Deen Dayal Upadhyaya College
Department of Computer Science
Workshop on Java J2SE

Dr. Rajni Bala
Associate Professor
Dept. of Computer Science
DDUC

Java Applets

Introduction to Java Applet Programs

- Applet is a small program
  - that resides on a web server .
  - Executes on client
  - Executed by the web browser
  - give web pages “dynamic content”
  - Does not have a main() method.

Java Applets

- Built using one of general definitions of applets
  - `Applet` class
  - `JApplet` class
- Java applets are usually graphical
  - Draw graphics in a defined screen area
  - Enable user interaction with GUI elements
  - Uses either AWT or Swings

Java Applet classes

- Package java.applet
  - Applet
  - AppletContext
  - AudioClip
  - AppletStub

Java Applets

- Applets are Java programs that can be embedded in HTML documents
  - To run an applet you must create a .html file which references the applet
- When browser loads Web page containing applet
  - Applet downloads into Web browser
  - begins execution
- Can be tested using appletviewer program

Life Cycle of an Applet

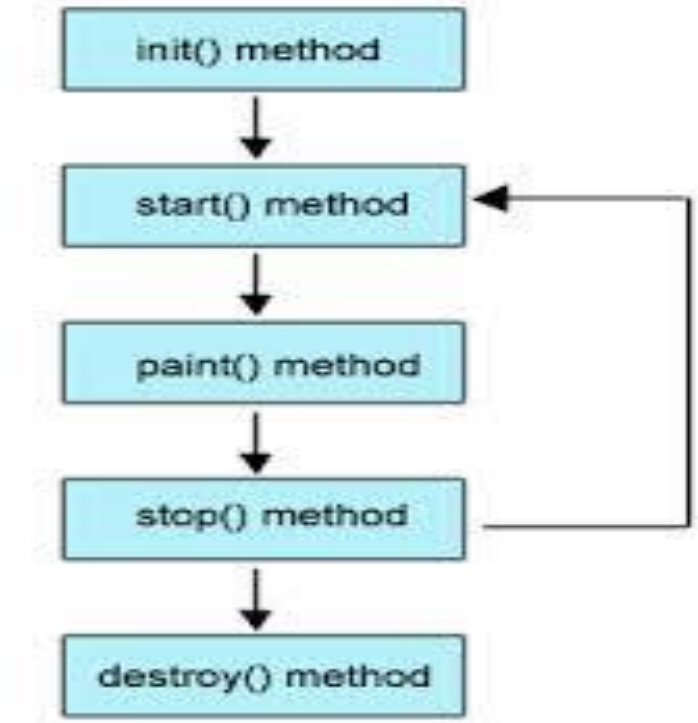
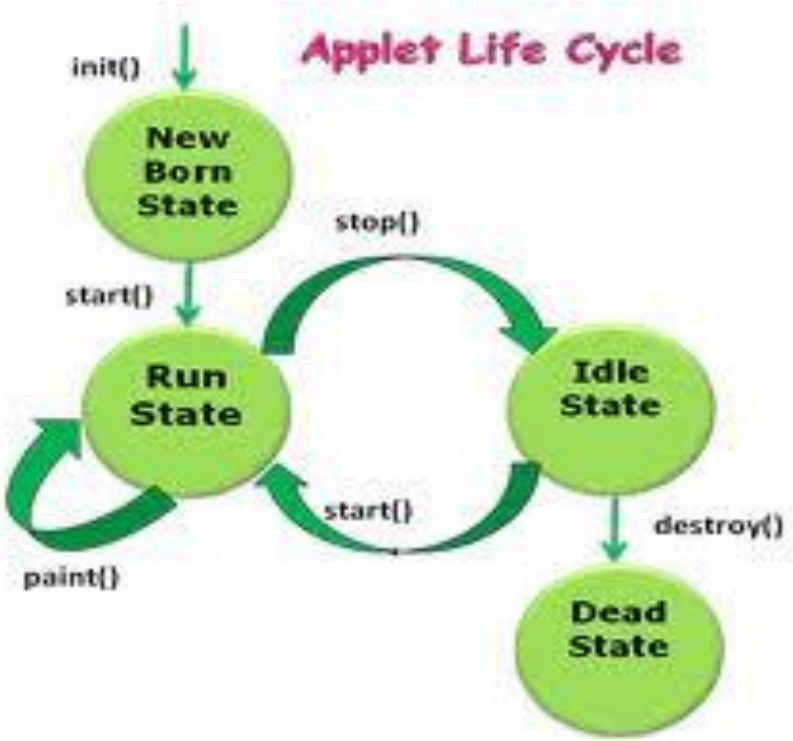


Figure: Life cycle of Applet

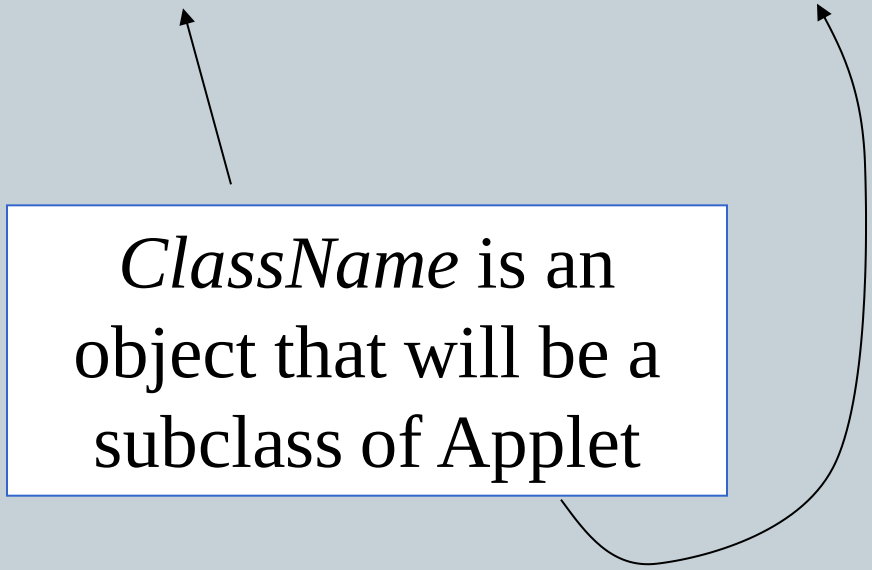
Life Cycle of an Applet



Applet Declaration

- Syntax (note difference from application declaration)

```
public class ClassName extends Applet
```



Sample Applet


```
// An Applet skeleton.
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/

public class AppletSkel extends Applet {
  // Called first.
  public void init() {
    // initialization
  }

  /* Called second, after init().  Also called whenever
     the applet is restarted. */
  public void start() {
    // start or resume execution
  }

  // Called when the applet is stopped.
  public void stop() {
    // suspends execution
  }

  /* Called when applet is terminated.  This is the last
     method executed. */
  public void destroy() {
    // perform shutdown activities
  }

  // Called when an applet's window must be restored.
  public void paint(Graphics g) {
    // redisplay contents of window
  }
}
```

First Applet


```
import java.awt.*;
import java.applet.*;
/*
<applet code="SimpleApplet" width=500 height=500>
</applet>
*/
public class SimpleApplet extends Applet {

    String msg;
    // set the foreground and background colors.
    public void init()
    {
        setBackground(Color.cyan);
        setForeground(Color.red);
        msg = "Inside init( ) --";
    }
    // Initialize the string to be displayed.
    public void start()
    {
        msg += " Inside start( ) --";
    }
    // Display msg in applet window.
    public void paint(Graphics g)
    {
        msg += " Inside paint( ).";
        g.drawString(msg, 10, 30);
    }
}
```

Running First Applet

- Save the file as SimpleApplet.java
- Click on Run
- In Submenu Click on Run as Applet
- Eclipse run applets using AppletViewer

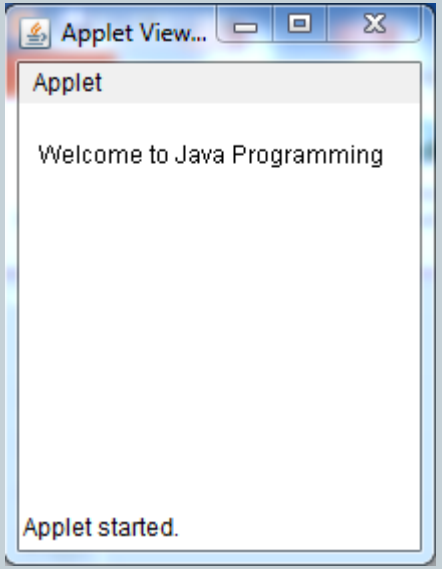
```
<APPLET CODE = "Whatever.class"
      WIDTH = nnn HEIGHT = mmmm>
  <\APPLET>
```

where **nnn** and **mmm** are specific pixel sizes

Applet to display a String


```
import java.awt.*;
import java.applet.*;
/*
<applet code="SimpleApplet" width=500 height=500>
</applet>
*/
public class SimpleApplet extends Applet {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    String msg;
    public void paint(Graphics g)
    {
        g.drawString("Welcome to Java Programming", 10, 30);
    }
}
```



- **Applet** provides all necessary support for applet execution, such as starting and stopping. It also provides methods that load and display images, and methods that load and play audio clips.
- **Applet** extends the AWT class **Panel**. In turn, **Panel** extends **Container**, which extends **Component**. These classes provide support for Java's window-based, graphical interface. Thus, **Applet** provides all of the necessary support for window-based activities.

More about Applets

- Applets are Window-based programs
- Applets are event-driven.

Applet

- An applet resembles a set of interrupt service routines.
- An applet waits until an event occurs. The AWT notifies the applet about an event by calling an event handler that has been provided by the applet.
- Once this happens, the applet must take appropriate action and then quickly return control to the AWT.
- For the most part, your applet should not enter a “mode” of operation in which it maintains control for an extended period. Instead, it must perform specific actions in response to events and then return control to the AWT run-time system.
- In those situations in which your applet needs to perform a repetitive task on its own (for example, displaying a scrolling message across its window), you must start an additional thread of execution.

Applet

- The user interacts with the applet as he or she wants, when he or she wants. These interactions are sent to the applet as events to which the applet must respond.

- For example, when the user clicks a mouse inside the applet's window,a mouse-clicked event is generated. If the user presses a key while the applet's window has input focus, a keypress event is generated. Applets can contain various controls, such as push buttons and check boxes. When the user interacts with one of these controls, an event is generated.

Applet that Scrolls msg right to left

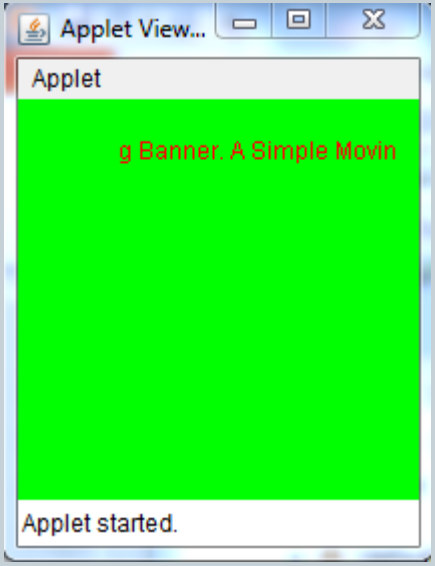

```
/* A simple banner applet.
This applet creates a thread that scrolls
the message contained in msg right to left
across the applet's window.
*/
import java.awt.*;
import java.applet.*;
/*
<applet code="BannerApplet" width=500 height=500>
</applet>
*/

public class BannerApplet extends Applet implements Runnable
{

    private static final long serialVersionUID = 1L;
    String msg = " A Simple Moving Banner.";
    Thread t = null;
    int state;
    boolean stopFlag;
    // Set colors and initialize thread.
    public void init()
    {
            setBackground(Color.green);
            setForeground(Color.red);
    }
    // Start thread
    public void start()
    {
            t = new Thread(this);
            stopFlag = false;
            t.start();
```

```
// Entry point for the thread that runs the banner.
    public void run()
    {
        char ch;
        // Display banner
        for( ; ; )
        {
            try
            {
                repaint();
                Thread.sleep(250);
                ch = msg.charAt(0);
                msg = msg.substring(1, msg.length());
                msg += ch;
                if(stopFlag)
                        break;
            }
            catch(InterruptedException e) {}
        }
    }
    // Pause the banner.
    public void stop()
    {
        stopFlag = true;
        t = null;
    }
    // Display the banner.
    public void paint(Graphics g)
    {
        g.drawString(msg, 50, 30);
    }
}
```

Result



Exercise

Create a Simple Calculator

Some other functions used

- Repaint(int maxdelay)
- Repaint(int maxdelay, int x,int y)
- showStatus

HTML Applet tags

< APPLET
[CODEBASE = *codebaseURL*]
CODE = *appletFile*
[ALT = *alternateText*]
[NAME = *appletInstanceName*]
WIDTH = *pixels* HEIGHT = *pixels*
[ALIGN = *alignment*]
[VSPACE = *pixels*] [HSPACE = *pixels*]
>
[< PARAM NAME = *AttributeName* VALUE = *AttributeValue*>]
[< PARAM NAME = *AttributeName2* VALUE = *AttributeValue*>]
. . .
[*HTML Displayed in the absence of Java*]
</APPLET>

Using parameters in Applets


```
import java.awt.*;
import java.applet.*;
/*
<APPLET CODE="ParamApplet.class" WIDTH="400" HEIGHT="50">
<PARAM NAME="fontName"    VALUE="Courier">
<PARAM NAME="fontsize"    VALUE="24">
<PARAM NAME="leading"  VALUE="2">
</APPLET>
*/

public class ParamApplet extends Applet
{
        private static final long serialVersionUID = 1L;
        String fontName;
        int fontSize;
        float leading;
        boolean active;
        // Initialize the string to be displayed.
        public void start()
        {
            String param;
            fontName = getParameter("fontName");
            if(fontName == null)
                fontName = "Not Found";
            param = getParameter("fontSize");
            try {
            if(param != null) // if not found
                fontSize = Integer.parseInt(param);
            else
                fontSize = 0;
            }
            catch(NumberFormatException e) {
                fontSize = -1;
            }
            param = getParameter("leading");
            try
            {
            if(param != null) // if not found
                leading = Float.valueOf(param).floatValue();
            else
                leading = 0;
            }
            catch(NumberFormatException e) {
                leading = -1;
            }
            param = getParameter("accountEnabled");
            if(param != null)
                active = Boolean.valueOf(param).booleanValue();
        }
        // Display parameters.
        public void paint(Graphics g)
        {
            g.drawString("Font name: " + fontName, 0, 10);
            g.drawString("Font size: " + fontSize, 0, 26);
            g.drawString("Leading: " + leading, 0, 42);
            g.drawString("Account Active: " + active, 0, 58);
        }
}
```

getDocumentBase( ) and getCodeBase( )


```
import java.awt.*;
import java.applet.*;
import java.net.*;
/*
<applet code="Bases" width=300 height=50>
</applet>
*/

public class Bases extends Applet{
  // Display code and document bases.
  public void paint(Graphics g) {
    String msg;

    URL url = getCodeBase(); // get code base
    msg = "Code base: " + url.toString();
    g.drawString(msg, 10, 20);

    url = getDocumentBase(); // get document base
    msg = "Document base: " + url.toString();
    g.drawString(msg, 10, 40);
  }
}
```

Handling Images


```
import java.applet.*;
import java.awt.*;
import java.io.IOException;
import java.net.*;

import javax.imageio.ImageIO;
/*
<applet code="ImageApplet.class" width="300" height="200">
<param name="image" value="ankita.jpg">
</applet>
*/
public class ImageApplet extends Applet
{
  private static final long serialVersionUID = 1L;
  private Image image;
  private AppletContext context;
  String imageURL;
  public void init()
  {
      context = this.getAppletContext();
      imageURL = this.getParameter("image");
      if(imageURL == null)
      {
         imageURL = "java.jpg";
      }
      try
      {
         URL url = new URL(getCodeBase(),imageURL);
         //image = context.getImage(url);
        image=ImageIO.read(url);
      }
      catch(MalformedURLException e)
      {
         e.printStackTrace();
         // Display in browser status bar
         context.showStatus("Could not load image!");
      } catch (IOException e) {
        // TODO Auto-generated catch block
            context.showStatus("Could not load image!");
        e.printStackTrace();
    }
   }
   public void paint(Graphics g)
   {
      context.showStatus("Displaying image");
      g.drawImage(image, 20, 20, 300, 200, null);
      g.drawString(imageURL, 10, 10);
   }
}
```

More functions

Method	Description
void destroy( )	Called by the browser just before an applet is terminated. Your applet will override this method if it needs to perform any cleanup prior to its destruction.
AccessibleContext getAccessibleContext( )	Returns the accessibilty context for the invoking object.
AppletContext getAppletContext( )	Returns the context associated with the applet.
String getAppletInfo( )	Returns a string that describes the applet.
AudioClip getAudioClip(URL *url*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url*.
AudioClip getAudioClip(URL *url*, String *clipName*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url* and having the name specified by *clipName*.
URL getCodeBase( )	Returns the URL associated with the invoking applet.
URL getDocumentBase( )	Returns the URL of the HTML document that invokes the applet.

Table 19-1. *The Methods Defined by* Applet

Some more functions….

Method	Description
Image getImage(URL *url*)	Returns an **Image** object that encapsulates the image found at the location specified by *url*.
Image getImage(URL *url*, String *imageName*)	Returns an **Image** object that encapsulates the image found at the location specified by *url* and having the name specified by *imageName*.
Locale getLocale( )	Returns a **Locale** object that is used by various locale-sensitive classes and methods.
String getParameter(String *paramName*)	Returns the parameter associated with *paramName*. **null** is returned if the specified parameter is not found.
String[ ] [ ] getParameterInfo( )	Returns a **String** table that describes the parameters recognized by the applet. Each entry in the table must consist of three strings that contain the name of the parameter, a description of its type and/or range, and an explanation of its purpose.
void init( )	Called when an applet begins execution. It is the first method called for any applet.
boolean isActive( )	Returns **true** if the applet has been started. It returns **false** if the applet has been stopped.
static final AudioClip newAudioClip(URL *url*)	Returns an **AudioClip** object that encapsulates the audio clip found at the location specified by *url*. This method is similar to **getAudioClip( )** except that it is static and can be executed without the need for an **Applet** object. (Added by Java 2)

Table 19-1. *The Methods Defined by* Applet (continued)

Method	Description
void play(URL *url*)	If an audio clip is found at the location specified by *url*, the clip is played.
void play(URL *url*, String *clipName*)	If an audio clip is found at the location specified by *url* with the name specified by *clipName*, the clip is played.
void resize(Dimension *dim*)	Resizes the applet according to the dimensions specified by *dim*. **Dimension** is a class stored inside **java.awt**. It contains two integer fields: **width** and **height**.
void resize(int *width*, int *height*)	Resizes the applet according to the dimensions specified by *width* and *height*.
final void setStub(AppletStub *stubObj*)	Makes *stubObj* the stub for the applet. This method is used by the run-time system and is not usually called by your applet. A *stub* is a small piece of code that provides the linkage between your applet and the browser.
void showStatus(String *str*)	Displays *str* in the status window of the browser or applet viewer. If the browser does not support a status window, then no action takes place.
void start( )	Called by the browser when an applet should start (or resume) execution. It is automatically called after **init( )** when an applet first begins.
void stop( )	Called by the browser to suspend execution of the applet. Once stopped, an applet is restarted when the browser calls **start( )**.

Table 19-1. *The Methods Defined by* Applet (continued)