Chapter 2 (Applet)

What is an Applet

Applets are small Java programs that are primarily used in Internet computing. They can be transported over the Internet from one computer to another and run using the Applet Viewer or any Web browser that supports Java. Applet, like any application program, can do many things for us.

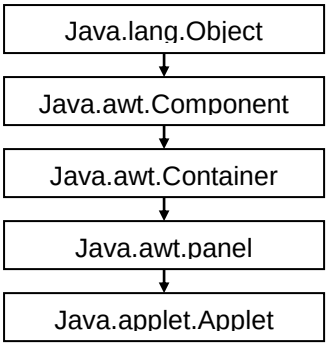

Fig. The Class Hierarchy of Applet

How Applets Differs from Applications

Although both the applets and stand-alone applications are Java programs, there are significant differences between them. Applets are not full-featured application programs. They are usually written to accomplish a small task or a component of task. Since they are usually designed for use on the Internet, they impose certain limitations and restriction in their design.

- Applets do not use the main ( ) method for initiating the execution of the code. Applets, when loaded, automatically call certain methods of Applet class to start and execute the applet code.
- Unlike stand-alone applications, applets cannot be run independently. They are run from inside a Web page using a special feature known as HTML tabs.
- *Other points are given below in Security Restrictions*

What Applets Can and Can't Do

Security Restrictions

Every browser implements security policies to keep applets from compromising system security. This section describes the security policies that current browsers adhere to. However, the implementation of the security policies differs from browser to browser. Also, security policies are subject to change. For example, if a browser is developed for use only in trusted environments, then its security policies will likely be much more lax than those described here.

Current browsers impose following restrictions on any applet that is loaded over the network:

- An applet cannot load libraries or define native methods.
- It cannot ordinarily read or write files on the host that's executing it.
- It cannot make network connections except to the host that it came from.
- It cannot start any program on the host that's executing it.
- It cannot read certain system properties.
- Windows that an applet brings up look different than windows that an application brings jp.

Each browser has a SecurityManager object that implements its security policies. When a SecurityManager detects a violation, it throws a SecurityException. Your applet can catch this SecurityException and react appropriately.

Applet Capabilities

The **java.applet** package provides an API that gives applets some capabilities that applications don't have. For example, applets can play sounds, which other programs can't do yet.

Here are some other things that current browser and other applet viewers let applets do:

- Applets can usually make network connections to the host they came from.
- Applets running within a Web browser can easily cause HTML documents to be displayed.
- Applets can invoke public methods of other applets on the same page.
- Applets that are loaded from the local file system (from a directory in the user's CLASSPATH) have none of the restrictions that applets loaded over the network do.
- Although most applets stop running once you leave their page, they don't have to.

Life Cycle of the Applet

Loading the Applet

When an applet is loaded, here's what happens:

- An instance of the applet's controlling class (an Applet subclass) is created.
- The applet *initializes* itself.
- The applet *starts* running.

Leaving and Returning to the Applet's Page

When the user leaves the page -- for example, to go to another page -- the applet has the option of *stopping* itself. When the user returns to the page, the applet can *start* itself again. The same sequence occurs when the user iconifies and then reopens the window that contains the applet. (Other terms used instead of *iconify* are *minaturize*, *minimize*, and *close*.)

Reloading the Applet

Some browsers let the user reload applets, which consists of unloading the applet and then loading it again. Before an applet is unloaded, it's given the chance to *stop* itself and then to perform a *final cleanup*, so that the applet can release any resources it holds. After that, the applet is unloaded and then loaded again, as described in **Loading the Applet**, above.

Methods for Milestones

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

The Simple applet, like every other applet, features a subclass of the Applet class. The Simple class overrides four Applet methods so that it can respond to major events:

init

To *initialize* the applet each time it's loaded (or reloaded).

start

To *start* the applet's execution, such as when the applet's loaded or when the user revisits a page that contains the applet.

stop

To *stop* the applet's execution, such as when the user leaves the applet's page or quits the browser.

destroy

To perform a *final cleanup* in preparation for unloading.

Important Note on init and Constructor

- The init method is useful for one-time initialization that doesn't take very long.
- In general, the init method should contain the code that you would normally put into a constructor.
- The reason ***applets shouldn't usually have constructors*** is that an applet isn't guaranteed to have a full environment until its init method is called.
- For example, the Applet image loading methods simply don't work inside of an applet constructor.
- The init method, on the other hand, is a great place to call the image loading methods, since the methods return quickly.

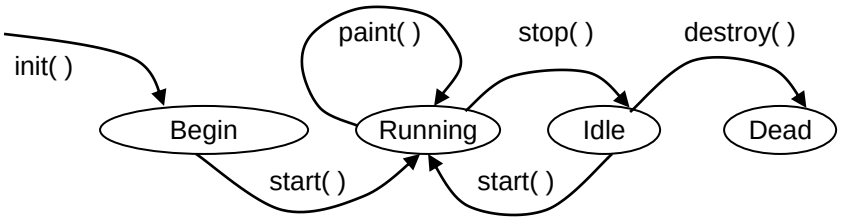

Fig: Life Cycle of Applet

Applet Tag format in HTML

```
<Applet
        [codebase=codebaseURL]
        code=Applet_file
        [Alt=alternate text]
        [Name=applet_Instance_name]
        width=pixel
        Height=pixel
        [align=alignment]
        [vspace=pixel]
        [hspace=pixel]
>
        [<param Name=Attributename value=attributevalue>]
        [<param Name=Attributename value=attributevalue>]
        ...
</Applet>
```

Examples

Sample Hello world Applet program (Example 1)

```
/*
Author: Shiba R. Tamrakar
Program: "hello world"
*/

import java.applet.*;
import java.awt.*; // required to paint on screen


public class HelloWorld extends Applet // Extending Applet class
{

    public void init() //automatically calls init() class
    {
        // It is required but does not need anything.
    }


// This method gets called when the applet is terminated
// That's when the user goes to another page or exits the browser.
    public void stop()
    {
        // no actions needed here now.
    }


// The standard method that you have to use to paint things on screen
// This overrides the empty Applet method, you can't called it "display" for example.

    public void paint(Graphics g)
    {
         //method to draw text on screen
        // String first, then x and y coordinate.
        g.drawString("Hey hey hey",20,20);
        g.drawString("Hellooow World",20,40);
    }
}
```

Sample Applet Program to sum two given number.

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

public class  Sum extends JApplet{
        JButton add = new JButton("Add");
        JTextField one = new JTextField(5);
        JTextField two = new JTextField(5);
        JTextField ans = new JTextField(5);
        JLabel plus = new JLabel("+");
```

```
        JLabel equalsto = new JLabel("=");
        JPanel jp = new JPanel();
        public void init(){
                jp.add(one);
                jp.add(plus);
                jp.add(two);
                jp.add(equalsto);
                jp.add(ans);
                add.addActionListener(new ActionListener(){public void
                actionPerformed(ActionEvent ae){
                        int x = Integer.parseInt(one.getText());
                        int y = Integer.parseInt(two.getText());
                        ans.setText(String.valueOf(x+y));
                        }
                });
                jp.add(add);
                getContentPane().add(jp);
        }
        public void start(){
        }
}
```

Sample Example to draw shapes with colors.

```
import java.awt.*;
import java.applet.*;
public class DrawExample extends Applet
{
        Font bigFont;  //font variable
        Color redColor;
        Color weirdColor;
        Color bgColor;
        public void init()
        {
                bigFont = new Font("Arial",Font.BOLD,16);
                weirdColor = new Color(60,60,122);
                bgColor = Color.blue;
        }
        public void stop()
        {

        }
        public void paint(Graphics g)
        {
                g.setFont(bigFont);
                g.drawString("Shapes and Colors",80,20);
                g.setColor(redColor);
                g.drawRect(100,100,100,100);
                g.fillRect(110,110,80,80);
                g.setColor(weirdColor);
                /* a circle (int x, int y, int width, int height,int
                startAngle, int arcAngle);*/
                g.fillArc(120,120,60,60,0,360);
                g.setColor(Color.yellow);
                // Draw a line (int x1, int y1, int x2, int y2)
                g.drawLine(140,140,160,160);
```

```
            // reset the color to the standard color for the next time
            the applets paints
            g.setColor(Color.black);
        }
}
```

Sample program to draw line

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

public class DrawingLines extends Applet {

  int width, height;

  public void init() {
    width = getSize().width;
    height = getSize().height;
    setBackground( Color.black );
  }

  public void paint( Graphics g ) {
    g.setColor( Color.green );
    for ( int i = 0; i < 10; ++i ) {
      g.drawLine( width, height, i * width / 10, 0 );
    }
  }
}
```

Sample program Drawing different shapes objects

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

public class DrawingStuff extends Applet {

  int width, height;

  public void init() {
    width = getSize().width;
    height = getSize().height;
    setBackground( Color.black );
  }

  public void paint( Graphics g ) {

    // As we learned in the last lesson,
    // the origin (0,0) is at the upper left corner.
    // x increases to the right, and y increases downward.

    g.setColor( Color.red );
    g.drawRect( 10, 20, 100, 15 );
    g.setColor( Color.pink );
    g.fillRect( 240, 160, 40, 110 );
```

```
        g.setColor( Color.blue );
        g.drawOval( 50, 225, 100, 50 );
        g.setColor( Color.orange );
        g.fillOval( 225, 37, 50, 25 );

        g.setColor( Color.yellow );
        g.drawArc( 10, 110, 80, 80, 90, 180 );
        g.setColor( Color.cyan );
        g.fillArc( 140, 40, 120, 120, 90, 45 );

        g.setColor( Color.magenta );
        g.fillArc( 150, 150, 100, 100, 90, 90 );
        g.setColor( Color.black );
        g.fillArc( 160, 160, 80, 80, 90, 90 );

        g.setColor( Color.green );
        g.drawString( "Groovy!", 50, 150 );
      }
    }
```

Sample program to Trapping mouse movement

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

public class Mouse1 extends Applet
  implements MouseListener, MouseMotionListener {

  int width, height;
  int mx, my;  // the mouse coordinates
  boolean isButtonPressed = false;

  public void init() {
    width = getSize().width;
    height = getSize().height;
    setBackground( Color.black );

    mx = width/2;
    my = height/2;

    addMouseListener( this );
    addMouseMotionListener( this );
  }

  public void mouseEntered( MouseEvent e ) {
    // called when the pointer enters the applet's rectangular area
  }
  public void mouseExited( MouseEvent e ) {
    // called when the pointer leaves the applet's rectangular area
  }
  public void mouseClicked( MouseEvent e ) {
    // called after a press and release of a mouse button
    // with no motion in between
    // (If the user presses, drags, and then releases, there will be
    // no click event generated.)
```

```
  }
  public void mousePressed( MouseEvent e ) {  // called after a button is pressed
down
    isButtonPressed = true;
    setBackground( Color.gray );
    repaint();
    // "Consume" the event so it won't be processed in the
    // default manner by the source which generated it.
    e.consume();
  }
  public void mouseReleased( MouseEvent e ) {  // called after a button is released
    isButtonPressed = false;
    setBackground( Color.black );
    repaint();
    e.consume();
  }
  public void mouseMoved( MouseEvent e ) {  // called during motion when no
buttons are down
    mx = e.getX();
    my = e.getY();
    showStatus( "Mouse at (" + mx + "," + my + ")" );
    repaint();
    e.consume();
  }
  public void mouseDragged( MouseEvent e ) {  // called during motion with buttons
down
    mx = e.getX();
    my = e.getY();
    showStatus( "Mouse at (" + mx + "," + my + ")" );
    repaint();
    e.consume();
  }

  public void paint( Graphics g ) {
    if ( isButtonPressed ) {
      g.setColor( Color.black );
    }
    else {
      g.setColor( Color.gray );
    }
    g.fillRect( mx-20, my-20, 40, 40 );
  }
}
```

Java Applet program which will get input of semester and 5 different subjects from web browser and display it in applet.

Java File

```
import java.awt.*;
import java.applet.*;
public class Question4 extends Applet
{
        String Semester,Sub1, Sub2, Sub3, Sub4, Sub5;
        public void init()
        {
                Semester=getParameter("Semester");
                Sub1=getParameter("Sub1");
```

```
            Sub2=getParameter("Sub2");
            Sub3=getParameter("Sub3");
            Sub4=getParameter("Sub4");
            Sub5=getParameter("Sub5");
      }
      public void paint(Graphics G)
      {
            G.drawString(Semester,10,30);
            G.drawString(Sub1,10,50);
            G.drawString(Sub2,10,70);
            G.drawString(Sub3,10,90);
            G.drawString(Sub4,10,110);
            G.drawString(Sub5,10,130);
      }
}
```

HTML File

```
<html>
      <head>
            <title>Question 4</title>
      </head>
<body>
      <applet code="Question4" width="250" height="150">
            <param name= Semester value=5>
            <param name= Sub1 value="Advance Programming. ">
            <param name= Sub2 value="Computer Graphics.">
            <param name= Sub3 value="Object Oriented Analysis and
Design.">
            <param name= Sub4 value="Software Engineering.">
            <param name= Sub5 value="Web Technology.">
      </applet>
</body>
</head>
</html>
```