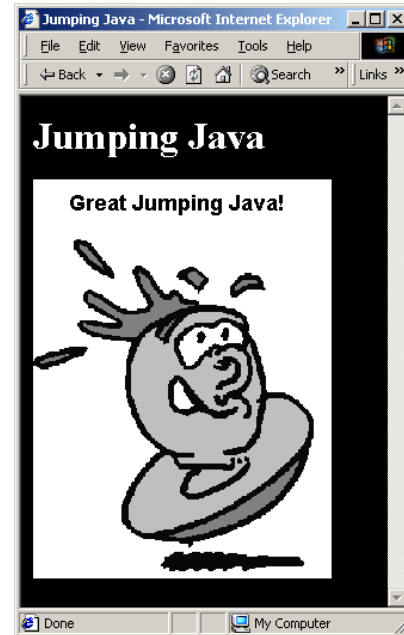
  public void init() { // Initializations here
    setBackground(Color.WHITE);
    setFont(new Font("SansSerif", Font.BOLD, 18));
    jumpingJava = getImage(getCodeBase(),
                          "images/Jumping-Java.gif");
    add(new Label("Great Jumping Java!"));
    System.out.println("Yow! I'm jiving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

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

12

Applet Example: Result

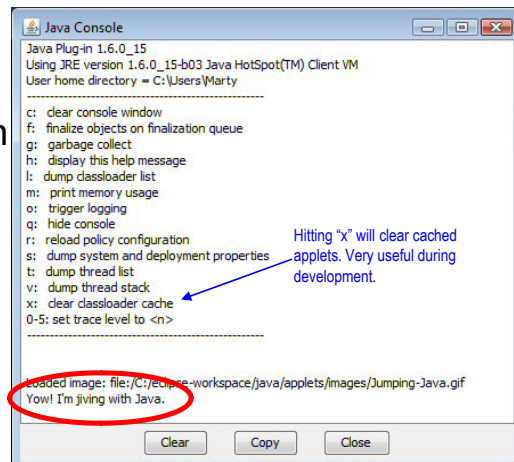
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Jumping Java</TITLE>
</HEAD>
<BODY BGCOLOR="BLACK" TEXT="WHITE">
<H1>Jumping Java</H1>
<P>
<APPLET CODE="JavaJump.class"
  WIDTH=250
  HEIGHT=335>
  <B>Sorry, this example requires
  Java.</B>
</APPLET>
</BODY>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



13

Debugging Applets: The Java Console

- **Standard output (from System.out.println) is sent to the Java Console**
 - IE6 and early releases of IE7: Tools → 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



14



Applet Life Cycle and Methods

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

15

The Applet Life Cycle

- **public void `init()`**
 - Called when applet is first loaded into the browser.
 - *Not* called each time the applet is executed
- **public void `start()`**
 - Called immediately after `init` initially
 - Reinvoked each time user returns to page after having left
 - Used to start animation threads
- **public void `paint(Graphics g)`**
 - Called by the browser after `init` and `start`
 - Reinvoked whenever the browser redraws the screen (typically when part of the screen has been obscured and then reexposed)
 - This method is where user-level drawing is placed

16

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!).

17

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
- **getWidth, getHeight**
 - Returns the width/height of the applet
- **getGraphics**
 - Retrieves the current `Graphics` object for the applet
 - The `Graphics` object does not persist across paint invocations

18

Useful Applet Methods (Continued)

- **showDocument (AppletContext method)**
 - `getAppletContext().showDocument(...)`
 - Asks the browser to retrieve and 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`

19

Useful Applet Methods (Continued)

- **getAudioClip, play**
 - Retrieves an audio file from a remote location and plays it
 - Java supports MIDI, .aiff, .wav, etc.
- **setBackground, 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)

20

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.

21

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

22



Applet Parameters: Letting the HTML Author Supply Data

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

23

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>
```

24

Reading Applet Parameters

- 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;  
        }  
    }  
    ...  
}
```

25

Reading Applet Parameters: Result



26



Drawing in Applets

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

27

Basic Approach

- **Ideas**
 - Draw from the paint method (or, later, event handler)
 - Use a Graphics (i.e., pen) object to draw
 - In paint, you are given Graphics object. In other methods of Applet subclass, call `getGraphics()` to obtain it.
 - The top-left corner is (0,0). x is to right, y is *down*.
- **Syntax**

```
public void paint(Graphics g) {  
    g.drawLine(0, 0, 100, 100);  
}
```

 - Draws a line from top-left corner of applet, diagonally down and to the right.

28

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)

29

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 animated GIF) format

30

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.

31

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`

32

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

33

© 2010 Marty Hall



Loading and Drawing Images

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

34

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

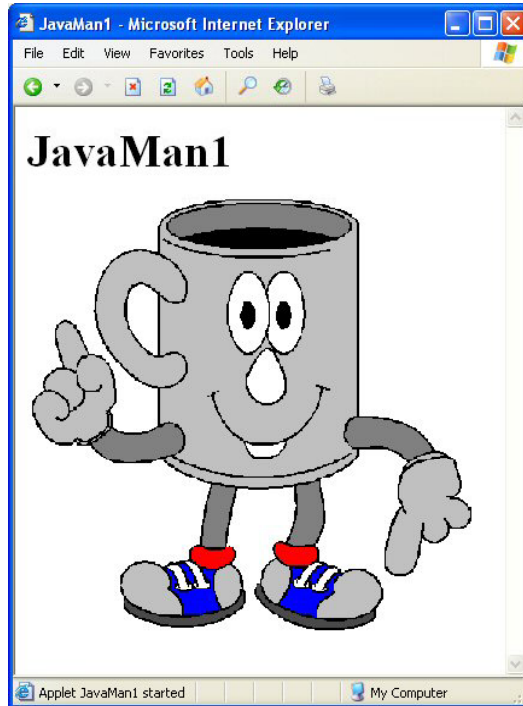
35

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);  
    }  
}
```

36

Image Loading Result



37

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.

38

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) ;
}
```

39

Details: try/catch Blocks

- **One catch**

```
try {
    statement1;
    statement2;
    ...
} catch (Eclass var) {
    doBlah (var) ;
}
```

- If no matching catch, exception is thrown from method. If method has no "throws" clause, then error

- **Multiple catches**

```
try {
    statement1;
    statement2;
    ...
} catch (Eclass1 var1) {
    ...
} catch (Eclass2 var2) {
    ...
} catch (Eclass3 var3) {
    ...
}
```

- First matching catch fires, so exceptions should be ordered from most specific to most general
 - Due to inheritance, more than one match possible

- **finally**

```
try {
    statement1;
    statement2;
    ...
} catch (E1 var1) {
    ...
} catch (E2 var2) {
    ...
} catch (E3 var3) {
    ...
} finally {
    ...
}
```

- `finally` always executed

40



Advanced Topics

Customized Java EE Training: <http://courses.coreservlets.com/>
 Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
 Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

41

Using @Override

- **Problem**

```
public class MyApplet extends Applet
    public void Paint(Graphics g) { doSomethingCool(); }
}
```

- Result: no error message, but nothing cool is drawn

- **Solution**

```
public class MyApplet extends Applet
    @Override
    public void Paint(Graphics g) { doSomethingCool(); }
}
```

- Result: won't compile, since there is no "Paint" method in the parent class that takes Graphics as an argument. Once you change it to "paint", it compiles normally.

- The source code on the Web site uses @Override consistently

42

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

43

Controlling Image Loading, Case I: No `prepareImage`

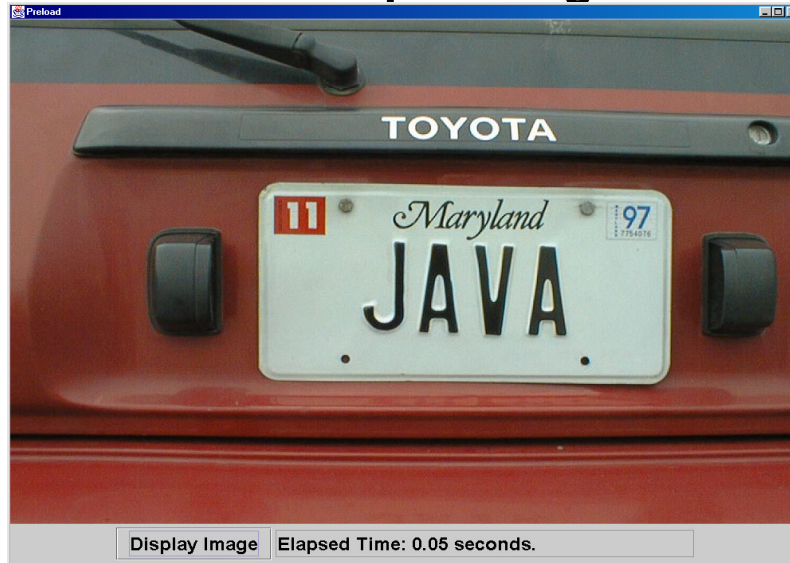
- Image is not loaded over network until **after** `Display Image` is pressed. 30.4 seconds.



44

Controlling Image Loading, Case 2: With prepareImage

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



45

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

46

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**

47

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);
}
```

48

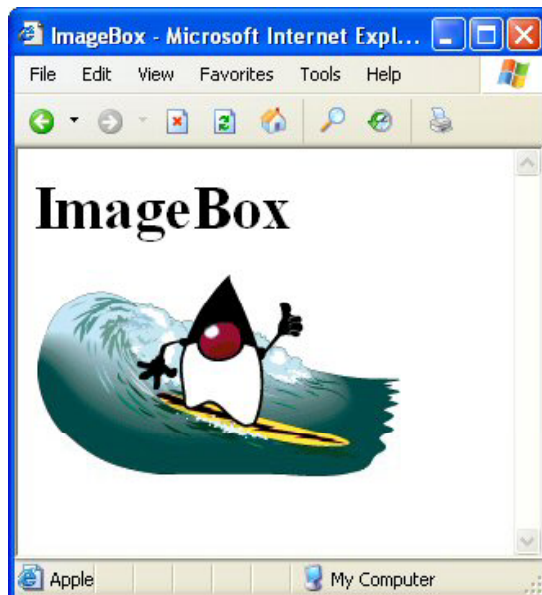
Loading Images, Case 2: With MediaTracker

- Image is loaded before determining size

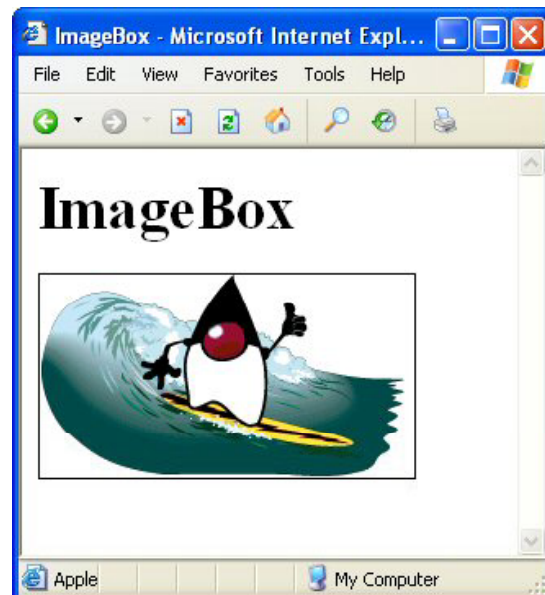
```
public void init() {  
    image = getImage(getDocumentBase(), imageName);  
    MediaTracker tracker = new MediaTracker(this);  
    tracker.addImage(image, 0);  
    try {  
        tracker.waitForAll();  
    } catch (InterruptedException ie) {}  
    ...  
    imageWidth = image.getWidth(this);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

49

Loading Images: Results



Case 1



Case 2

50



Wrap-Up

Customized Java EE Training: <http://courses.coreservlets.com/>
Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.

51

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

52



Questions?

Customized Java EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, JSF 2.0, Struts, Ajax, GWT 2.0, Spring, Hibernate, SOAP & RESTful Web Services, Java 6.

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