Session 12

Java Applets

Outline

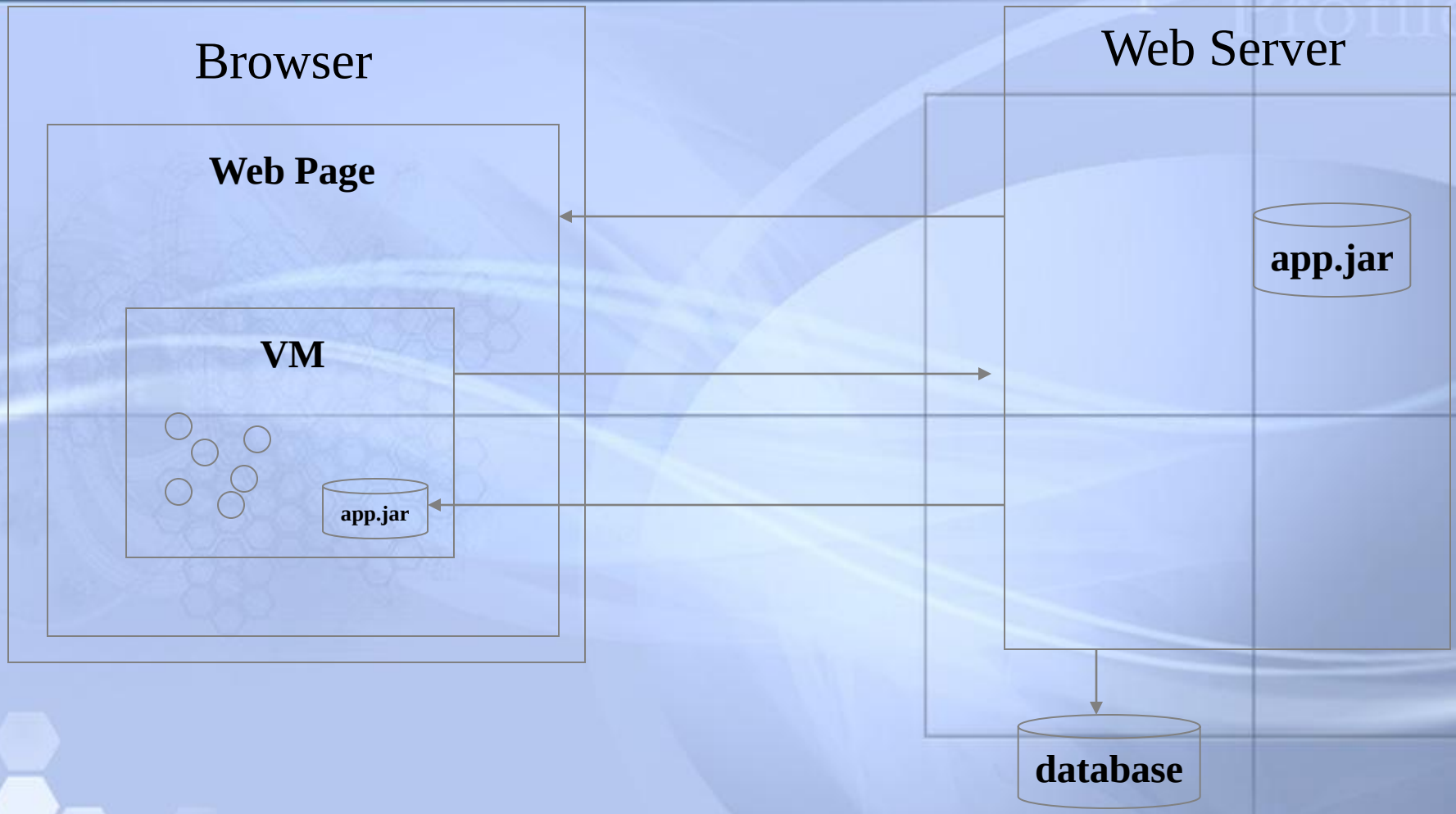
- What is an Applet?
- Why an Applet Work?
- HTML and Applet
- Graphical Applet
- GUI Elements in Applet

What is an Applets?

What is an Applets?

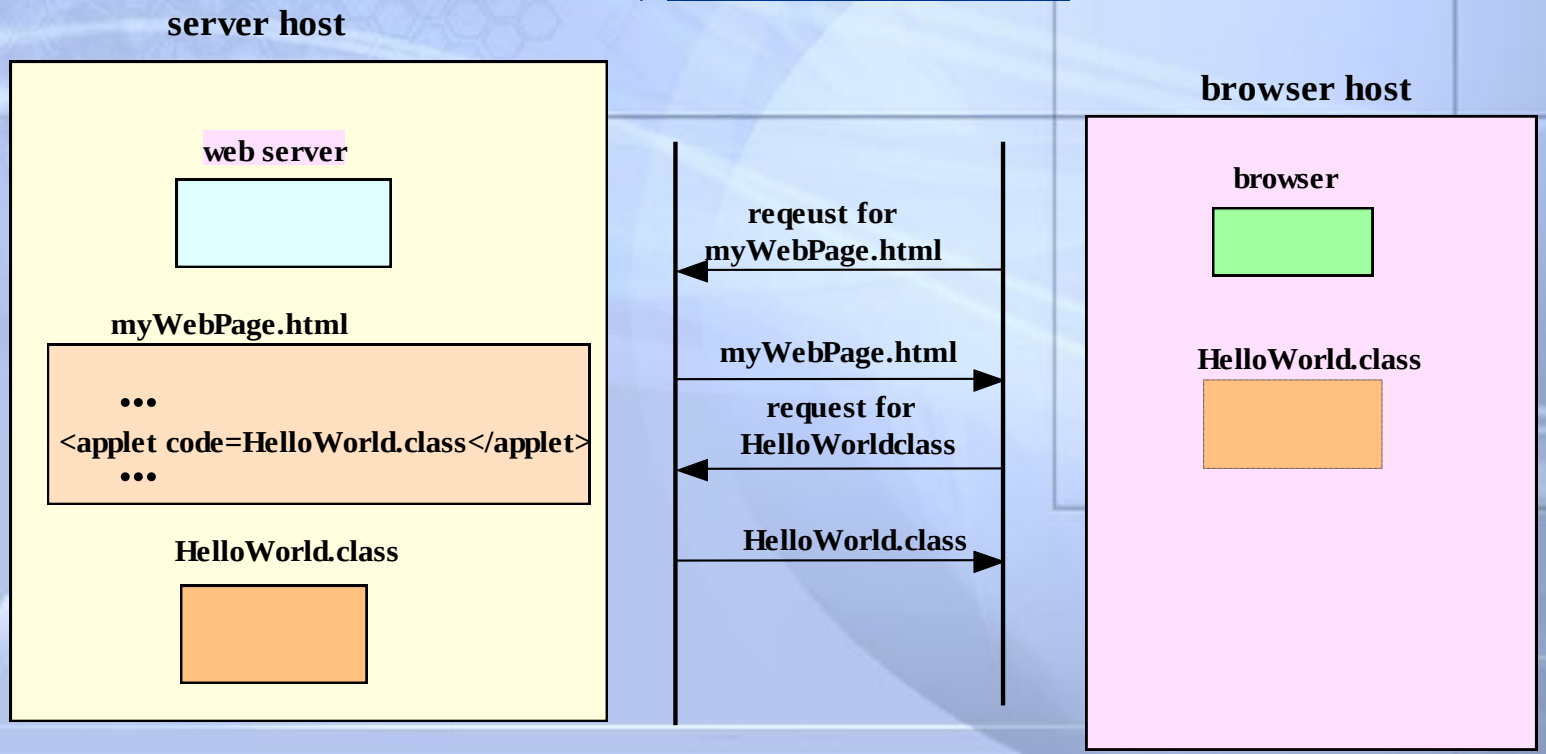
- Applet is a special Java program that can be embedded in HTML documents and running within a Web browser, or executed by appletviewer during the development.
- It is automatically executed by (applet-enabled) web browsers.
- In Java, non-applet programs are called a*pplications*.

Applet Architecture

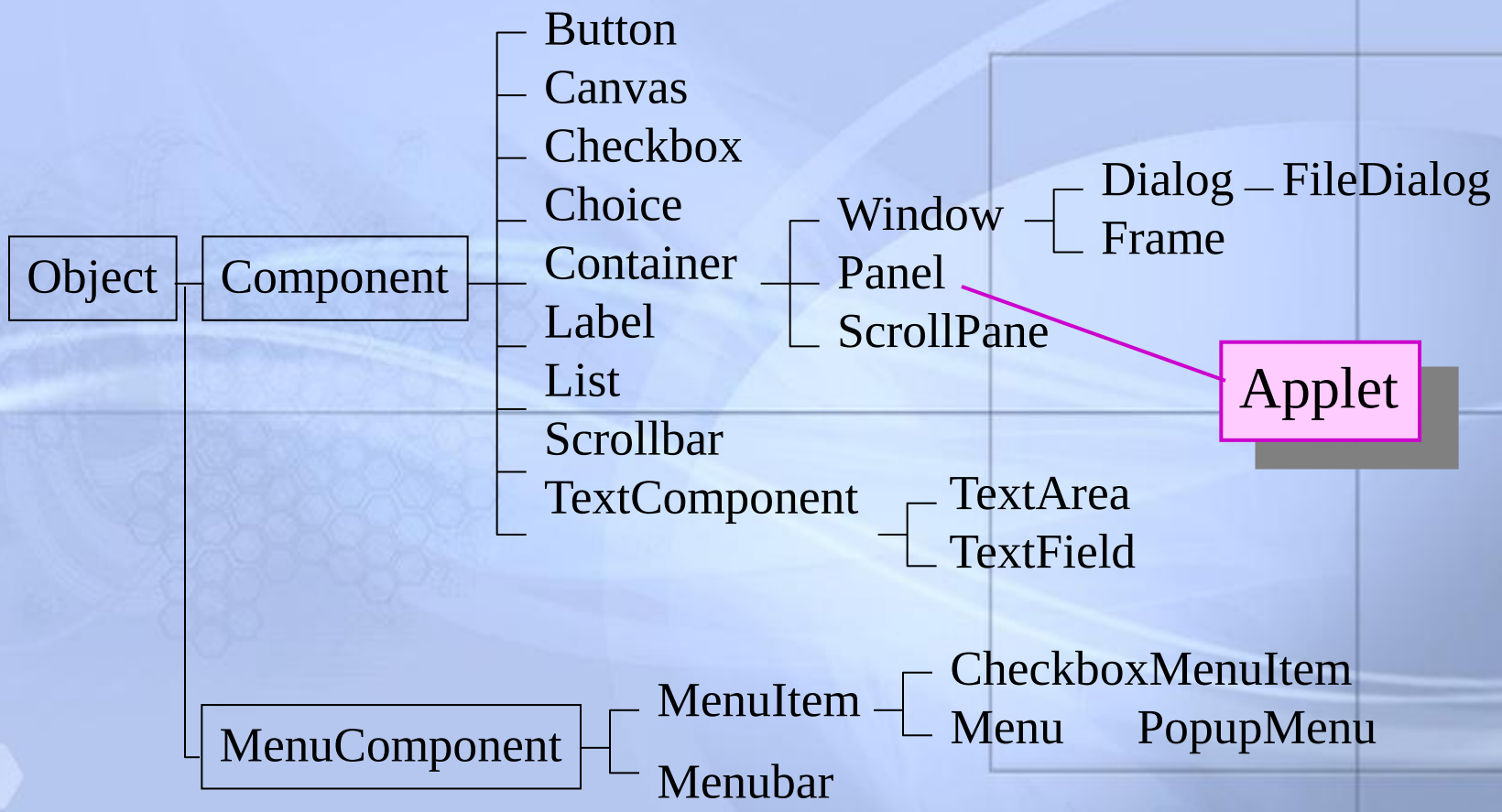


Applets, web page, client, server

- 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 program in the browser's execution context, on the client host.



Where is Applet?



Application, Applet, Servlet

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 demand on a server program and that runs in the context of a web server process.

Conversions Between Applications and Applets

- Conversions between applications and applets are simple and easy.
- You can always convert an applet into an application.
- You can convert an application to an applet as long as security restrictions are not violated.

How Applets Differ from Applications

- Although both the Applets and stand-alone applications are Java programs, there are certain restrictions are imposed on Applets due to security concerns:
  - Applets don't use the main() method, but when they are load, automatically call certain methods (init, start, paint, stop, destroy).
  - They are embedded inside a web page and executed in browsers.
  - They cannot read from or write to the files on local computer.
  - They cannot communicate with other servers on the network.
  - They cannot run any programs from the local computer.
  - They are restricted from using libraries from other languages.
- The above restrictions ensures that an Applet cannot do any damage to the local system.

Java Applet Methods

- **Applets & HTML**
  - default methods (init, paint, …)
  - APPLET & OBJECT tags, applet parameters
- **Graphical Applets**
  - Graphics object: drawString, drawLine, drawOval, drawRect, …
  - double buffering
- **GUI Applets**
  - GUI elements, layout, event handling

Applet methods

- public void init ()
- public void start ()
- public void stop ()
- public void destroy ()
- public void paint (Graphics)

Also:

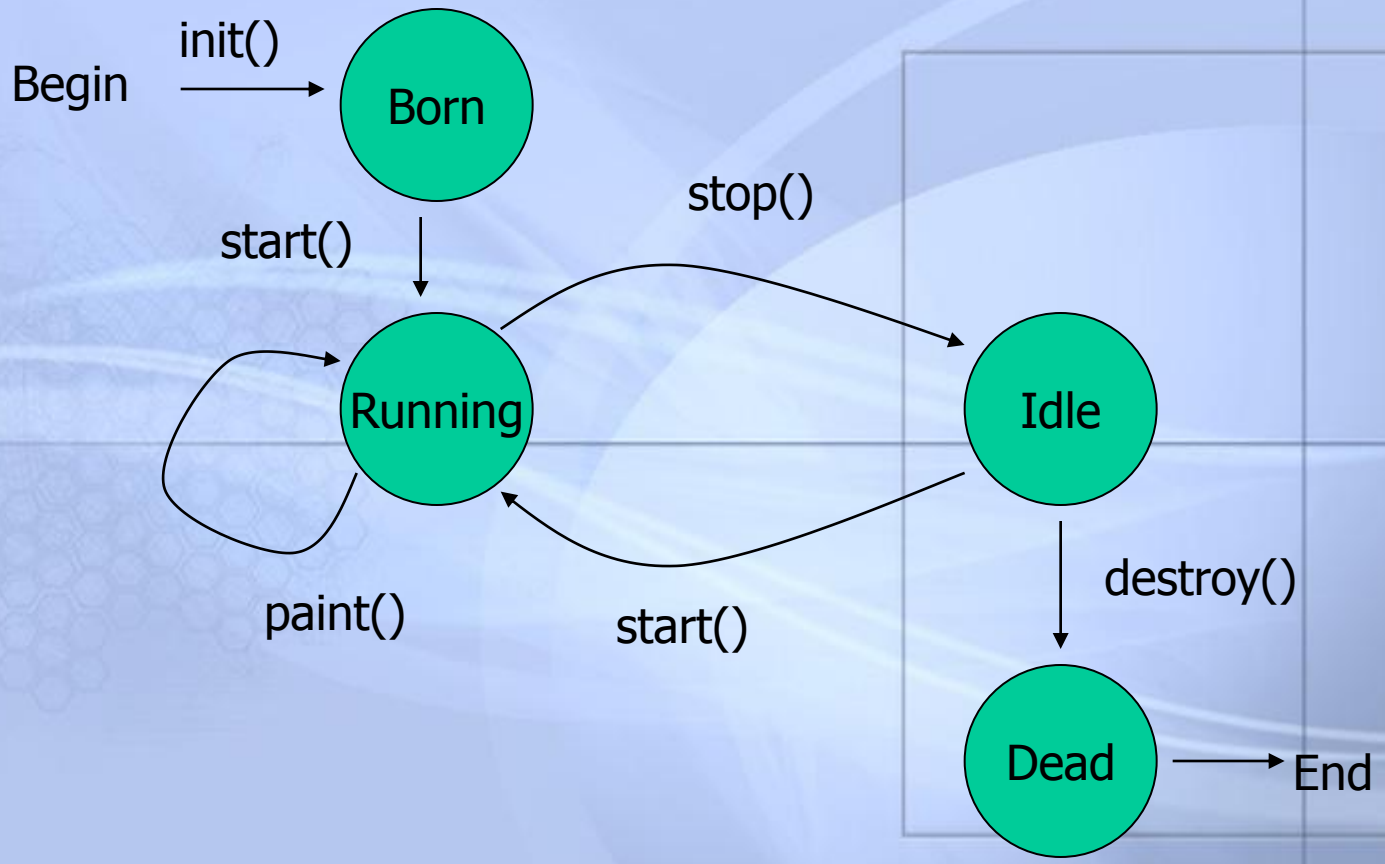
- public void repaint()
- public void update (Graphics)
- public void showStatus(String)
- public String getParameter(String)

Why an Applet Works?

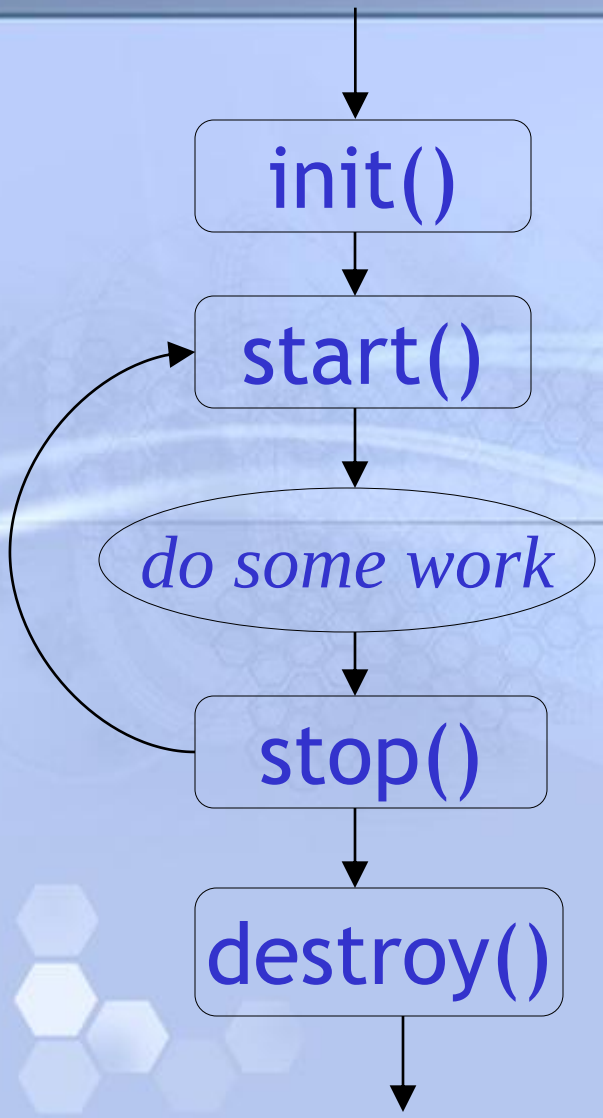
Why an Applet Works?

- Always extends the **`Applet`** class, for swing components it always used the **JApplet** class
- Override **`init()`**, **`start()`**, **`stop()`**, and **`destroy()`** if necessary. By default, these methods are empty.
- Add your own methods and data if necessary.
- Applets are always embedded in an HTML page.

Applet Life Cycle Diagram



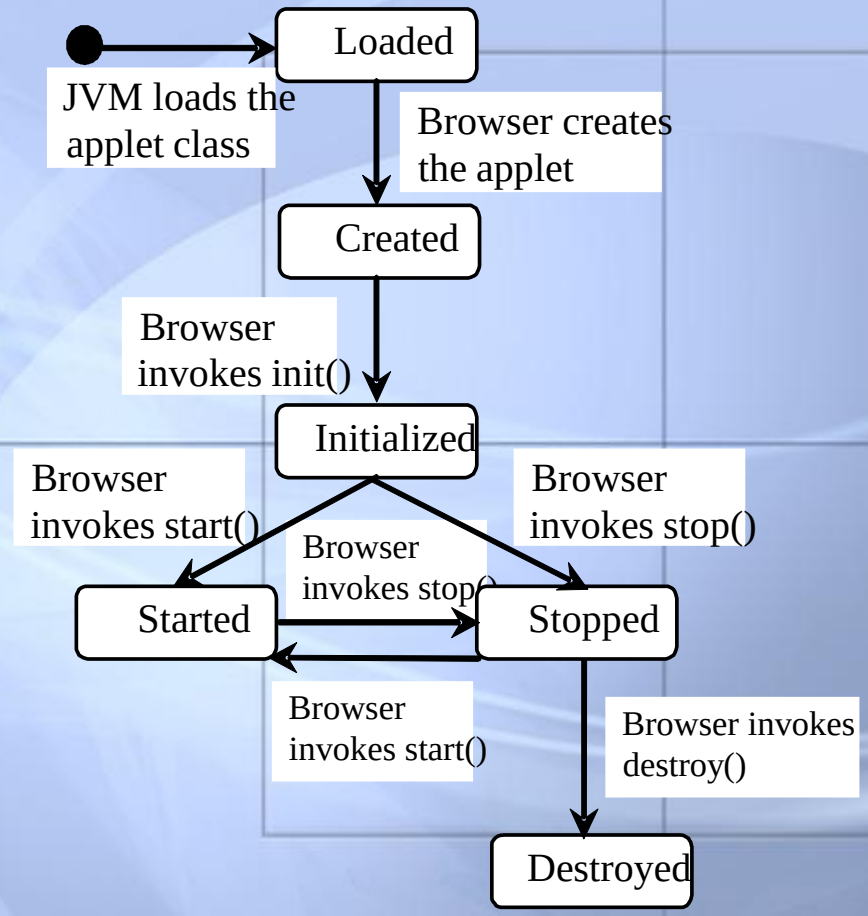
Methods are called in this order



- init and destroy are only called once each
- start and stop are called whenever the browser enters and leaves the page
- *do some work* is code called by your *listeners*
- paint is called when the applet needs to be repainted

Browser Calling Applet Methods

1. JVM loads the applet class
2. Browser creates the applet.
3. Browser invokes init().
4. Browser invokes start() when a page is loaded and restarted.
5. Browser invokes stop() when the browser released the page.
6. Browser invokes destroy() to release system resources.



public void init ( )

- init() is the first method to execute
  - **init() is an ideal place to initialize variables**
  - init() is the best place to define the GUI Components (buttons, text fields, checkboxes, etc.), lay them out, and add listeners to them
  - Almost every applet you ever write will have an init( ) method

start( ), stop( ) and destroy( )

- start() and stop( ) are used when the Applet is doing time-consuming calculations that you don't want to continue when the page is not in front
- public void start() is called:
  - **Right after init( )**
  - Each time the page is loaded and restarted
- public void stop( ) is called:
  - When the browser leaves the page
  - Just before destroy( )
- public void destroy( ) is called after stop( )
  - Use destroy() to explicitly release system resources (like threads)
  - System resources are usually released automatically

HTML and Applet

Applets in HTML

- To put an applet into an HTML page, you use the <applet> tag, which has three required attributes, code (the class file) and width and height (in pixels)
- Example:
  - <applet code="MyApplet.class" width=150 height=100> </applet>
- If your applet contains several classes, you should jar them up; in this case you also need the archive attribute:
  - <applet code="MyApplet.class" archive="MyApplet.jar"
    width=150 height=100>
    </applet>

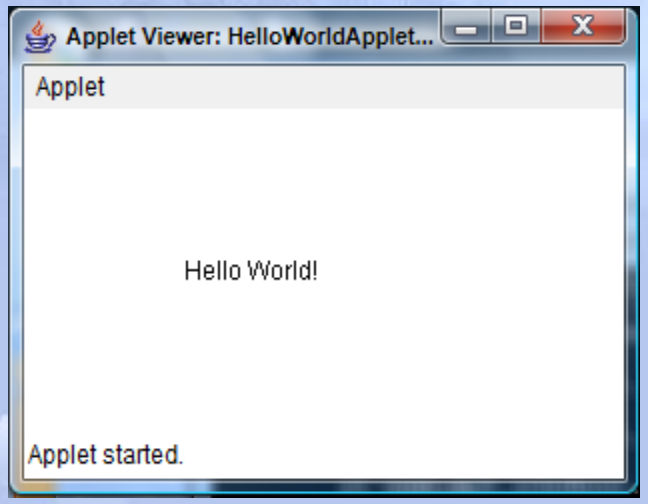
Applet in HTML

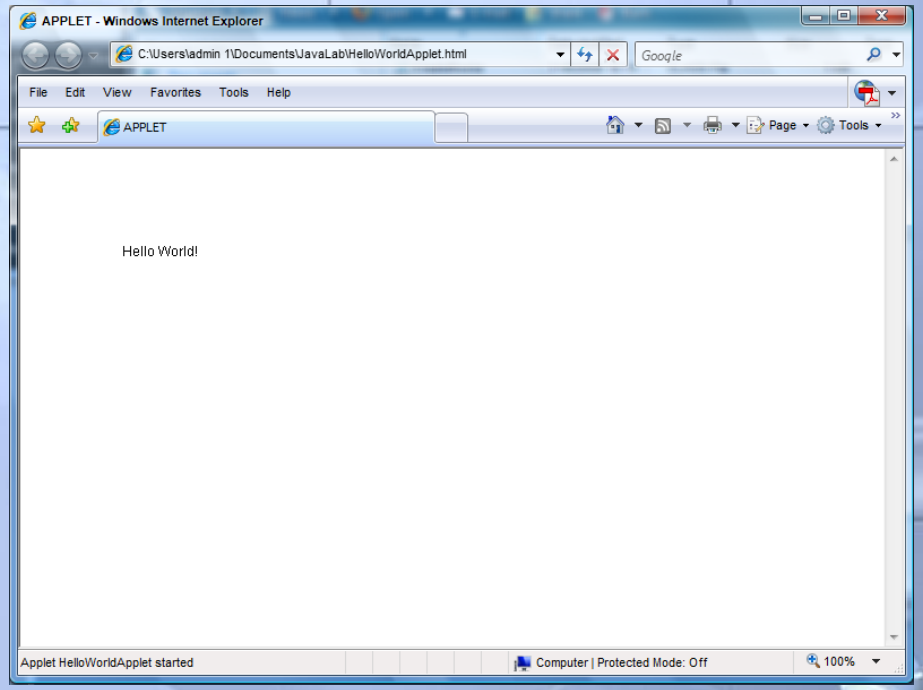
```
<html>
  <head>
    <title> Hello World Applet </title>
  </head>

  <body>
    <applet code="HelloWorld.class" width=300 height=150>
    </applet>
  </body>
</html>
```

Applet in HTML

```
Import java.applet.*;
Import java.awt.*;
public class HelloWorld extends Applet
{ public void paint(Graphic g)
    { g.drawString("Hello World", 300, 150);
    }
}
```





Applets with parameters

- You can pass parameters to your applet from your HTML page:
  - <applet code="MyApplet.class" width=150 height=100>
    <param name="message" value="Hello World">
    <param name="arraySize" value="10">
    </applet>
- The applet can retrieve the parameters like this:
  - String message = getParameter("message");
    String sizeAsString = getParameter("arraySize");
- All parameters are passed as Strings, so you may need to do some conversions:
  - try { size = Integer.parseInt (sizeAsString) }
    catch (NumberFormatException e) {…}

Applet method overriding

- Override Applet methods to customize the class
- In following example, paint() method overrides the JApplet class's method
- Example: overriding

```
public class SimpleClick extends Applet {
        public void paint(Graphics g)
        {
            // draw new image
        }
    }
```

Graphical Applet

Graphical Applet

- **paint() method is called when the applet is to be redrawn.** It is passed as object of the Graphics class, and you can use that object's methods to draw in the applet.

- **update() method is called when a portion of the applet is to be redrawn.** The default version fills the applet with the background color before redrawing the applet, which can lead to flickering when you are performing animation, in which case you would override this method.

- *Never call* paint(*Graphics*), *call* repaint( ).

Drawing Methods of Graphics

- In **paint()** method …
  - drawString(msg, x, y)
  - drawLine(x1, y1, x2, y2)
  - drawRect(x, y, wide, high)
  - drawRoundRect(x, y, wide, high, rwide, rhigh)
  - drawOval(x, y, wide, high)
  - drawArc(x, y, wide, high, startangle, wideangle)
  - drawPolyline, drawPolygon, drawString, drawImage
  - fill*
    - *3DRect, *Arc, *Oval, *Polygon, *Rect, *RoundRect

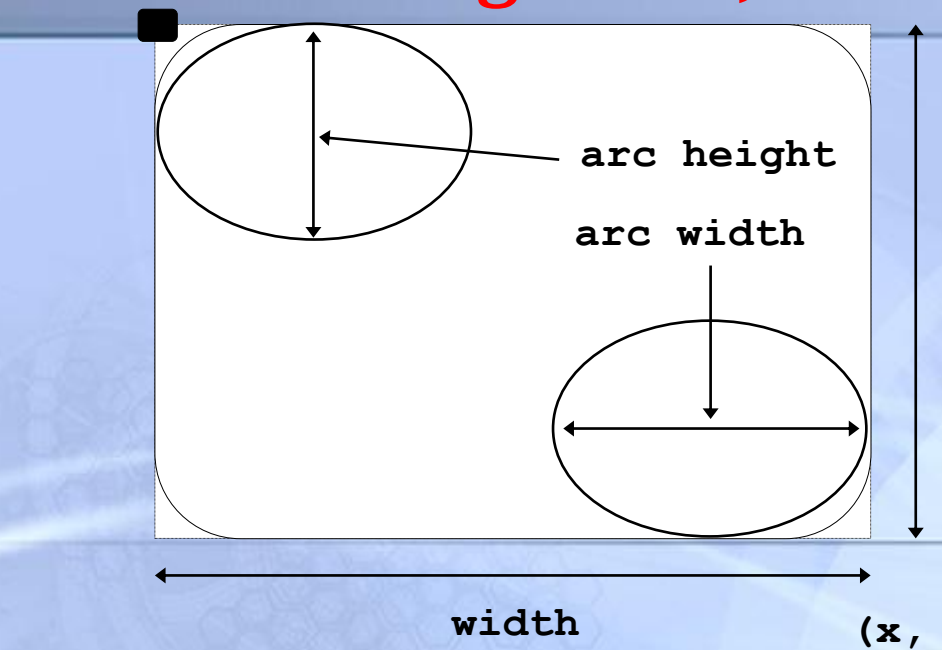
Drawing Lines, Rectangles and Ovals

- Graphics methods for drawing shapes
  - **drawLine(x1, y1, x2, y2)**
    - Line from **x1**, **y1** to **x2**, **y2**
  - **drawRect(x1, y1, width, height)**
    - Draws rectangle with upper left corner **x1, y1**
  - **fillRect(x1, y1, width, height)**
    - As above, except fills rectangle with current color
  - **clearRect (x1, y1, width, height)**
    - As above, except fills rectangle with background color
  - **draw3DRect(x1, y1, width, height, isRaised)**
    - Draws **3D** rectangle, raised if **isRaised true,** else lowered

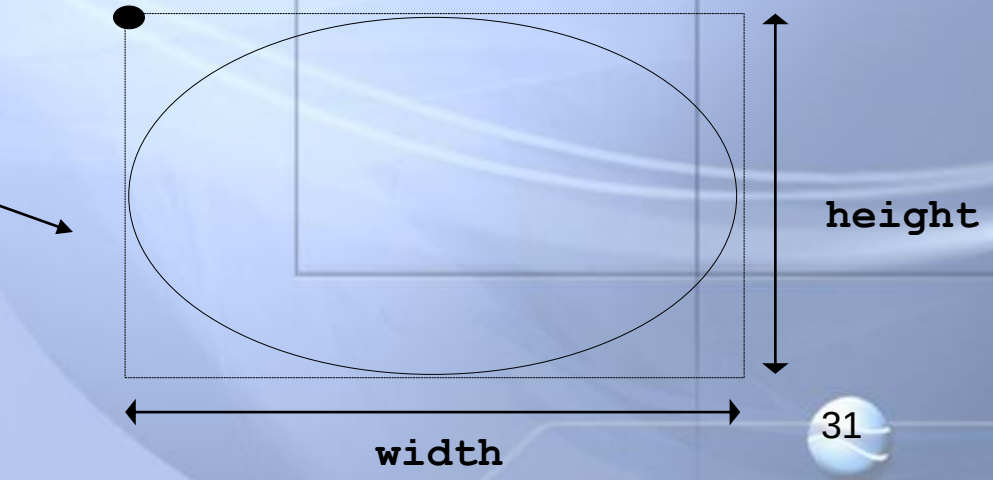
Drawing Lines, Rectangles and Ovals

- Graphics methods for drawing shapes (continued)
  - **fill3DRect**
    - As previous, but fills rectangle with current color
  - **drawRoundRect(x, y, width, height, arcWidth, arcHeight)**
    - Draws rectangle with rounded corners. See diagram next slide.
  - **fillRoundRect(x, y, width, height, arcWidth, arcHeight)**
  - **drawOval(x, y, width, height)**
    - Draws oval in bounding rectangle (see diagram)
    - Touches rectangle at midpoint of each side
  - **fillOval (x, y, width, height)**

Drawing Lines, Rectangles and Ovals



drawRoundRect parameters



drawOval parameters

Sample of Graphics methods

g.drawString("Hello", 20, 20);

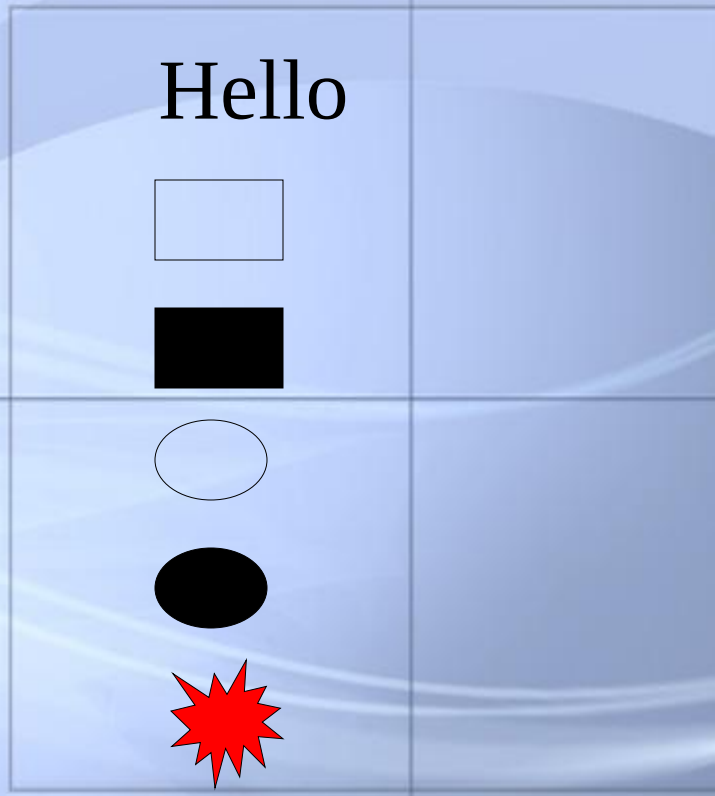
g.drawRect(x, y, width, height);

g.fillRect(x, y, width, height);

g.drawOval(x, y, width, height);

g.fillOval(x, y, width, height);

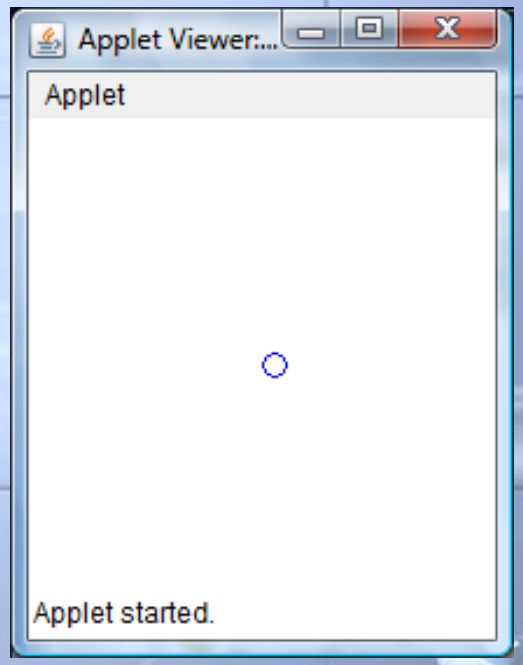
g.setColor(Color.red);



Use Graphics class to draw image

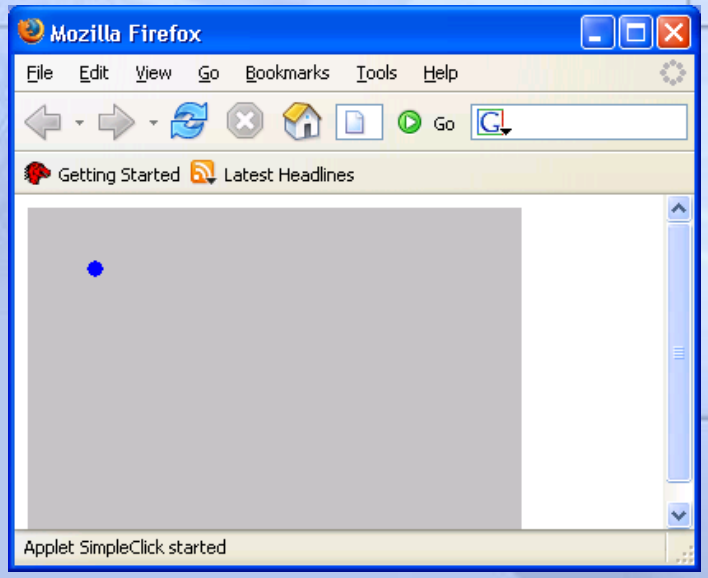
- Paint method draws image of Applet
  - Use graphics object to draw image
  - g parameter represents the applet image
  - Use commands on image to draw blue circle

```
public class SimpleClick extends Applet {
    public void paint(Graphics g) {
        g.setColor(Color.blue);
        g.drawOval(100, 100, 10, 10);
    }
}
```



Use Graphics class to draw image

```
public class SimpleClick extends Applet {
    public void paint(Graphics g) {
        g.setColor(Color.blue);
        g.fillOval(100, 100, 10, 10);
    }
}
```



repaint( )

- **Call repaint( ) when you have changed something and want your changes to show up on the screen**
  - You do *not* need to call repaint() when something in Java's own components (Buttons, TextFields, etc.)
  - You *do* need to call repaint() after drawing commands (drawRect(...), fillRect(...), drawString(...), etc.)
- **repaint( ) is a *request*--it might not happen**
- **When you call repaint( ), Java schedules a call to update(Graphics *g*)**

update( )

- When you call repaint( ), Java schedules a call to update(Graphics g)
- Here's what update does:
  - public void update(Graphics g) {
    // Fills applet with background color, then
    paint(g);
    }

Example of Using Graphics methods

```
import java.awt.*;
import java.applet.*;
import java.util.Random;
public class HelloWorld2 extends Applet
{    private static final int NUM_WORDS=100;
     private static final Color[] colors =
     {    Color.black,Color.red,Color.blue,Color.green,Color.yellow};
     private static Random randy;
     private int randomInRange(int low, int high)
     {    return (Math.abs(randy.nextInt()) % (high-low+1)) + low;
     }
     public void init()
     {    randy = new Random();
     }
```

can override init method to allocate & initialize (similar to a constructor)

```
public void paint(Graphics g)
    {    for (int i = 0; i < NUM_WORDS; i++) {
            int x = randomInRange(1, 200);
            int y = randomInRange(1, 200);
            g.setColor(colors[randomInRange(0,colors.length-1)]);
            g.drawString("Hello world!", x, y);
         }
    }
}
```

can change drawing color using setColor method

GUI Elements in Applet

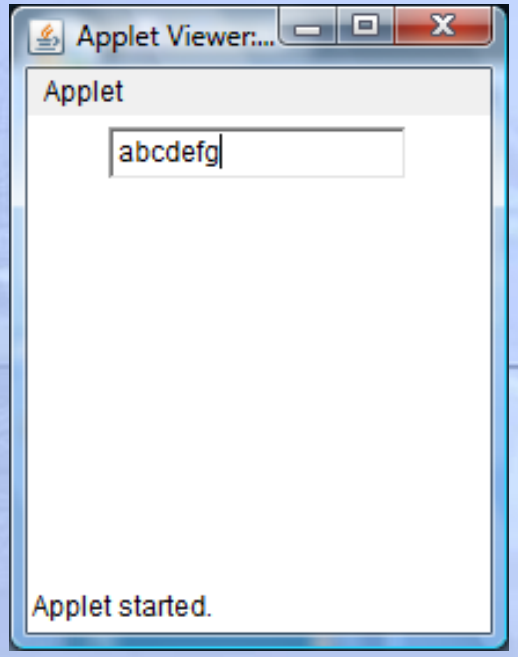
GUI elements in applets

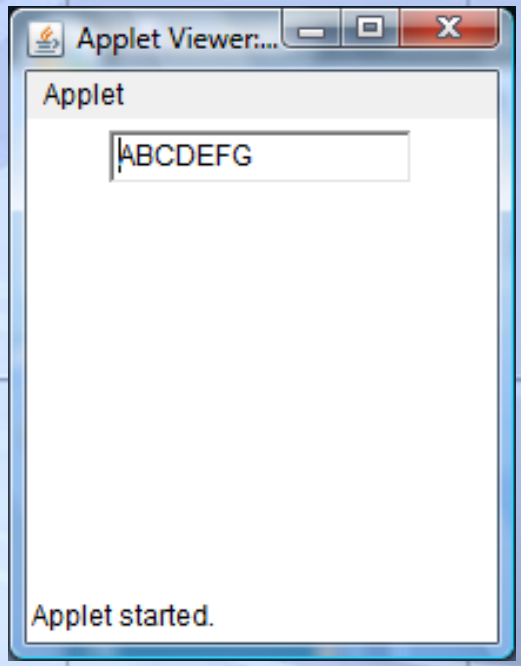
- Java has extensive library support for GUIs (Graphical User Interfaces)
  - has elements corresponding to HTML buttons, text boxes, text areas, …
- each element must be created and explicitly added to the applet.

```
Label nameLabel = new Label("User's name");
add(nameLabel);
TextField nameField = new TextField(20);
nameField.setValue("Dave Reed");
add(nameField);
```

- by default, GUI elements are placed in the order they were added, with elements moved to the next line as needed to fit.

Output of GUI Example with Applet





GUI Example with Applet

```
import java.awt.*;
import java.awt.event.*; import java.applet.*;
public class MyApplet extends Applet implements
ActionListener {
 TextField inputLine = new TextField(15);
public MyApplet()
 { add(inputLine);
   inputLine.addActionListener(this);
 }
public void actionPerformed(ActionEvent event)
 { String s = inputLine.getText();
   String sUp = s.toUpperCase();
   inputLine.setText(sUp);
 }
Public static void main(String args[])
 { MyApplet obj=new MyApplet();
 }
}
```

Interactive Applets

- Applets work in a graphical environment. Therefore, applets treats inputs as text strings.
- We need to create an area on the screen in which use can type and edit input items.
- We can do this using TextField class of the applet package.
- When data is entered, an event is generated. This can be used to refresh the applet output based on input values.

Interactive Applet Program..(cont)

```
import java.applet.Applet;
import java.awt.*;
public class SumNumsInteractive extends Applet {
    TextField text1, text2;
    public void init()
    {      text1 = new TextField(10);
           text2 = new TextField(10);
           text1.setText("0");
           text2.setText("0");
           add(text1);
           add(text2);
    }
```

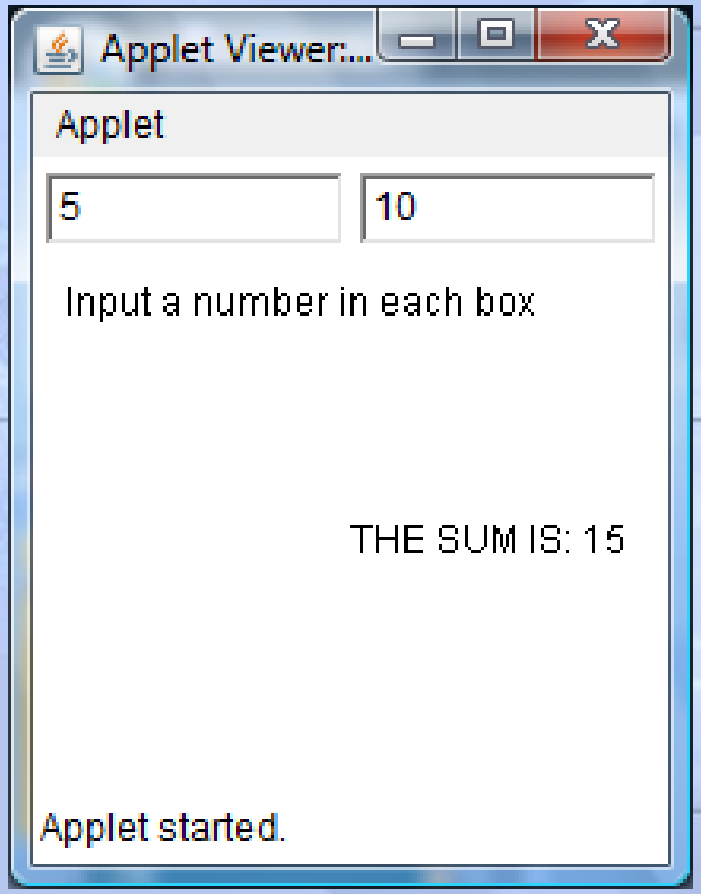
Interactive Applet Program..(cont)

```
public void paint(Graphics g) {
        int num1 = 0;
        int num2 = 0;
        int sum;
        String s1, s2, s3;
        g.drawString("Input a number in each box ", 10, 50);
        try {
                s1 = text1.getText();
                num1 = Integer.parseInt(s1);
                s2 = text2.getText();
                num2 = Integer.parseInt(s2);
        }
        catch(Exception e1)
        {}
```

Interactive Applet Program.

```
        sum = num1 + num2;
        String str = "THE SUM IS: "+String.valueOf(sum);
        g.drawString (str,100, 125);
    }
    public boolean action(Event ev, Object obj)
    {
        repaint();
        return true;
    }
}
```

Interactive Applet Execution



Summary

- An **applet** is a Java class
- Its code is downloaded from a web server
- It is run 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

Summary

- Applets were going to solve everything.
  - All the hype about applets got a lot of people talking about Java.
    - Applets are little Java programs that download on the fly as part of a web page
  - The good news
    - No manual deployment.
    - Very interactive U/I can be created.
    - Applets take advantage of all the processing power on the client computers that is idle with web apps.
  - The bad news
    - Applets took forever to download.
    - Applets took forever to launch once they did download.
    - Applets were slow and loaded with bugs and browser incompatibilities (Netscape was actually worse then IE).
    - The built-in widget toolkit (AWT) was so limited you couldn't create a decent user interface with it.