

COMP455- OBJECT ORIENTED PROGRAMMING

Early intro into (basics)
Applets (more on Chapter 20)
&
More on..
..Java methods

Dr. Constandinos X. Mavromoustakis

Java applets



Introduction

Java applets are one of three kinds of Java programs:

- An *application* is a standalone program that can be invoked from the **command line**.
- An *applet* is a program that **runs in the context of a browser session**.
- A *servlet* is a program that is invoked on a server program, and it **runs in the context of a web server process**.



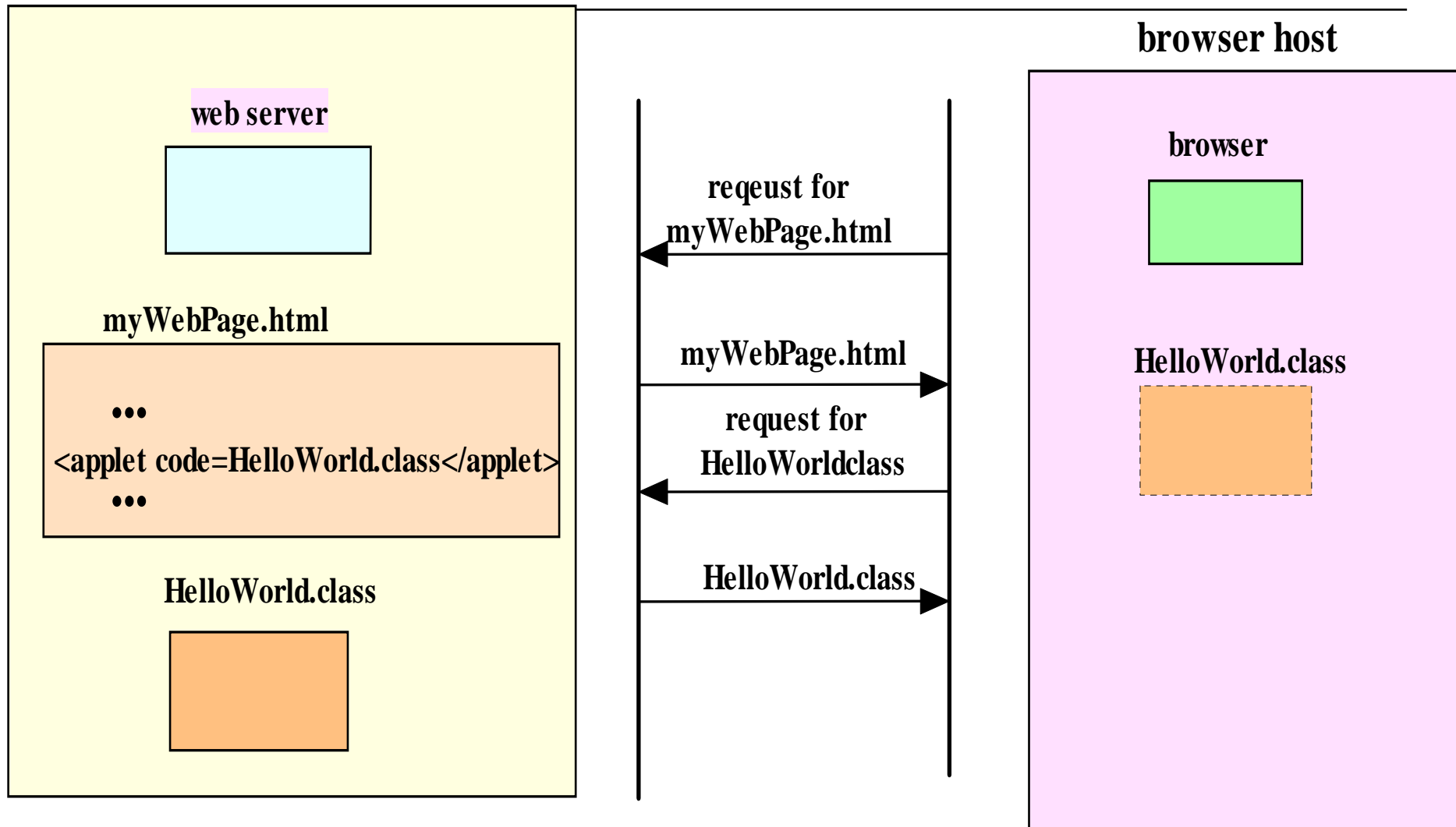
Applets, web page, client, server

- **Applets** are programs stored on a **web server**, similar to web pages.
- When an applet is referred to in a web page that has been fetched and processed by a browser, **the browser generates** a request to **fetch** (or **download**) **the applet program**, then **executes the applet program** in the browser's execution context **on the client host**.

Applets, web page, client, server

server host

browser host





Early intro into (basics)

Applets (more on Chapter 20)

- – Java programs embedded in web pages
- – when page is opened compiled Java code is
- transferred to client and run there
 - client-side programs
- – we are going to look at
 - structure and coding of applets
 - `<APPLET>` tag



Lifecycle of an applet

- `init()`
 - initialises the applet each time the page is loaded
 - • code that would normally be put in a constructor
 - • can also do things like start downloading any files you need
 - • often all code goes here
- • `start ()`
 - • often code that performs function of applet
 - • code dealing with threads or time (e.g. playing sounds)
 - – called when applet becomes visible
- • `stop()`
 - stops the applet's execution
 - – Useful if application is resource-heavy
browser is quit or user leaves the page
- • `destroy ()`
 - performs a clean-up for unloading before browsing quitting
 - not always run (browser may crash)
 - usually do not override this method

Example

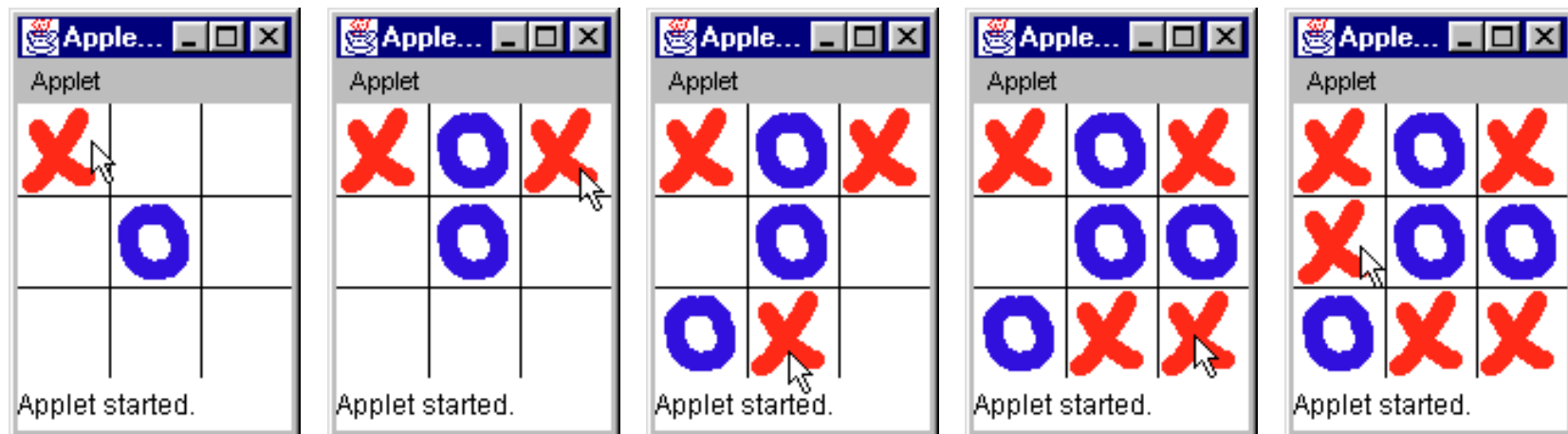
no main method

(for an applet that plays a video file)

- • `init()`
 - draw the controls and start loading the video file
- • `start()`
 - • wait until the file was loaded, and then start playing it
- • `stop()`
 - • pause the video, but not rewind it.
 - • if `start()` were called again, the video picks up where it left off
 - it would not start over from the beginning
 - • `destroy()` then `init()`
 - • video starts over from the beginning
- *note: no main method*

Ex. TicTacToe Applet

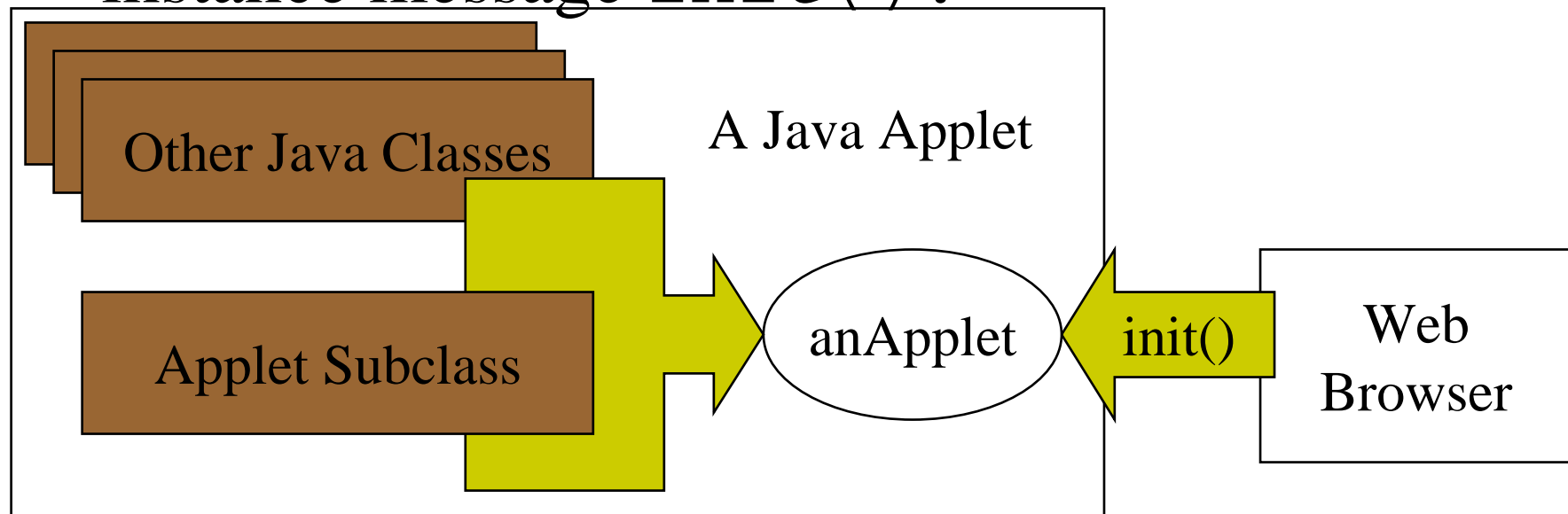
- You start as player "X"



Sample execution of the **TicTacToe** applet.

Review Java Applets - launching

- When the web browser reads a document that tells it to load an applet, it creates an instance of your applet subclass and sends it the instance message `init()`.





Java Applets - `init`

- The `init()` message creates all of the graphical objects in the applet, like buttons and fields and puts them into your applet object.
- If you do not want to put any graphical objects in your applet, you do not need to implement an `init()` method in your applet subclass.



Java Applets - paint

- Whenever your applet must be displayed, the paint message is sent to your applet.
- For example, the paint message is sent after your applet is first initialized and any time the screen must be refreshed.
- The protocol for the paint message is:
public void paint(Graphics aGraphics);
- The paint method in your applet subclass must display any objects that you did not put in your applet with the init() method.

The DrawTest Applet



Sample execution of applet
DrawTest.



Applet Execution - 1

- An applet program is written as a subclass of the `java.Applet` class or the `javax.swing.JApplet` class.
- There is **no `main()`** method in an Applet.
- An applet uses **AWT** for graphics, or **JApplet**, a subclass of `javax.swing`.



Applet Execution - 2

□ **Life Cycle of an Applet:**

- **init:** This method is intended for whatever initialization is needed for an applet.
- **start:** This method is automatically called after init method. It is also called whenever user returns to the page containing the applet after visiting other pages.
- **stop:** This method is automatically called whenever the user moves away from the page containing applets. This method can be used to stop an animation.
- **destroy:** This method is only called when the browser shuts down normally.
- Ref: <http://java.sun.com/docs/books/tutorial/deployment/applet/index.html/>



Applet Execution - 3

- The applet is running and rendered **on the web page.**
- Every Applet needs to implement **one of more** of the **init()**, the **start()** and the **paint()** methods.
- At the end of the execution, the **stop()** method is invoked, followed by the **destroy()** method to **deallocate the applet's resources.**



Applet Security

For security reasons, applets that are loaded over the network have **several restrictions**.

- an **applet cannot** ordinarily **read** or **write** files on the computer that it's executing on.
- an applet **cannot** **make network connections** except to the **host** that it came from.

■ Ref:

<http://java.sun.com/docs/books/tutorial/deployment/applet/index.html/>



What Are Applets?

- ❑ An *applet* is a special Java program that can be embedded in HTML documents.
- ❑ It is automatically executed by (applet-enabled) web browsers.
- ❑ In Java, non-applet programs are called *applications*.



Application vs. Applet

□ Application

- Trusted (i.e., has full access to system resources)
- Invoked by Java Virtual Machine (JVM, java), e.g.,
java HelloWorld
- Should contain a main method, i.e.,
`public static void main(String[])`

□ Applet

- Not trusted (i.e., has limited access to system resource to prevent security breaches)
- Invoked automatically by the web browser
- Should be a subclass of class `java.applet.Applet`

Example

HelloWorld.java



Examples

- HelloWorld.java

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

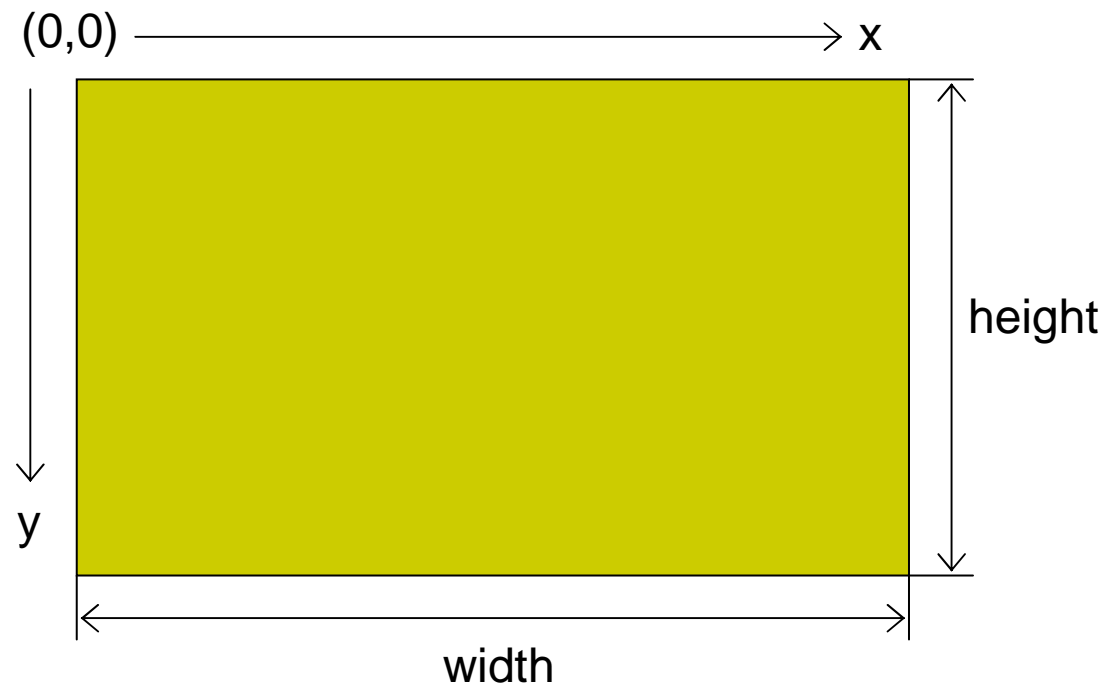
- HelloWorldApplet.java

First Java Applet

- Java source in HelloWorldApplet.java

```
import java.awt.*;
import java.applet.Applet;
public class HelloWorldApplet extends Applet {
    public void paint(Graphics g) {
        Dimension d = getSize();
        g.setColor(Color.BLACK);
        g.fillRect(0, 0, d.width, d.height); // paint background
        g.setFont(new Font("San-serif", Font.BOLD, 24));
        g.setColor(new Color(255, 215, 0));
        g.drawString("Hello, world!", 60, 40);
        g.drawImage(getImage(getCodeBase(), "Rabbit.jpg"),
            20, 60, this);
    }
}
```

Graphics Coordinate



Embedding Applet into HTML

□ HTML source in HelloWorld.html

```
<!--HelloWorld.html-->
<html>
  <head>
    <title>HelloWord</title>
  </head>
  <body>
    <center>
      <applet code="HelloWorldApplet.class"
        width=300 height=350></applet>
    </center>
    <hr/>
    <a href="HelloWorldApplet.java">The source.</a>
  </body>
</html>
```



Compiling and Running

- To compile

```
javac HelloWorldApplet.java
```

```
Produces HelloWorldApplet.class
```

- To run

- Open page `HelloWorld.html` from web browser or

- Use appletviewer of JDK

```
appletviewer HelloWorld.html
```

More on html tags

HTML tags for applets

HTML tags for applets - 1

<APPLET

// the beginning of the HTML applet code

CODE="demoxx.class"

// the actual name of the applet (usually a 'class' file)

CODEBASE="demos/"

// the location of the applet (relative as here, or a full URL)

NAME="SWE622"

// the name of the instance of the applet on this page

WIDTH="100"

// the physical width of the applet on the page

HEIGHT="50"

// the physical height of the applet on the page

ALIGN="Top"

// align the applet within its page space (top, bottom, center)

HTML tags for applets - 2

```
<APPLET CODE="SWE622.class" CODEBASE="example/"  
        WIDTH=460 HEIGHT=160  
        NAME="buddy" >  
<PARAM NAME="imageSource" VALUE="images/Beans">  
<PARAM NAME="backgroundColor" VALUE="0xc0c0c0">  
<PARAM NAME="endImage" VALUE=10>  
</APPLET>
```

The HelloWorld Applet

```
<HTML>
<BODY>
<APPLET code=hello.class width=900 height=300>
</APPLET>
</BODY>
</HTML>
```

```
// applet to display a message in a window
import java.awt.*;
import java.applet.*;

public class hello extends Applet {
    public void init() {
        setBackground(Color.yellow);
    } // end of init()
```

```
    public void paint(Graphics g) {
        final int FONT_SIZE = 42;
        Font font = new Font("Serif",
            Font.BOLD, FONT_SIZE);
        // set font, and color and display message
        // on the screen at position 250,150
        g.setFont(font);
        g.setColor(Color.blue);
        // The message in the next line is the one
        // you will see
        g.drawString("Hello,
            world!",250,150);
    } // end of paint()

} // end of hello
```

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

```
<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

```
<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>
```



Reading Applet Parameters

- Use `getParameter(name)` to retrieve the value of the PARAM element and 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





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



Graphics 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 `Graphics` object
 - In JDK 1.1, only 5 fonts are available: `Serif` (aka `TimesRoman`), `SansSerif` (aka `Helvetica`), `Monospaced` (aka `Courier`), `Dialog`, and `DialogInput`

Example

Example

```
/**
 * This is a simple applet.
 */

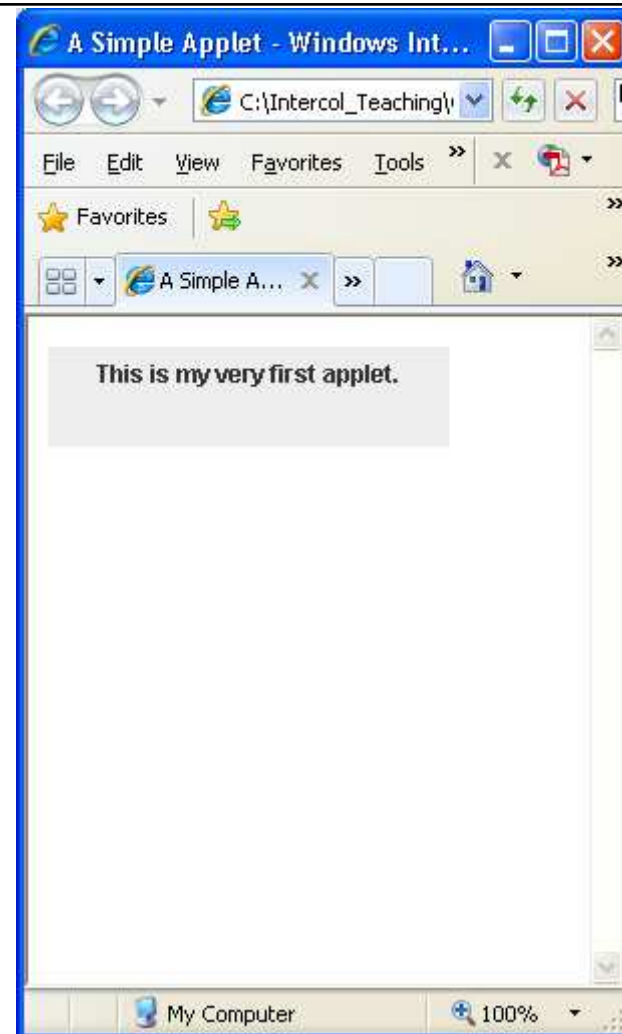
public class SimpleApplet extends JApplet
{
    /**
     * The init method sets up the applet, much
     * like a constructor.
     */

    public void init()
    {
        // Create a label.
        JLabel label =
            new JLabel("This is my very first applet.");

        // Set the layout manager.
        setLayout(new FlowLayout());

        // Add the label to the content pane.
        add(label);
    }
}
```

Output



Example

Run it on your terminal!

Example

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

/**
 * This applet demonstrates how the mouse adapter
 * classes can be used.
 */

public class DrawBoxes2 extends JApplet
{
    private int currentX = 0; // Current X coordinate
    private int currentY = 0; // Current Y coordinate
    private int width = 0; // Rectangle width
    private int height = 0; // Rectangle height

    /**
     * init method
     */

    public void init()
    {
        // Add a mouse listener and a mouse motion listener.
        addMouseListener(new MyMouseListener());
        addMouseMotionListener(new MyMouseMotionListener());
    }
}
```

```

/**
 * paint method
 */

public void paint(Graphics g)
{
    // Call the superclass's paint method.
    super.paint(g);

    // Draw a rectangle.
    g.drawRect(currentX, currentY, width, height);
}

/**
 * Mouse listener class
 */
private class MyMouseListener extends MouseAdapter
{
    public void mousePressed(MouseEvent e)
    {
        // Get the mouse cursor's X and Y coordinates.
        currentX = e.getX();
        currentY = e.getY();
    }
}

/**
 * Mouse Motion listener class
 */
private class MyMouseMotionListener
    extends MouseMotionAdapter
{
    public void mouseDragged(MouseEvent e)
    {
        // Calculate the size of the rectangle.
        width = e.getX() - currentX;
        height = e.getY() - currentY;

        // Repaint the window.
        repaint();
    }
}
}

```



The *.html file should contain the following:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Draw Boxes</TITLE>
```

```
<BODY>
```

```
<APPLET CODE="DrawBoxes2.class" WIDTH=400  
    HEIGHT=300>
```

```
</APPLET>
```

```
</BODY>
```

```
</HTML>
```



Summary

- An **applet** is a Java class
- Its code is **downloaded** from a web server
- It runs in the browser's environment on the client host
- It is invoked by a browser when it scans a web page and encounters a class specified with the *APPLET* tag
- For security reason, the execution of an applet is normally subject to restrictions:
 - applets cannot access files in the file system on the client host
 - Applets cannot make network connection exception to the server host from which it originated