Applet in Java

An **applet** is a Java program that runs in a Web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

There are some important differences between an applet and a standalone Java application, including the following –

- An applet is a Java class that extends the java.applet.Applet class.
- A main() method is not invoked on an applet, and an applet class will not define main().
- Applets are designed to be embedded within an HTML page.
- When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
- A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
- The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.
- Applets have strict security rules that are enforced by the Web browser. The security of an applet is often referred to as sandbox security, comparing the applet to a child playing in a sandbox with various rules that must be followed.
- Other classes that the applet needs can be downloaded in a single Java Archive (JAR) file.

Life Cycle of an Applet

Four methods in the Applet class gives you the framework on which you build any serious applet

- **init** – This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
- **start** – This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages.
- **stop** – This method is automatically called when the user moves off the page on which the applet sits. It can, therefore, be called repeatedly in the same applet.
- **destroy** – This method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you should not normally leave resources behind after a user leaves the page that contains the applet.
- **paint** – Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. The paint() method is actually inherited from the java.awt.

A "Hello, World" Applet

Following is a simple applet named HelloWorldApplet.java –

```
import java.applet.*;
import java.awt.*;
public class HelloWorldApplet extends Applet {
  public void paint (Graphics g) {
    g.drawString ("Hello World", 25, 50);   } }
```

These import statements bring the classes into the scope of our applet class –

- java.applet.Applet
- java.awt.Graphics

Without those import statements, the Java compiler would not recognize the classes Applet and Graphics, which the applet class refers to.

The Applet Class

Every applet is an extension of the *java.applet.Applet class*. The base Applet class provides methods that a derived Applet class may call to obtain information and services from the browser context.

These include methods that do the following –

- Get applet parameters
- Get the network location of the HTML file that contains the applet

- Get the network location of the applet class directory
- Print a status message in the browser
- Fetch an image
- Fetch an audio clip
- Play an audio clip
- Resize the applet

Additionally, the Applet class provides an interface by which the viewer or browser obtains information about the applet and controls the applet's execution. The viewer may –

- Request information about the author, version, and copyright of the applet
- Request a description of the parameters the applet recognizes
- Initialize the applet
- Destroy the applet
- Start the applet's execution
- Stop the applet's execution

The Applet class provides default implementations of each of these methods. Those implementations may be overridden as necessary.

The "Hello, World" applet is complete as it stands. The only method overridden is the paint method.

Invoking an Applet

An applet may be invoked by embedding directives in an HTML file and viewing the file through an applet viewer or Java-enabled browser.

The <applet> tag is the basis for embedding an applet in an HTML file. Following is an example that invokes the "Hello, World" applet –

```
<html>
  <title>The Hello, World Applet</title>
  <hr>
  <applet code = "HelloWorldApplet.class" width = "320" height = "120">
    If your browser was Java-enabled, a "Hello, World"
    message would appear here.
  </applet>
  <hr>
</html>
```

Note – You can refer to HTML Applet Tag to understand more about calling applet from HTML.

The code attribute of the <applet> tag is required. It specifies the Applet class to run. Width and height are also required to specify the initial size of the panel in which an applet runs. The applet directive must be closed with an </applet> tag.

If an applet takes parameters, values may be passed for the parameters by adding <param> tags between <applet> and </applet>. The browser ignores text and other tags between the applet tags.

Non-Java-enabled browsers do not process <applet> and </applet>. Therefore, anything that appears between the tags, not related to the applet, is visible in non-Java-enabled browsers.

The viewer or browser looks for the compiled Java code at the location of the document. To specify otherwise, use the codebase attribute of the <applet> tag as shown –

```
<applet codebase = "https://amrood.com/applets" code = "HelloWorldApplet.class"
  width = "320" height = "120">
```

If an applet resides in a package other than the default, the holding package must be specified in the code attribute using the period character (.) to separate package/class components. For example –

```
<applet  = "mypackage.subpackage.TestApplet.class"
  width = "320" height = "120">
```

Getting Applet Parameters

The following example demonstrates how to make an applet respond to setup parameters specified in the document. This applet displays a checkerboard pattern of black and a second color.

The second color and the size of each square may be specified as parameters to the applet within the document. CheckerApplet gets its parameters in the init() method. It may also get its parameters in the paint() method. However, getting the values and saving the settings once at the start of the applet, instead of at every refresh, is convenient and efficient.

The applet viewer or browser calls the init() method of each applet it runs. The viewer calls init() once, immediately after loading the applet. (Applet.init() is implemented to do nothing.) Override the default implementation to insert custom initialization code.

The Applet.getParameter() method fetches a parameter given the parameter's name (the value of a parameter is always a string). If the value is numeric or other non-character data, the string must be parsed.

The following is a skeleton of CheckerApplet.java –

```
import java.applet.*;
import java.awt.*;
public class CheckerApplet extends Applet {
  int squareSize = 50;   // initialized to default size
  public void init() {}
  private void parseSquareSize (String param) {}
  private Color parseColor (String param) {}
  public void paint (Graphics g) {} }
```

Here are CheckerApplet's init() and private parseSquareSize() methods –

```
public void init () {
  String squareSizeParam = getParameter ("squareSize");
  parseSquareSize (squareSizeParam);
  String colorParam = getParameter ("color");
  Color fg = parseColor (colorParam);
   setBackground (Color.black);
  setForeground (fg);}
private void parseSquareSize (String param) {
  if (param == null) return;
  try {
    squareSize = Integer.parseInt (param);
  } catch (Exception e) {
    // Let default value remain
  }}
```

The applet calls parseSquareSize() to parse the squareSize parameter. parseSquareSize() calls the library method Integer.parseInt(), which parses a string and returns an integer. Integer.parseInt() throws an exception whenever its argument is invalid.

Therefore, parseSquareSize() catches exceptions, rather than allowing the applet to fail on bad input.

The applet calls parseColor() to parse the color parameter into a Color value. parseColor() does a series of string comparisons to match the parameter value to the name of a predefined color. You need to implement these methods to make this applet work.

Specifying Applet Parameters

The following is an example of an HTML file with a CheckerApplet embedded in it. The HTML file specifies both parameters to the applet by means of the <param> tag.

```
<html>
  <title>Checkerboard Applet</title>
  <hr>
  <applet code = "CheckerApplet.class" width = "480" height = "320">
    <param name = "color" value = "blue">
    <param name = "squaresize" value = "30">
  </applet>
  </hr>
</html>
```

Note − Parameter names are not case sensitive.

Application Conversion to Applets

It is easy to convert a graphical Java application (that is, an application that uses the AWT and that you can start with the Java program launcher) into an applet that you can embed in a web page.

Following are the specific steps for converting an application to an applet.

- Make an HTML page with the appropriate tag to load the applet code.
- Supply a subclass of the JApplet class. Make this class public. Otherwise, the applet cannot be loaded.
- Eliminate the main method in the application. Do not construct a frame window for the application. Your application will be displayed inside the browser.
- Move any initialization code from the frame window constructor to the init method of the applet. You don't need to explicitly construct the applet object. The browser instantiates it for you and calls the init method.
- Remove the call to setSize; for applets, sizing is done with the width and height parameters in the HTML file.
- Remove the call to setDefaultCloseOperation. An applet cannot be closed; it terminates when the browser exits.
- If the application calls setTitle, eliminate the call to the method. Applets cannot have title bars. (You can, of course, title the web page itself, using the HTML title tag.)
- Don't call setVisible(true). The applet is displayed automatically.

Event Handling

Applets inherit a group of event-handling methods from the Container class. The Container class defines several methods, such as processKeyEvent and processMouseEvent, for handling particular types of events, and then one catch-all method called processEvent.

In order to react to an event, an applet must override the appropriate event-specific method.

```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;
import java.applet.Applet;
import java.awt.Graphics;
public class ExampleEventHandling extends Applet implements MouseListener {
  StringBuffer strBuffer;
  public void init() {      addMouseListener(this);    strBuffer = new StringBuffer();
    addItem("initializing the apple ");   }
  public void start() {      addItem("starting the applet ");   }
  public void stop() {      addItem("stopping the applet ");   }
  public void destroy() { addItem("unloading the applet");   }
  void addItem(String word) {      System.out.println(word);    strBuffer.append(word);      repaint();   }
  public void paint(Graphics g) {
    // Draw a Rectangle around the applet's display area.
    g.drawRect(0, 0, getWidth() - 1, getHeight() - 1);
    // display the string inside the rectangle.
    g.drawString(strBuffer.toString(), 10, 20);   }
  public void mouseEntered(MouseEvent event) {   }
  public void mouseExited(MouseEvent event) {   }
public void mousePressed(MouseEvent event) {    }
public void mouseReleased(MouseEvent event) {    }
public void mouseClicked(MouseEvent event) {   addItem("mouse clicked! ");   }}
```

Now, let us call this applet as follows –

```
<html>
  <title>Event Handling</title>
  <hr>
  <applet code = "ExampleEventHandling.class"  width = "300" height = "300">
  </applet>
  <hr>
</html>
```

Initially, the applet will display "initializing the applet. Starting the applet." Then once you click inside the rectangle, "mouse clicked" will be displayed as well.

Displaying Images

An applet can display images of the format GIF, JPEG, BMP, and others. To display an image within the applet, you use the drawImage() method found in the java.awt.Graphics class.

Following is an example illustrating all the steps to show images –

```
import java.applet.*;
import java.awt.*;
import java.net.*;
public class ImageDemo extends Applet {
  private Image image;
  private AppletContext context;
  public void init() {
    context = this.getAppletContext();
    String imageURL = this.getParameter("image");
    if(imageURL == null) {  imageURL = "java.jpg";      }
    try {
      URL url = new URL(this.getDocumentBase(), imageURL);
      image = context.getImage(url);
    } catch (MalformedURLException e) {
      e.printStackTrace();        // Display in browser status bar
      context.showStatus("Could not load image!");      }  }
  public void paint(Graphics g) {
    context.showStatus("Displaying image");
    g.drawImage(image, 0, 0, 200, 84, null);
    g.drawString("www.javalicense.com", 35, 100);    }  }
```

Now, let us call this applet as follows –

```
<html>
  <title>The ImageDemo applet</title>
  <hr>
  <applet code = "ImageDemo.class" width = "300" height = "200">
    <param name = "image" value = "java.jpg">
  </applet>
  <hr>
</html>
```

Playing Audio

An applet can play an audio file represented by the AudioClip interface in the java.applet package. The AudioClip interface has three methods, including –

- **public void play()** – Plays the audio clip one time, from the beginning.
- **public void loop()** – Causes the audio clip to replay continually.
- **public void stop()** – Stops playing the audio clip.

To obtain an AudioClip object, you must invoke the getAudioClip() method of the Applet class. The getAudioClip() method returns immediately, whether or not the URL resolves to an actual audio file. The audio file is not downloaded until an attempt is made to play the audio clip.
Following is an example illustrating all the steps to play an audio –

```
import java.applet.*;
import java.awt.*;
import java.net.*;
public class AudioDemo extends Applet {
  private AudioClip clip;
  private AppletContext context;
  public void init() {
    context = this.getAppletContext();
    String audioURL = this.getParameter("audio");
    if(audioURL == null) {    audioURL = "default.au";    }
    try {
      URL url = new URL(this.getDocumentBase(), audioURL);
      clip = context.getAudioClip(url);     }
catch (MalformedURLException e) {
      e.printStackTrace();
      context.showStatus("Could not load audio file!");     }
  }
  public void start() {
    if(clip != null) {  clip.loop();     }  }
    public void stop() {   if(clip != null) {
      clip.stop();   }
  } }
```

Now, let us call this applet as follows –

```
<html>
  <title>The ImageDemo applet</title>
  <hr>
  <applet code = "ImageDemo.class" width = "0" height = "0">
    <param name = "audio" value = "test.wav">
  </applet>
  <hr>
</html>
```

AWT Tutorial

JAVA provides a rich set of libraries to create Graphical User Interface in platform independent way. In this article we'll look in AWT (Abstract Window Toolkit).
This tutorial is designed for Software Professionals who are willing to learn JAVA GUI Programming in simple and easy steps. This tutorial will give you great understanding on JAVA GUI Programming concepts and after completing this tutorial you will be at intermediate level of expertise from where you can take yourself at higher level of expertise.

Prerequisites

Before proceeding with this tutorial you should have a basic understanding of Java programming language, text editor and execution of programs etc.