CHAPTER

2

THE APPLET CLASS

Chapter Outlines

After comprehensive study of this chapter, you will be able to:

- To discuss and program applets by using Applet and JApplet Class
- To understand basics of applet tag and its attributes
- To pass parameters to applet and use them

2.1 Applet Basics

An applet is a Java program, which can be downloaded from a remote server and executed inside the web browser of the local machine to generate the dynamic content. Unlike standalone java programs, an applet class will not define main() method. 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. While creating applets, class must be public because class must be public because its object is created by Java Plug-in software that resides on the browser. All applets import the java.awt package. The following are two issues with Java applets:

- **Security:** Java resolves the security issue by restricting applets to Java's execution environment and preventing access to system resources.
- **Portability:** Portability is defined as the applet's ability to run on different computers and operating systems.

Advantage of Applet

There are many advantages of applet. They are as follows:

- It works at client side so less response time.
- Secured
- It can be executed by browsers running under many platforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

- Plug-in is required at client browser to execute applet.
- Applets cannot read or write local files.
- Applets cannot execute local programs.
- Applets cannot connect to any machine on the network except the machine from which they are downloaded.

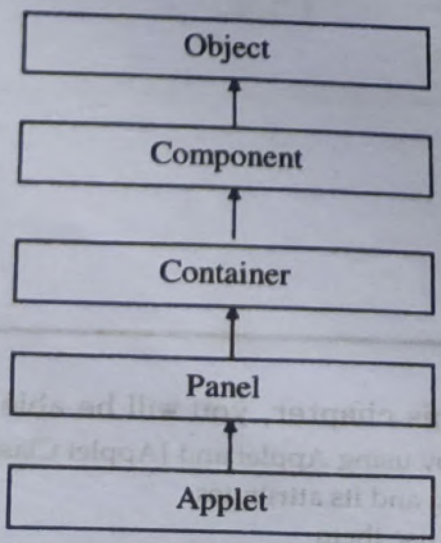


Figure 2.1: Hierarchy of Java Applets

2.2 THE APPLET TAG

HTML provides an <applet>...</applet> container tag for embedding Java applet, with the following attributes:

- Code="*JavaClassName*": Specifies the applet class name, with the ".class" extension.
- Codebase=URL: (optional) Specifies the URL of the directory in which applet code is to be found.
- Width, height: Specify the width and height of the applet's display area inside the browser.
- Alt: (optional) alternate text if the applet fails to be displayed.
- Archive: (optional) specifies the JAR file that contains the applet classes. If your applet involves more than one classes, it is common and more efficient to jar (i.e., zip) them together into a single JAR file for distribution.
- Align: specifies alignment of the applet
- Hspace, Vspace: Hspace specifies the space allocated to the right and left of the applet and Vspace specifies the space allocated above and below the applet.
- Name: Specifies name of the applet

Skeleton of applet tag looks like below:

```
<Applet
    [code=filename]
    [codebase=URL]
    [alt=alternate_text]
    [name=name of applet]
    width= pixel
        height=pixel
        [align=alignment]
        [hspace=pixel]
        [vspace=pixel]
>
[<Param name=.....  Value=.....>]
.
.
.
[<Param name=.....  Value=.....>]
</Applet>
```

There are two ways to run an applet

1. By html file.
2. By appletViewer tool (for testing purpose).

2.2.1 Applet by HTML File

To execute the applet by html file, create an applet and compile it. After that create an html file and place the applet tag in html file. Now double click the html file to execute the applet.

```
//First.java  File
        import java.applet.Applet;
        import java.awt.Graphics;
        public class First extends Applet
        {
                public void paint(Graphics g)
                {
                        g.drawString("welcome",150,150);
                }
        }
//First.html  File
        <html>
        <body>
        <applet code="First.class" width="300" height="300">
        </applet>
        </body>
        </html>
```

2.2.2 Applet by Appletviewer Tool

To execute the applet by appletviewer tool, create an applet that contains applet tag in comment and compile it. After that run it by using command: *appletviewer First.java*. Now Html file is not required but it is for testing purpose only.

```
        //First.java
        import java.applet.Applet;
        import java.awt.Graphics;
        public class First extends Applet
        {
                public void paint(Graphics g)
                {
                        g.drawString("welcome to applet",150,150);
                }
}
/*
<applet code="First.class" width="300" height="300">
</applet>
*/
```

2.3 Two Types of Applets

Depending upon the base class used by the programmer for creating applets, it can be divided into two categories:

- Applets Derived from Applet Class
- Applets Derived from JApplet Class

Applets Derived from Applet Class

These applets use the Abstract Window Toolkit (AWT) to provide the graphical user interface (or use no GUI at all). This style of applet has been available since Java was first created. Examples presented previously in this chapter falls under applets of this category. Applet class is defined in package *java.awt .applet*. To create GUI by using Applet class, we must write all GUI related codes inside *paint()* method of the Applet class.

Applets Derived from JApplet Class

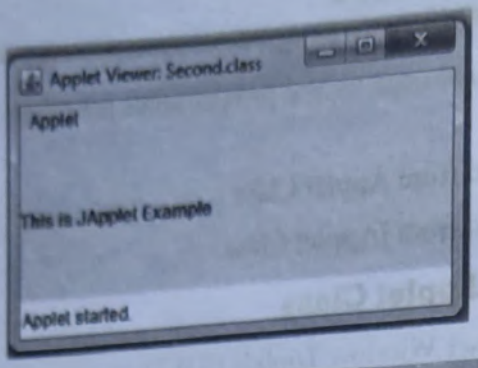
These types of applets are those based on the Swing class **JApplet**. Swing applets use the Swing classes to provide the GUI. Swing offers a richer and often easier-to-use user interface than does the AWT. Thus, Swing-based applets are now the most popular. However, traditional AWT-based applets are still used, especially when only a very simple user interface is required. Thus, both AWT- and Swing-based applets are valid.

Because **JApplet** inherits **Applet**, all the features of **Applet** are also available in **JApplet**. However, JApplets have a lot of extra structure that plain Applets don't have. Because of this structure, the painting of a JApplet is a more complex affair and is handled by the system. When we make a subclass of JApplet we should not write a paint() method for it. If we want to draw on a JApplet, we should add a component to the applet to be used for that purpose. On the other hand, we can and generally should write an init() method for a subclass of JApplet.

Example

```
//Second.java
import javax.swing.*;
public class Second extends JApplet
{
   public void init()
   {
      JLabel lbl=new JLabel("This is JApplet Example");
      this.add(lbl);
   }
}
/*
<applet code="Second.class" width="300" height="300">
</applet>
*/
```

Output



2.4 An Applet Skeleton (Life Cycle of an Applet)

There are five primary methods in the Applet class which provide the framework for just about every applet. An applet can be in any one of the following states during its lifecycle.

- **Newborn State**: Applet enters into this state when **init()** method is invoked. This method is used to initialize any components of your applet before the program begins to run. The init method is called after the param tags have been processed by the JVM. Overriding init() method is optional while creating applet. Like main method of standalone java programs, in case of applets init method is always executed first.

  Syntax of init method

  public void init()

- **Running State**: Applet enters into running state when start() method is invoked. The start method is automatically called once the browser calls the init method. This method is also called automatically whenever a user returns to the page containing the applet. Paint method is called immediately after start, the paint method is what displays your applet on a webpage. Paint is also called any time the applet needs to repaint itself. Overriding start() method is also optional while creating applet. But, overriding paint() method is mandatory while creating applets.

  Syntax of start method

  public void start()

  Syntax of paint method

  public void paint(Graphics)

- **Idle State**: Applet enters into idle state when stop() method is invoked. Stop is automatically called by the browser when the user moves off the page where the applet is embedded. It can be called multiple times within the same applet. Overriding stop() method is again optional while creating applet.

  Syntax of stop method

  public void stop()

- **End State**: Applet enters into end state when destroy() method is called. Once the browser shuts down, the destroy method is called automatically and removes any

resources that would otherwise be left behind by the applet. Overriding destroy() method is also optional while creating applet.

Syntax of destroy method

Public void destroy()

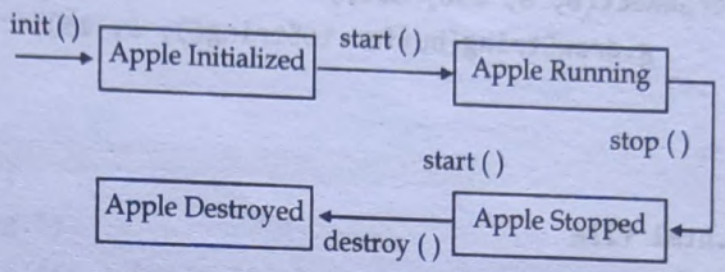
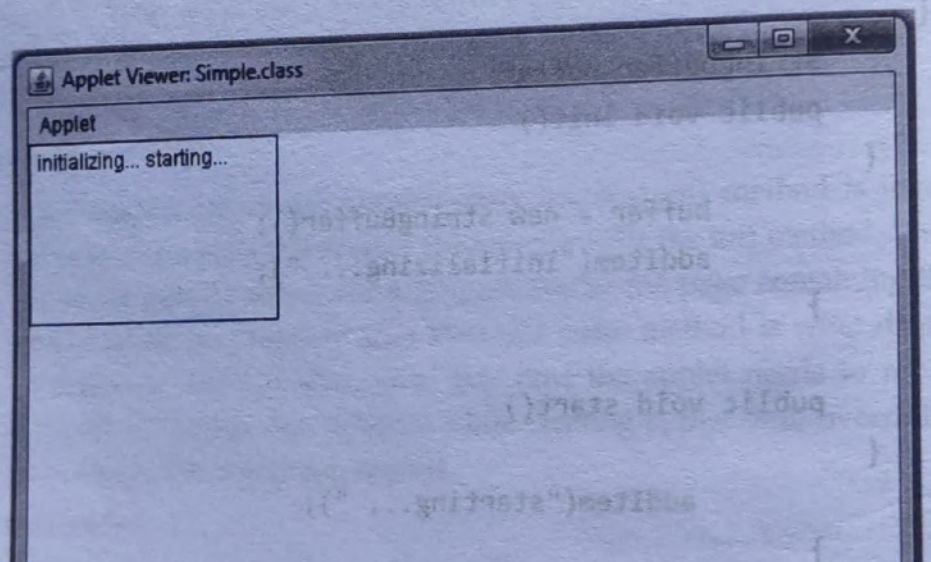


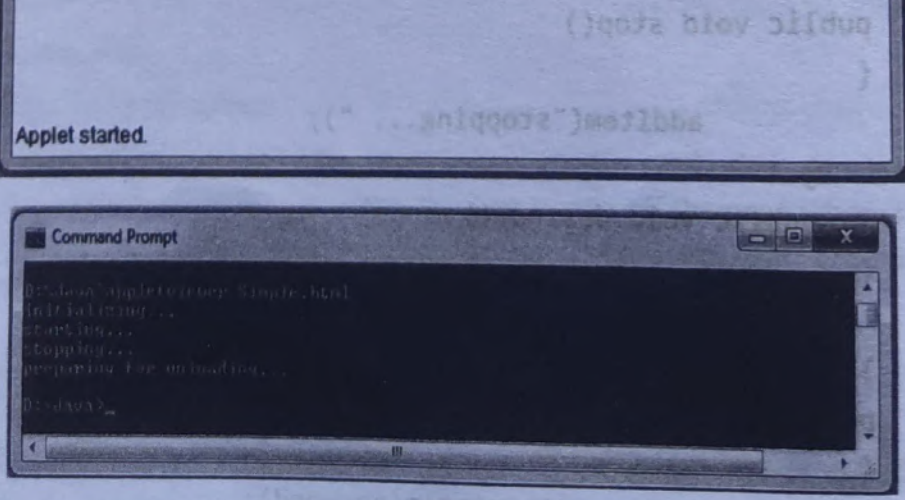
Figure 2.2: Applet Life Cycle

```
//Example with complete skeleton
//Simple.java File
import java.applet.Applet;
import java.awt.Graphics;
public class Simple extends Applet
{
    StringBuffer buffer;
    public void init()
  {
        buffer = new StringBuffer();
        addItem("initializing... ");
    }

    public void start()
  {
         addItem("starting... ");
    }
  public void stop()
  {
        addItem("stopping... ");
    }
    public void destroy()
  {
        addItem("preparing for unloading...");
  }
  private void addItem(String newWord)
  {
        System.out.println(newWord);
        buffer.append(newWord);
```

```
            repaint();
        }
        public void paint(Graphics g)
    {
        g.drawRect(0, 0, 150, 100);
            g.drawString(buffer.toString(), 5, 15);
        }
}

//Simple.html file
      <html>
      <body>
      <applet code="Simple.class" width="500" height="300">
      </applet>
      </body>
      </html>
```

Output





Note: Swing-based applets or JApplets uses different mechanism to accomplish this task.

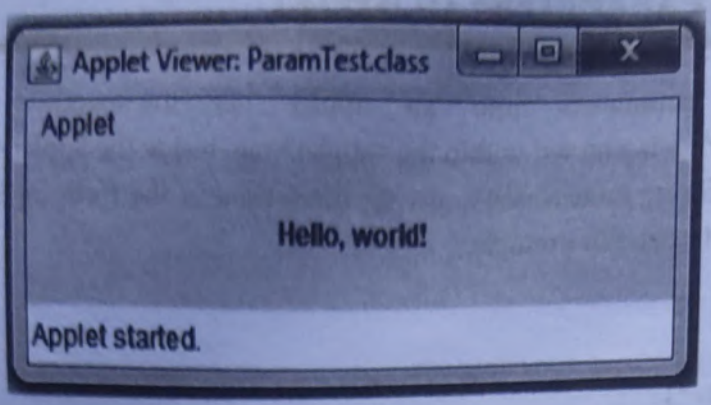
2.5 PASSING PARAMETERS TO APPLET

We can pass parameters into an applet via the <param name="*paramName*" value="*paramValue*"/> tag nested within the <applet> tag. Inside the applet source code, we can use getParameter(String *paramName*) to get the *paramValue* in the form og String; or null if the parameter does not exist. For example,

Example

```
//.java File
import java.awt.*;
import javax.swing.*;
public class ParamTest extends JApplet
{
   String msg;
   public void init ()
   {
      msg = getParameter("message");
         if (msg == null) msg = "Hi";
         createGUI();
         }
   private void createGUI()
   {
         Container cp = getContentPane();
         cp.setBackground(new Color(204, 238, 241));
         cp.add(new JLabel(msg, JLabel.CENTER));
   }
}
//.html File
<html>
<head>
<title>A Swing Applet with Parameter</title>
</head>
<body>
  <h3>A Swing Applet with Parameter</h3>
  <applet code="ParamTest.class" width="300" height="60"
      alt="Error Loading Applet!">
    <param name="message" value="Hello, world!" />
  </applet>
</body>
</html>
```

Output



EXERCISE

1. How applet is different from desktop applications? Explain its advantages and drawbacks.
2. Write an applet that simply displays information like name, location, ESTD, Services Offered etc of XYZ hospital.
3. Discuss applet tag briefly. What are different ways to run an applet? Explain briefly with example.
4. What are different states of an java applet? Explain applet life cycle with suitable state diagram.
5. What are two different types of applets? Explain each in brief with suitable example.
6. How can we pass parameters to applet from HTML files? Explain with proper example.
7. Write an applet that reads to numbers from HTML file as parameter and displays the sum of theses numbers in applet.

❑❑❑