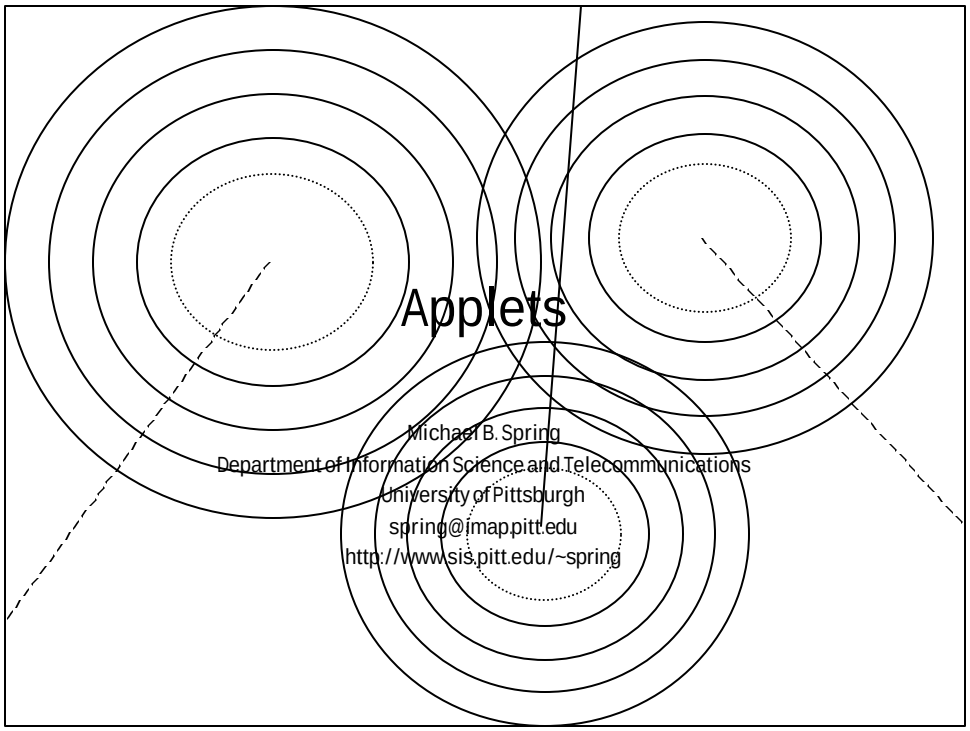

# Applets

Michael B. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
spring@imap.pitt.edu
http://www.sis.pitt.edu/~spring

## A Definition

- An applet is Java program that may be downloaded over a network connection and run on a local machine via a Java-enabled browser.
- An applet is security restricted in several ways and most applets are written to the old Java interface standards for reasons of browser compatibility.
- Applets provide a means of doing client side data manipulation under the http protocol

# Overview

- Advantages of Applets
  - Swing vs. AWT Applets
- File components
- Viewing applets
- HTML tags
- Life cycle methods
- A simple applet
  - Passing parameters
  - Additional methods
- JAR files
  - Applet security
- Common errors



# Advantages of Applets

- Can be run from anywhere (using a Java-enabled browser)
- No installation
- No redistribution or reinstallation between upgrades
- Ensures a consistent version amongst users
- Utilizes the processing power of the client machine
- Enables dynamic web content (far beyond the capabilities of scripting languages)
- Built-in security

## Swing vs. AWT Applets

AWT

- Extend the class Applet
- Because the browser world has not yet caught up with Swing and Java 2, this course will focus on AWT applets

Swing

- Extend the class JApplet
- Requires a Java 1.2 enabled browser or a browser with the Java Plug-In



## File Components

An applet will generally consist of:

- One or more class files
- An associated HTML file

In addition:

- An applet may also include resource files such as image (e.g. .gif, .jpg), sound (e.g. .wav, .au), or data (flat text or DB files)
- The class file(s) and resource file(s) may also be compressed into a single file called a JAR

# Viewing Applets

- Applets are normally invoked from an HTML page
  - When an HTML page in a Java-enabled web browser has an <APPLET> tag, the applet is invoked when the page is opened
  - When an applet is viewed in a web browser, it is invoked by the browser's JVM rather than the JVM installed on your machine by the JDK.
- Applets may also be viewed by using the command line utility "appletviewer" located in the bin directory of your JDK installation
  - The argument to "appletviewer" is the associated HTML file rather than the name of a class
  - When testing and debugging applets, you should use appletviewer rather than a browser



# HTML Tags

- An example of the minimal HTML required to embed an applet within an HTML page:

```
<APPLET CODE="SomeApplet.class"
 WIDTH=400 HEIGHT=300>
</APPLET>
```

# HTML Tags

Attributes of the <APPLET> tag:

- CODE – the name of the class file to be invoked
- CODEBASE – the directory where the class or JAR files are located.
- ARCHIVE – the name of the JAR file(s), if any
- NAME – a string used to refer to the applet
- WIDTH – the width of the applet in pixels
- HEIGHT – the height of the applet in pixels
- ALIGN – specifies the alignment of the applet within the HTML page



# Life Cycle Methods

- Applets do not invoke a main() method as Java applications do
- Instead, applets inherit four special methods associated with an applet's life cycle:
  - init()
  - start()
  - stop()
  - destroy()
- These methods are invoked by the browser and are not typically called explicitly

# Life Cycle Methods

- init() – invoked when the applet is first loaded into memory
  - Invoked once
- start() – invoked each time the browser enters the page containing the applet
  - Invoked after init()
  - May be invoked repeatedly
- stop() – invoked each time the browser leaves the page containing the applet
  - May be invoked repeatedly
- destroy() – invoked when the browser exits



# Life Cycle Methods

- In general, most of the code which you would normally put in the main() method of a Java application will be put in the init() method
- Typically, you will not implement a constructor for an applet. Initialization code should be placed inside init()
- Typically, code dealing with the starting and halting of threads, animations, or sound will go in the start() and stop() methods
- Code which reclaims any non-memory resources may go in the destroy() method
- Many Java applets only override the init() method

# A Simple Applet

```
// HelloWWW.java

import java.applet.*;
import java.awt.*;

public class HelloWWW extends Applet
{
    public void init()
    {
      Label label = new Label("Hello World Wide Web!");
      add(label);
    }
}
```



# A Simple Applet

```
<!– HelloWWW.html →
<HTML>
<HEAD>
<TITLE>Hello WWW</TITLE>
</HEAD>
<BODY>
<CENTER>
<APPLET CODE="HelloWWW.class" WIDTH=400 HEIGHT=100>
</APPLET>
</BODY>
</HTML>
```

# Passing Parameters

- An applet can be passed parameters via parameter tags in the HTML page in which the applet is embedded
- Parameters tags can be used by webmasters with no Java knowledge to customize a Java applet
- Parameters specified in the HTML code can be retrieved by the applet using the method getParameter()
- All parameters are passed as Strings



# Passing Parameters

- HTML tags example:

```
<APPLET CODE="someApplet.class"
 WIDTH=400 HEIGHT=300>
<PARAM NAME="font_name" VALUE="Arial">
<PARAM NAME="font_size" VALUE="12">
</APPLET>
```

- Applet code example:

```
String fontName = getParameter("font_name");
int fontSize = Integer.parseInt(getParameter("font_size"));
```

# getParameterInfo()

- The Applet class provides a method called getParameterInfo(), which returns a two-dimensional array of strings containing information about the parameters that an applet accepts
- The return value should contain an array of three strings for each parameter
- The three strings correspond to the name, type, and a brief description for each parameter
- e.g.

```
public String[][] getParameterInfo() {
        String[][] pi = {
                {"font_name", "String", "font family name"},
                {"font_size", "int", ", "size of the font in points"}
        };
        return pi;
}
```



# Additional Methods

- AppletContext getAppletContext() – returns an object corresponding to the applet's environment
- String getAppletInfo() – returns a string containing information about the applet (author, version, etc.)
- void showStatus(String msg) – displays a string in the status bar of the browser

# Additional Methods

- URL getCodeBase() – returns the base URL of the applet's class file
- URL getDocumentBase() – returns the base URL of the applet's HTML file
- Image getImage(URL url, String name) – returns an image from a specified URL
- AudioClip getAudiioClip(URL url, String name) – returns an audio clip from a specified URL



# Additional methods

- The AppletContext object returned from getAppletContext() can be used to load a new URL in the browser from which the applet was opened
- The showStatus() method can be used to deliver messages that would normally be directed to System.out, to the browser's status bar
- The code in the following slide demonstrates the use of getAppletContext(), showStatus(), and get DocumentBase(), in an applet which loads a new page at the press of a button

# Loading a new URL

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
public class LoadPage extends Applet implements ActionListener {
    public void init() {
        Button button = new Button("Press me");
        button.addActionListener (this);
        add(button);
    }
    public void actionPerformed (ActionEvent evt) {
        try {
            URL url = new URL(getDocumentBase (), "page2.html");
            getAppletContext().showDocument(url);
        } catch (Exception e) {
            showStatus ("Error: " + e);
        }
    }
}
```



# JAR Files

- The overhead involved in making repeated HTTP connections is often greater than the transfer time itself
- An efficient way to deliver applets consisting of multiple files is to create a JAR
- A JAR file is an archive similar to a zip file
- A JAR file may contain multiple class files as well as other resource files associated with an applet
- The location of a JAR file is specified by the ARCHIVE attribute of the <APPLET> tag
- JAR files are created using the command line utility “jar” located in the bin directory of your JDK installation

# JAR files

- Creating a JAR file
  - Command line example:

```
jar –cf SomeApplet.jar SomeApplet.class HelperClass.class picture.gif
```

- Referencing a JAR file
  - HTML tag example:

```
<APPLET CODE="SomeApplet.class"
 ARCHIVE="SomeApplet.jar"
 WIDTH=400 HEIGHT=300>
</APPLET>
```



# Applet Security

- Applet security is provided by the browser in which an applet is loaded
- The security policy of a browser is configurable
- A browser may impose lesser security restrictions on an applet which has been signed with a digital signature
- A signed applet verifies the origin of the applet, but places no guarantees on its behavior

# Applet Security

The three key components of the Java security model:

- Security Manager – enforces the security policy of a given JVM (in web browsers, responsible for the applet-specific restrictions described previously)
- Class loader – loads classes into memory and interacts with the security manager to select policies based upon the origin of each class
- Bytecode verifier – ensures that bytecode conforms to the format of the Java virtual machine specification



# Applet Security

- By default, an applet loaded over the network is considered untrusted
- Untrusted applets cannot:
  - Read, write, list, rename, or delete files on the local machine
  - Spawn a new process
  - Make a network connection to a host other than the one it came from
  - Access the clipboard
  - Start a print job
  - Define native methods

# Applet Security

- An untrusted applet can access the following system properties by using the method System.getProperty():
  - java.version
  - java.vendor
  - java.vendor.url
  - java.class.version
  - os.name
  - os.version
  - os.arch
  - file.separator
  - path.separator
  - line.separator



# Applet Security

- An untrusted applet cannot access the following system properties:
  - java.home
  - java.class.path
  - user.name
  - user.home
  - user.dir

## Common Errors

Common reasons for an applet not working:

- You didn't upload class files for all needed classes (especially class files for nested classes)
- You FTP'd your class files to the server in ASCII mode rather than binary
- Your class files are not world readable (in Unix, use "chmod a+r *.class")
- You forgot a closing </APPLET> tag



## Exercise

- Write an applet which computes the shipping and handling for an order, based on the criterion in the following two slides.

# Exercise

- Create a text field for the initial amount
- Create a non-editable text field for the final amount
- Use radio buttons for the user to select US or Canada
- Use check boxes for first and second day air and COD options
- Make next day and second day air mutually exclusive
- Clear the initial amount field and put an error message in the status bar when the user enters an invalid amount (characters, negative number, etc.)
- Create a parameter which allows a webmaster to configure the font color or some other aspect of the applet



# Exercise

| | US | Canada |
|---|---|---|
| Under $50 | $5 | $9 |
| $50 - $99.99 | $8 | $15 |
| $100 - $299.99 | $12 | $23 |
| $300 and over | Free | Free |

- Next day air: add $20
- Second day air: add $9
- C.O.D.: add $5