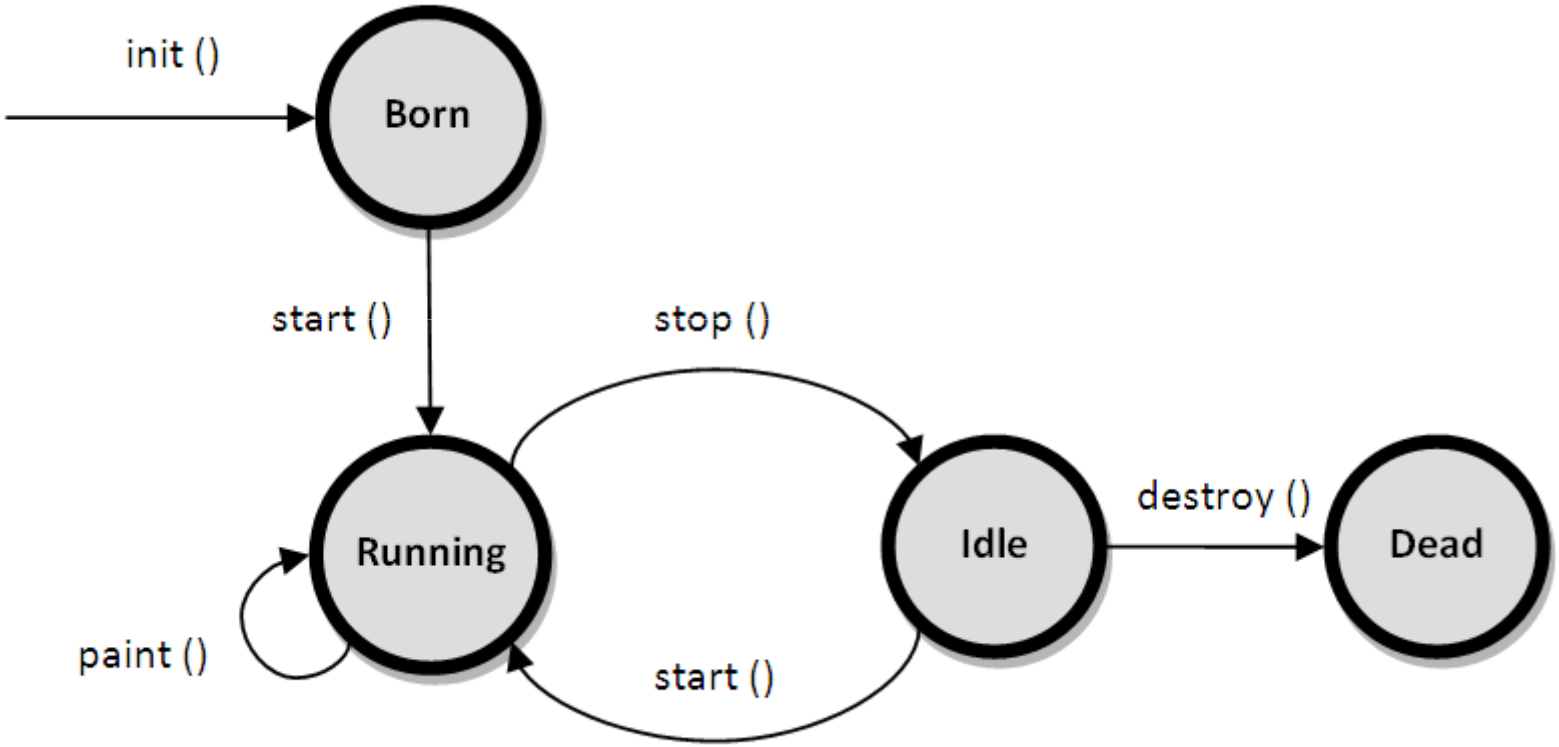
Java Applets

Applet Viewer and browsers

- An applet is a small Internet-based program written in Java to perform a specific task
- The applet is also able to run in HTML
- The applet is usually embedded in an HTML page on a Web site and can be executed from within a browser.
- Modern browsers come with applet capabilities.
- The JDK also comes with an applet viewer.
  - The JDK applet viewer is really just a minimum browser. It only understands the applet tag.

Life cycle of Applet



. . . Contd

- For creating any applet Applet class must be inherited. It provides 4 life cycle methods
- **public void init()** → To initializ the Applet. It is invoked only once.
- **public void start()** → Invoked after the init() method or browser is maximized. It is used to start the Applet.
- **public void stop()** → Used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
- **public void destroy()** → Used to destroy the Applet. It is invoked only once.

3.3 Simple Java Applet: Drawing a String

- Sample program
  - Create an applet to display
    `"Welcome to Java!!“`
- public void paint(Graphics g) → to paint the Applet
- It provides **Graphics class** object that can be used for drawing oval, rectangle, arc etc.

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



public class WelcomeApplet extends Applet {


public void init() {
 }


public void paint(Graphics g) {
    g.drawString("Welcome to Java Programming!",
25, 25 );
 }
}
```

`extends` allows us to inherit the capabilities of class `Applet`.

Method `paint` is guaranteed to be called in all applets. Its first line must be defined as above.

3.3 Simple Java Applet: Drawing a String

```
import java.awt.*;      // import package with class Graphics
import javax.applet.*;  // import class Applet
```

- Import predefined classes grouped into packages
  - When you create applets, `import` `Applet` class (in the package java.applet)
  - `import` the `Graphics` class (package `java.awt`) to draw graphics
    - Can draw lines, rectangles ovals, strings of characters
  - `import` specifies directory structure

3.3 Simple Java Applet: Drawing a String

```
<html>
<applet code = "WelcomeApplet.class" width = "300" height = "45">
</applet>
</html>
```

- Simple HTML file (`WelcomeApplet.html`)
  - Usually in same directory as **`.class`** file
  - Remember, `.class` file created after compilation
- HTML codes (tags)
  - Usually come in pairs
  - Begin with < and end with >
- Lines 1 and 4 - begin and end the HTML tags
- Line 2 - begins `<applet>` tag
  - Specifies code to use for applet
  - Specifies `width` and `height` of display area in pixels
- Line 3 - ends `<applet>` tag

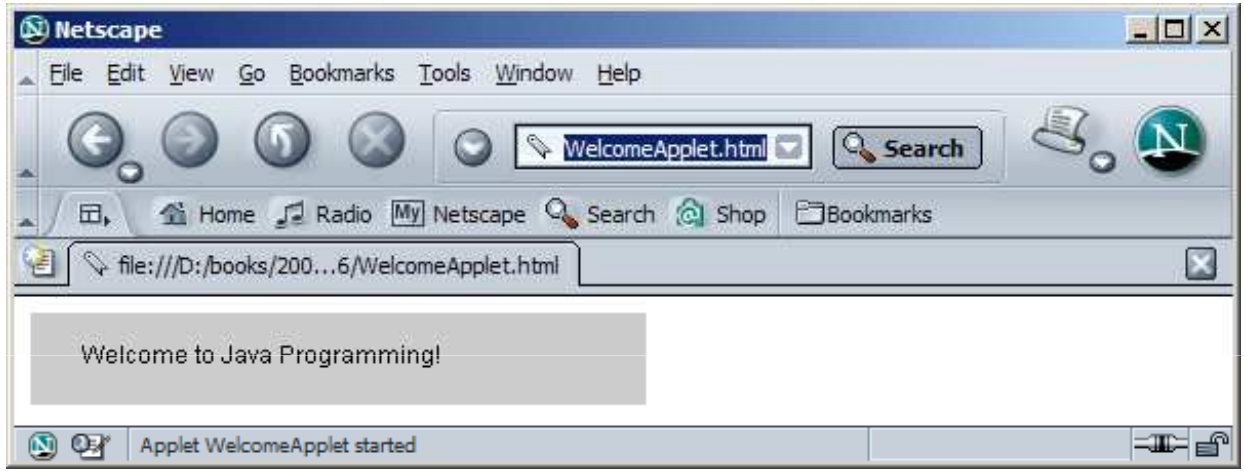
3.3 Simple Java Applet: Drawing a String

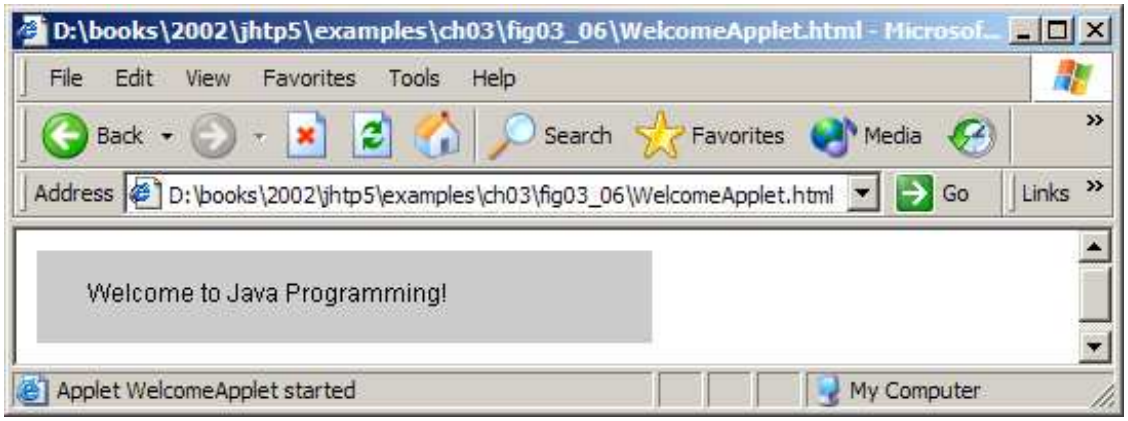
```
<html>
<applet code = "WelcomeApplet.class" width = "300" height = "45">
</applet>
</html>
```

- `appletviewer` only understands `<applet>` tags
  - Ignores everything else
  - Minimal browser
- Executing the applet
  - `appletviewer WelcomeApplet.html`
  - Perform in directory containing `.class` file

3.3 Simple Java Applet: Drawing a String

- Running the applet in a Web browser





3.4 Drawing Strings and Lines

- More applets
  - First example
    - Display two lines of text
    - Use `drawString` to simulate a new line with two `drawString` statements
  - Second example
    - Method `g.drawLine(x1, y1, x2, y2 )`
      - Draws a line from (`x1`, `y1`) to (`x2`, `y2`)
      - Remember that (`0`, `0`) is upper left
    - Use `drawLine` to draw a line beneath and above a string

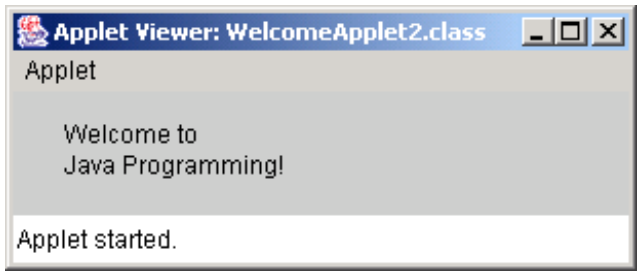
```
import java.awt.*;
import java.applet.*;


public class WelcomeApplet2 extends Applet {

        public void init() {
        }

        public void paint(Graphics g) {
                g.drawString( "Welcome to", 25, 25 );
                g.drawString( "Java Programming!", 25, 40 );
        }
}
```

The two `drawString` statements simulate a newline. In fact, the concept of lines of text does not exist when drawing strings.

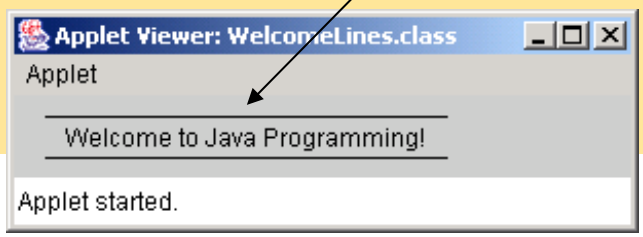


```
import java.awt.*;
import java.applet.*;


public class WelcomeApplet3 extends Applet {

          public void init() {
          }

          public void paint(Graphics g) {
                    g.drawString("Welcome to Java Programming!", 25, 25 );
                    g.drawLine (15, 10, 210, 10);
                    g.drawLine (15, 30, 210, 30);
          }
}
```



drawLine has coordinates for start and end point

3.4 Drawing Strings and Lines

- Method `drawLine` of class `Graphics`
  - Takes as arguments `Graphics` object and line's end points
  - X and y coordinate of first endpoint
  - X and y coordinate of second endpoint

Example - Draw an image

```
import java.awt.*;
import java.applet.*;
public class DisplayImage extends Applet
{
   Image picture;
   public void init()
 { picture = getImage(getDocumentBase(),"sonoo.jpg");    }
   public void paint(Graphics g)
   {  g.drawImage(picture, 30,30, this);    }
 }
```

Example – Bouncing ball

Step1: Draw a ball

Step 2: Position the ball at different place every time

Step 3: Paint each time at different position continuously

Step 4: Make thread to execute this

Step 5: Retrieve the height and width of the window

```
package exapplet;
import java.awt.*;
import java.applet.*;
public class Exbouncingball extends Applet
   implements Runnable{
   int x,y;
   int incx,incy;
  public void init()
  {  x=20;  y=20;  incx=1;  incy=1;    }
  public void start()
  {   Thread th=new Thread(this);
      th.start();  }
```

```
public void run()  {
      while(true)
      {
         repaint();
         try
         {        Thread.sleep(50);                    }
         catch(Exception e)
         {        }
```

```
int width  = getWidth();
int height = getHeight();
 if(x+40>width)     incx=-1;
 if(x<10)            incx=1;
 if(y+40>height)    incy=-1;
if(y<10)            incy=1;
x=x+incx;
y=y+incy;
 } }
 public void paint(Graphics g)
   {   g.drawOval(x,y,20,20);   }
}
```

Example – Display Form in Applet

```
// A Button to click
    Button okButton;
 // A textField to get text input
    TextField nameField;
// necessary to only allow one radio button to be
   selected at the same time.
    CheckboxGroup radioGroup;
 // The radio buttons to be selected
    Checkbox radio1;      Checkbox radio2;
 // An independant selection box
    Checkbox option;
```

In init() method

```
// Tell the applet not to use a layout manager.
        setLayout(null);
  // initialze the button and give it a text.
        okButton = new Button("A button");
  // text and length of the field
        nameField = new TextField("A
    TextField",100);
// initialize the radio buttons group
          radioGroup = new CheckboxGroup();
```

...contd

```
// group it belongs and sets the default state
   (unselected)
        radio1 = new Checkbox("Radio1",
   radioGroup,false);
  // same but selected
        radio2 = new Checkbox("Radio2",
   radioGroup,true);
  // Label and state of the checkbox
        option = new Checkbox("Option",false);
```

...contd

```
// now we will specify the positions of the GUI
   components.
  // this is done by specifying the x and y coordinate
   and
  //the width and height.
        okButton.setBounds(20,20,100,30);
        nameField.setBounds(20,70,100,40);
        radio1.setBounds(20,120,100,30);
        radio2.setBounds(140,120,100,30);
        option.setBounds(20,170,100,30);
```

...contd

```
// now that all is set we can add these components to
   the applet
     add(okButton);
     add(nameField);
     add(radio1);
     add(radio2);
     add(option);
```