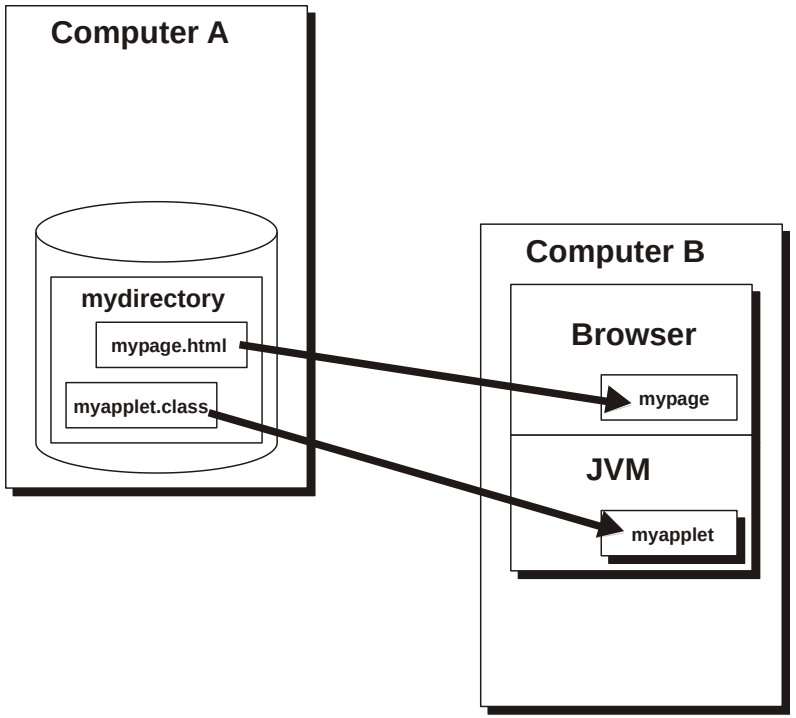
Network Computing Lecture 2

Professor Louise E. Moser

Spring 2012

Applets

- An applet is a self-contained Java program that can be downloaded from a server over the Internet and run on your local computer, inside a Web browser or a special Java program, the *appletviewer*, whose purpose is to load applets
- Like any other object, an applet contains code and data



The Applet and JApplet Classes

- Every applet is declared as an extension of the *Applet* class
- For Swing, instead of the *Applet* class, use the *JApplet* class
- The inheritance hierarchy for *JApplet* is

java.lang.Object
java.awt.Component
java.awt.Container
java.awt.Panel
java.applet.Applet
javax.swing.JApplet

with the implemented interfaces

javax.accessibility.Accessible
java.awt.image.ImageObserver
java.awt.MenuContainer
javax.swing.RootPaneContainer
java.io.Serializable

Lifecycle of an Applet

- You cannot "run" an applet, rather you must "load" it
- What happens then is:
  - An instance of the applet's class is created
  - The applet initializes itself with its *init()* method
  - The applet starts itself running with its *start()* method
  - When the user's mouse leaves the page or iconifies it, the applet has the option of stopping itself
  - When the user returns to the page or reopens the icon, the applet can start itself again
  - Before an applet is unloaded, it's given the chance to stop itself
- There is no constructor for an applet, because loading the applet automatically creates the instance

The Applet Lifecycle Methods

```
public class Simple extends JApplet {
        ...
        public void init() { . . . }
        public void start() { . . . }
        public void stop() { . . . }
        public void destroy() { . . . }
        ...
}
```

- The *init()* method contains the initialization code that, otherwise, would be put into the constructor of an object
- Most applets override the *start()* method provided by the JApplet class; the *start()* method performs the applet's work or starts up one or more threads to do so
- The *stop()* method is useful for stopping threads
- If a user leaves the applet's page, it is appropriate to stop the animation threads
- The *destroy()* method allows an applet to release resources and close sockets cleanly

Using a Swing GUI inside an Applet

- You can use all of the Swing facilities to build a GUI for applets that extends the JApplet class, including buttons, menus, etc, and you can use a layout manager to arrange them nicely
- After adding one or more GUI components, you must invoke *validate()* to make sure that the browser displays them
- You must also invoke *repaint()* to display each change

```
mb = new JMenubar();
Menu m = new JMenu("Menu");
m.add(new JMenuItem("Menu item 1"));
m.add(new JMenuItem("Menu item 2"));
mb.add(m);
setJMenuBar(mb);
validate();
```

Using a Swing GUI inside an Applet

- You can also use Swing's event handling mechanisms
  - You must implement the **event handler interface**, as in

    public class TextDemo extends JApplet

    implements ActionListener {

    ...
  - You must provide the **event handling method**, as in

    public void actionPerformed(ActionEvent evt) {

    ...
- These work exactly as in a standard Swing GUI

A Simple JApplet

```
import java.applet.Applet;
import java.awt.Graphics;
import javax.swing.*;

public class Simple extends JApplet {
    StringBuffer buffer;
    public void init() {
        buffer = new StringBuffer();
        addItem("initializing ... ");
    }
    public void start() {
        addItem("starting ... ");
    }
    public void stop() {
        addItem("stopping ... ");
    }
    public void destroy() {
        addItem("preparing for unloading ... ");
    }
```

A Simple JApplet (cont)

```
    void addItem(String newWord) {
        System.out.println(newWord);
        buffer.append(newWord);
        repaint();
    }

    public void paint(Graphics g) {
        // Draw a Rectangle around the applet's display area
        g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);

        // Draw the current string inside the rectangle
        g.drawString(buffer.toString(), 5, 15);
    }
}
```

Applet Security Restrictions

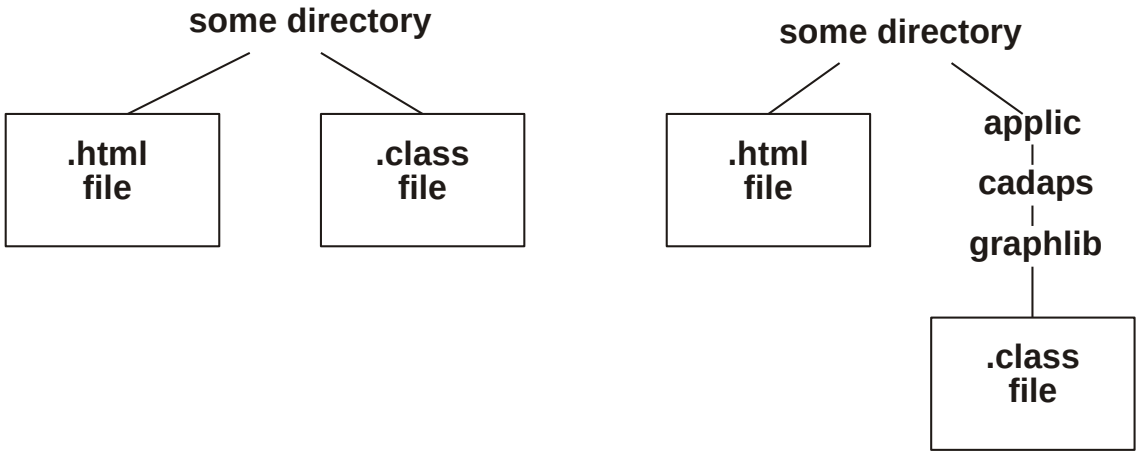
- Because an applet is a guest on the computer on which it is running, an applet is subject to certain security restrictions:
  - An applet cannot load libraries or define native methods (code written in a non-Java language, typically C/C++)
  - It cannot make network connections except to the computer from which it came
  - It cannot read or write files on the computer on which it is executing, except on its own computer
  - It cannot start any program on the computer on which it is executing
  - It cannot read certain system properties (e.g., processor type, memory size, etc)
  - Windows that an applet brings up look a bit different from windows that an application brings up
- If a security violation occurs, the JVM throws a *SecurityException*
- The applet can catch the *SecurityException* and take action

What CAN an Applet Do?

- An applet running within a Web browser can cause HTML documents to be displayed
- It can play sounds, which other Java programs can't do, because browsers have interface to sound, but the JVM does not
- It can invoke public methods of other applets on the same page
- An applet can make network connections to the computer from which it came
- An applet that is loaded from the local file system (from a directory in the user's *CLASSPATH*) are not subject to the restrictions imposed on applets loaded over the network
- Most applets should stop running once you leave their page (e.g., if the applet involves animation that uses a lot of processing cycles)

Creating an Applet in an HTML file

- You can include an applet in an HTML file by including
  <APPLET CODE=xxx.class WIDTH=123 HEIGHT=123>
  </APPLET>
- The *<APPLET>* tag tells the browser to load the applet whose *Applet* class is named *xxx.class*
- The class file of the applet must be in the same directory as the HTML file that contains the tag, or in a subdirectory of that directory



Running an Applet

- When a Java-enabled browser encounters an <APPLET> tag in an HTML document, it
  - Reserves a display area of width and height for the applet
  - Loads the byte codes for the specified Applet class
  - Creates an instance of the Applet class
  - Calls the instance's *init()* and *start()* methods
- An applet can use other classes

  These classes can be local to the browser, part of the Java environment, or custom classes that you supply
- When an applet tries to use a class for the first time, the browser looks for it on the computer running the browser
  If the browser can't find it there, it looks for the class on the computer from which the applet came
- When the browser finds the class, it loads the byte codes for that class and continues executing the applet

Tag Options

- The *<APPLET>* tag has many options that you can use to customize your applet's execution
- *CODEBASE* allows you to tell the browser which directory to use to find the applet's class files

```
<APPLET  CODEBASE="example/"  CODE=Simple.class
      WIDTH=500  HEIGHT=20>
</APPLET>
```

- *CODEBASE* can specify an absolute URL or a URL relative to the directory of the HTML file
- If the above tag occurs in mydirectory/myfile.html, the applet is in mydirectory/example/Simple.class

Displaying a Document

- An applet can display an HTML document, using one of two methods:
  - *public void showDocument(java.net.URL url)*
  - *public void showDocument(java.net.URL url, String targetWindow)*
- Here url is some URL (absolute or relative) and targetWindow is one of the string values:
  - *"_blank"* -- Display the document in a new, nameless window
  - *windowName* -- Display the document in a window with the indicated name; this window is created if necessary
  - *"_self"* -- Display the document in the window and frame that contain the applet
  - *"_parent"* -- Display the document in the applet's window but in the parent frame of the applet's frame
  - *"_top"* -- Display the document in the applet's window but in the top-level frame

Sending Messages to Other Applets

- Applets can find other applets and send messages to them (invoke methods on them) with certain security restrictions:
  - The applets must originate in the same directory on the same machine (the same codebase)
  - The applets must be running on the same page in the same browser window
- An applet can find another applet by
  - Looking it up by name, using the *getApplet()* method of the *AppletContext* class, or
  - Finding all the applets on the page, using the *getApplets()* method of the *AppletContext* class
- Both methods, if successful, give the caller one or more *Applet* objects
- Once the caller has a applet object, the caller can invoke methods of the object

Giving an Applet a Name

- By default, an applet has no name. For an applet to have a name, you must specify it in the HTML code that adds the applet to a page
- You can specify an applet's name in two ways:
  - *<APPLET CODEBASE=example/ CODE=Sapplet.class WIDTH=450 HEIGHT=200 NAME="buddy"> ....*

    </APPLET>
  - *<APPLET CODEBASE=example/ CODE=Rapplet.class WIDTH=450 HEIGHT=35>*

    <PARAM NAME="NAME" VALUE="buddy"> ....

    </APPLET>
- The first is shorthand for the second. Both create a parameter with name *"NAME"*, whose type is String and whose value is *"buddy"*
- An applet can get its own name
  - *myName = getParameter("NAME");*

Finding Another Applet

- The *getApplet()* method looks through all of the applets on the current page to see if one of them has the specified name
- If so, *getApplet()* returns the applet object
- In the example below, *AppletContext* identifies the applet's *AppletContext* interface within which the *getApplet()* method is defined

```
JApplet rapplet = null;
String rappletName = ... // Set name to search for
rapplet = AppletContext().getApplet(rappletName);
```

Finding All of the Applets

- The *getApplets()* method returns an *Enumeration* of all of the applets on the page, as in

  Enumeration e = AppletContext().getApplets();

- We can then extract the applets one at a time, as in

```
while (e.hasMoreElements()) {
    JApplet anApplet = (Applet)e.nextElement();
    String name = anApplet.getClass().getName();
```

- Here the *(Applet)* is a "cast" that tells the JVM that the *nextElement* in *e* is an *Applet*
- If it is not an *Applet*, you will get garbage

One Applet Invoking a Method of Another Applet

- Once you have an object for an applet, you can invoke a method of that applet, as in

```
Applet rapplet = null;
String rappletName = ... // Set name to search for
rapplet = AppletContext().getApplet(rappletName);
...
((Rapplet)rapplet).processRequestFrom(sappletName);
```

- Here, sapplet is invoking a method of rapplet
- *processRequestFrom()* is a method of class *Rapplet* and *(Rapplet)* is a cast that tells the JVM that the object *rapplet* is of class *Rapplet*

One Object Telling Another Object How to Invoke It

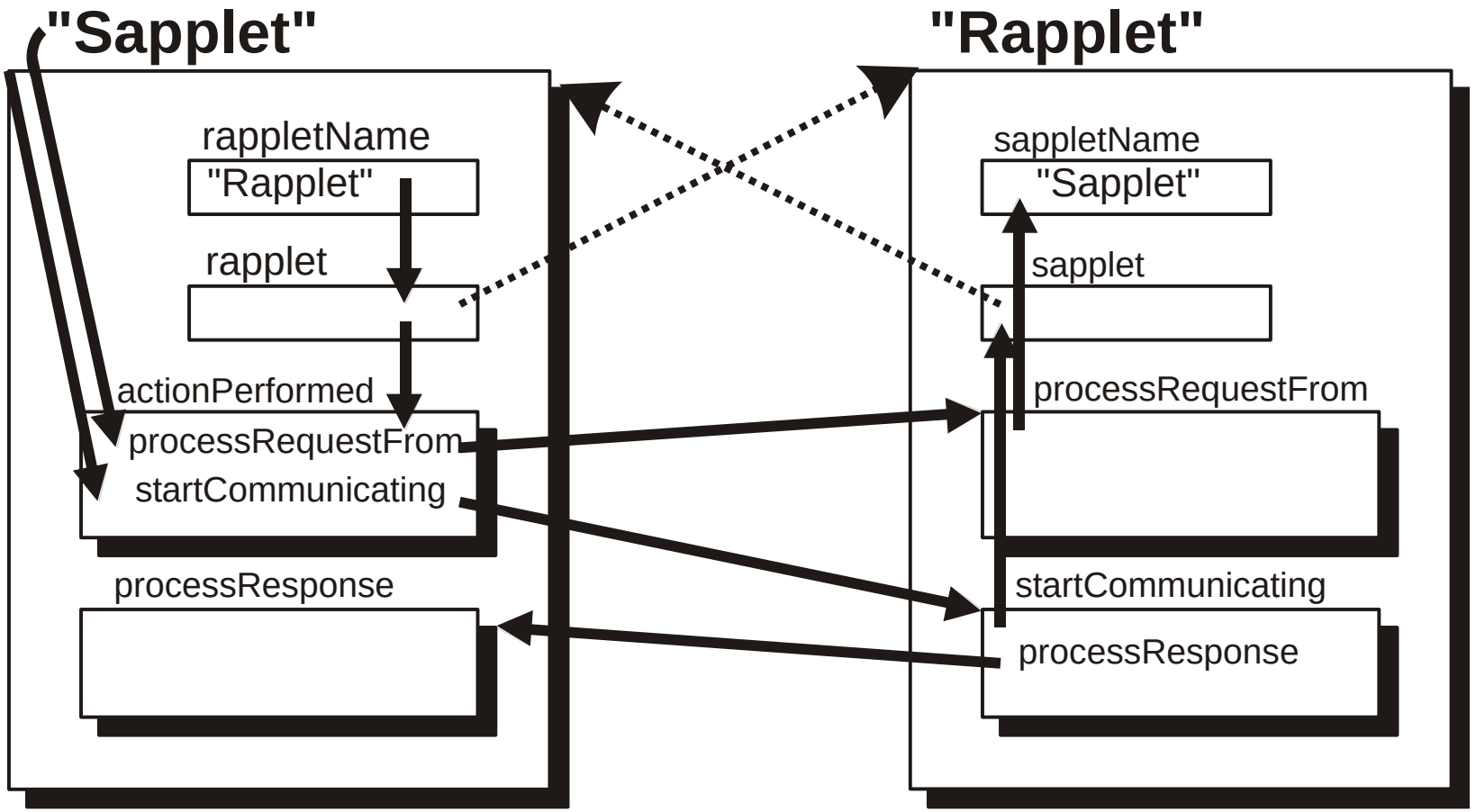
- One object (sapplet) can send its own applet object *this*, as a parameter, to another object (rapplet) so that the other object can later invoke a method of it, as in

  ((Rapplet)rapplet).startCommunicating(this);

- Here, *startCommunicating()* is a method of the class Rapplet that expects an applet object as a parameter
- Now, rapplet can invoke a method of sapplet, as in

  ((Sapplet)sapplet).processResponse(...);

Objects Invoking Each Other's Methods



Objects Invoking Each Other's Methods

- Sapplet has Rapplet's name
- *processRequestFrom()* is a method of the class *Rapplet*
- *startCommunicating()* is a method of the class *Rapplet*
- *processResponse()* is a method of the class *Sapplet*

1. Sapplet invokes *getApplet()* on Rapplet's name to get a reference to Rapplet's object
2. Sapplet stores Rapplet's object reference
3. Sapplet invokes *processRequestFrom()* on Rapplet, passing its own name
4. Rapplet stores Sapplet's name
5. Sapplet invokes *startCommunicating()* on Rapplet, passing *this*, which is its own object reference
6. Rapplet stores Sapplet's object reference
7. Rapplet invokes *processResponse()* on Sapplet