Applet Programming

Introduction

- Applet are small java program primarily used for Internet computing.
- They can run using **Applet Viewer** or any web browser that supports java.
- An applet can perform arithmetic operation, display graphics, play sounds, accept user input, create animation and play interactive games.
- Local applet (store on local computer)
- Remote applet (downloaded from remote computer and run locally)(The URL must be specified in the **CODEBASE** value in html tag.)

Applet v/s Application Program

Applet are not full featured application program. They are usually written to a accomplish a small task or a component of a task. Since they are usually designed for use on the internet, they impose certain limitation and restrictions in their design.

1. 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.
2. Unlike stand-alone applications, applets can not be run independently. They have to be embedded inside a web page to get using a special feature known as HTML tag.
3. Applets can only be executed inside the web browser of appletviewer.

4. Applets can not read from or write to the files on the local computer.
5. Applet can not communicate with the other servers on the network.
6. Applet can not run any program from the local computer.
7. Applet are restricted from using libraries from other languages such as C or C++.(java language supports this feature through **native** methods)

- Applet actually are of two types.
    1. One which use the AWT (abstract window toolkit) to draw and display graphics.(Use Applet class)
    2. Another type which uses Swing for drawing and display of graphics.(Use JApplet class). The swing offers a richer and easer to use user interface then AWT. So Swing based applet are more popular then AWT based.
- But AWT based applet are still used, especially when only a very simple user interface is required.
- The JApplet class inherits the Applet class, so all the methods of Applet class are available in the Japplet class.

Writing Applet

- Building an applet code (.java file)
- Creating an executable applet (.class file)
- Designing a web page using HTML tags
- Writing an applet tag <APPLET>
- Adding <APPLET> tag into the web page
- Creating HTML file
- Testing the applet code

Building applet code

```
import java.awt.*;
Import java.applet.*;
…
Public class abc extends Applet
{
…..
…..
   public void paint(Graphics g)
    {
     …..
     …..              //Applet operation code
     …..
   }
…..
…..
}
```

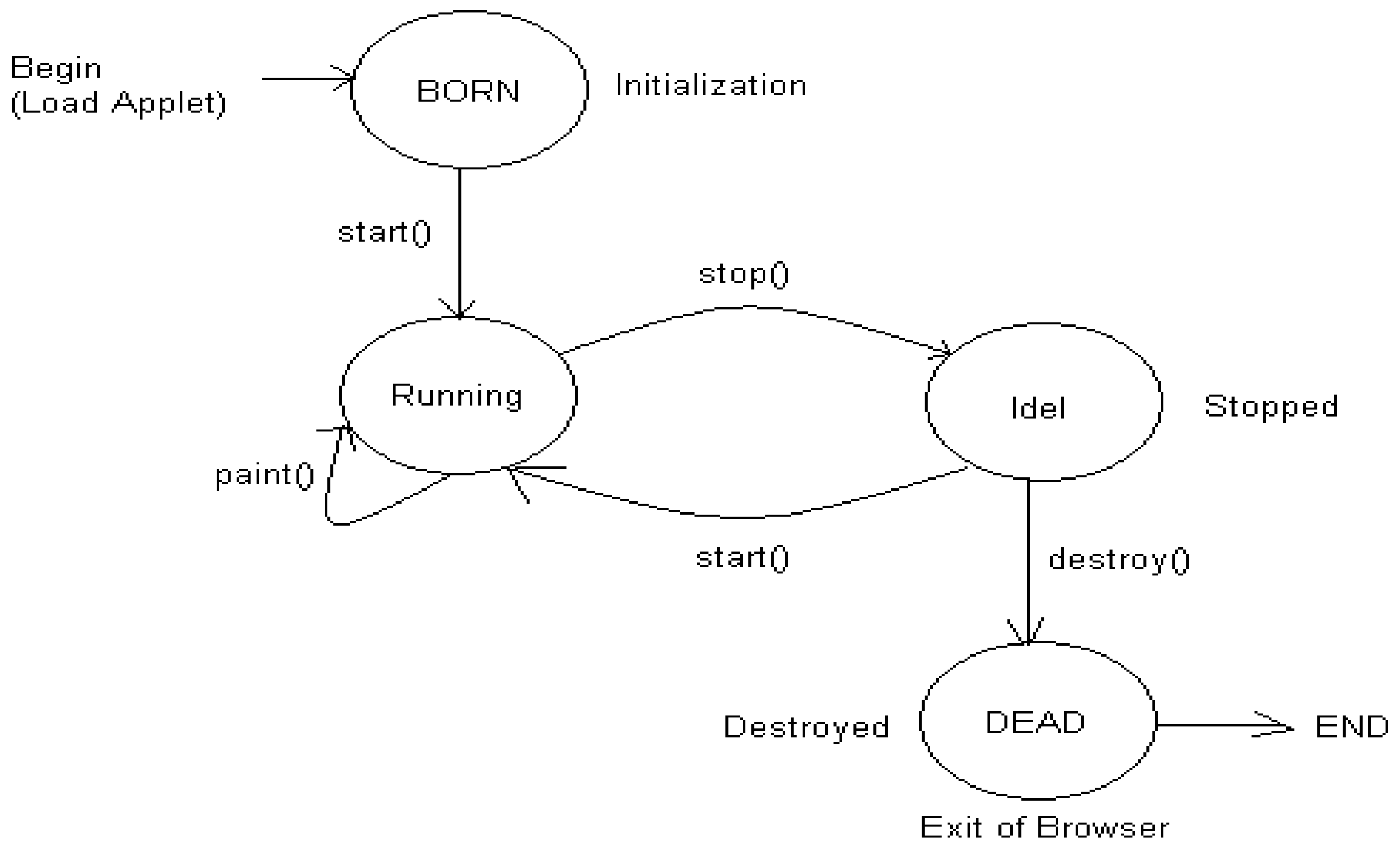
```
import java.awt.*;
import java.applet.*;…
public class abc extends Applet{
    public void paint(Graphics g){
    g.drawString(“Hello Java”,10,100);
  }
}
```

When executed , draw the string **Hello Java** at position 10,100(pixel value).

- Chain of classes inherited by Applet class

$$\begin{tikzcd}
\text{java.lang.Object} \arrow[d] \\
\text{java.awt.Component} \arrow[d] \\
\text{java.awt.container} \arrow[d] \\
\text{java.awt.Panel} \arrow[d] \\
\text{java.applet.Applet}
\end{tikzcd}$$

Applet life cycle



Applet states

- **Initialization state:** Applet enters the initialization state when it is first loaded. It calls init method of Applet class. The initialization occurs only once in the applet's life cycle. We can override the init method as

  public void init(){------------}

- **Running state:** Applet enters the running state when the system calls the start method of Applet class from born state or Idle state. The start method is automatically called when we return back to the web page after leaving it. We can override the start method as

  public void start(){---------}

- **Stopped /Idel state:** An applet become idle when it is stopped from running. The stop method is automatically called when we leave the web page. We can override the stop method as

  public void stop(){----------}

- **Dead State**: An applet is said to be dead when it is removed from memory. This occurs automatically by invoking the destroy() method when we quit the browser. The destroying stage occurs only once(same as init()). We can override the destroy() method to clean up these resources:

    public void destroy(){------------}

- **Display state**: Applet moves to display state whenever it has to perform some output operation on the screen. This happen immediately after the applet enters into the running state. The paint() method is used for this. We must overriding this method if we want anything to be displayed on screen.

    public void paint(Graphics g){-------------}

  The display state is not actually the part of applet's life cycle. The paint method is inherited from the **Component class**, a super class of Applet.

Example of applet life cycle

```
import java.awt.*;
import java.applet.*;
/*
<applet code=abc.class height=200 width=200>
  </applet>
*/
public class abc extends Applet{
String text;
public void init()
{
setBackground(Color.green);
setForeground(Color.red);
Text=“this is the example applet”;

}
public void start()
{
System.out.println("…………..Starting the applet");
}
public void stop()
{
System.out.println("…………..Stopping the applet");
}
public void destroy()
{
System.out.println("…………..exiting the applet");
}
public void paint(Graphics g){
   g.drawString("Hello Java",30,30);
showStatus("this is status bar");
  }
}
```

Creating and executing Applet

- Create an html file with <Applet > tag(abc.html)
- Create a java code for applet(Hellojava.java)
- Compile the code and get class file (Hellojava.class)
- Run the html file(By web browser or by appletviewer)

```
<html>
<!My first java applet>
     <head>  <title>   Welcome to java applet    </title>
     </head>
 <body>
   <center>
     <h1> Welcome to the world of java applet</h1>
   </center>
   <br>
   <center>
     <applet
          code=Hellojava.class
          width=400
         height=200>
     </applet>
   </center>  </body>  </html>
```

```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet{
    public void paint(Graphics g)
    {
      g.drawString("Hello Java",10,100);
    }
}
```

The class Hellojava is a public so it must be saved by Hellojava.java file name.

The applet contain only one executable statement g.drawString(---);

Which draw the string when executed at position 10,100(pixels) of the applets reserved space.

- To execute the applet we put the html file and the Hellojava class file in same directory and then run the html file with any java supported web browser.
- We can also run the applet by using appletviewer tool which is supplied with JDK by running appletviewer abc.html
- One more method is to put <applet> tag in java source file in comments. Then compile the code. Now use appletviewer sourcefile.java. This will start the applet.

```
import java.awt.*;
import java.applet.*;
/*
<applet code=abc.class
   height=200
   width=200
   >
   </applet>
*/
public class abc extends Applet{
    public void paint(Graphics g){
    g.drawString("Hello Java",10,100);
  }
}
```

The applet tag is included in the java source file(abc.java) in comments. Now compile the code to get abc.class file. Then run appletviewer abc.java to start the applet. By this method it is easy to develop and change the applet code.

The applet tag

```
<APPLET
  [CODEBASE = CODEBASE_URL]        (specify url of dir where applet class is stored)
    CODE=Appletfilename.class            (name of applet class file)
     [ALT=alternate_text]                    (show text if applet doesn't run)
     [NAME = applet_instance_name]      (name of applet instance)
     WIDHT = pixels
     HEIGHT= pixels
     [ALIGN = alignment]                      ( TOP BOTTOM LEFT RIGHT etc)
     [ VSPACE =pixels]                        (for TOP BOTTOM specify blank space)
     [ HSPACE =pixels] >                      (for LEFT RIGHT specify blank space)
[  <PARAM NAME =name1 VALUE=value1>]
[  <PARAM NAME =name2 VALUE=value2>]      (pass parameter to applet)
-------------
[Text to be displayed in the absence of java]
</APPLET>                                      *(Bolds are necessary)
```

Paint( ) method

- The **paint( )** method is called each time your applet’s output must be dreaw/redrawn.
- For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.
- The **paint( )** method has one parameter of type **Graphics class** , which describes the graphics environment in which the applet is running.

Update() method

- This method is called when your applet has requested that a portion of its window be redrawn.
- The default version of **update( )** first fills an applet with the default background color and then calls **paint( )** method to paint the rest of the component.

repaint() method

- Calling a repaint () method causes the whole component to be repainted.
- repaint() -> update() ->paint()
- The **repaint( )** method has four forms.
- void repaint( )
- void repaint(int *left*, int *top*, int *width*, int *height*)
- void repaint(long *maxDelay*)
- void repaint(long *maxDelay*, int *x*, int *y*, int *width*, int *height*)

Passing parameter to applet

```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet{
     String str;
      public void init()
      {
       str=getParameter("string");        //receive parameter value
        if(str==null)
         str="java";
         str="Hello "+str;                  //use the value
      }
      public void paint(Graphics g)
      {
        g.drawString(str,10,100);
      }
}
```

```
<applet
  code=Hellojava.class
  width=400
  height=200>
  <PARAM NAME = "string"  VALUE = "Applet"]
  >
</applet>
```

Displaying numerical values

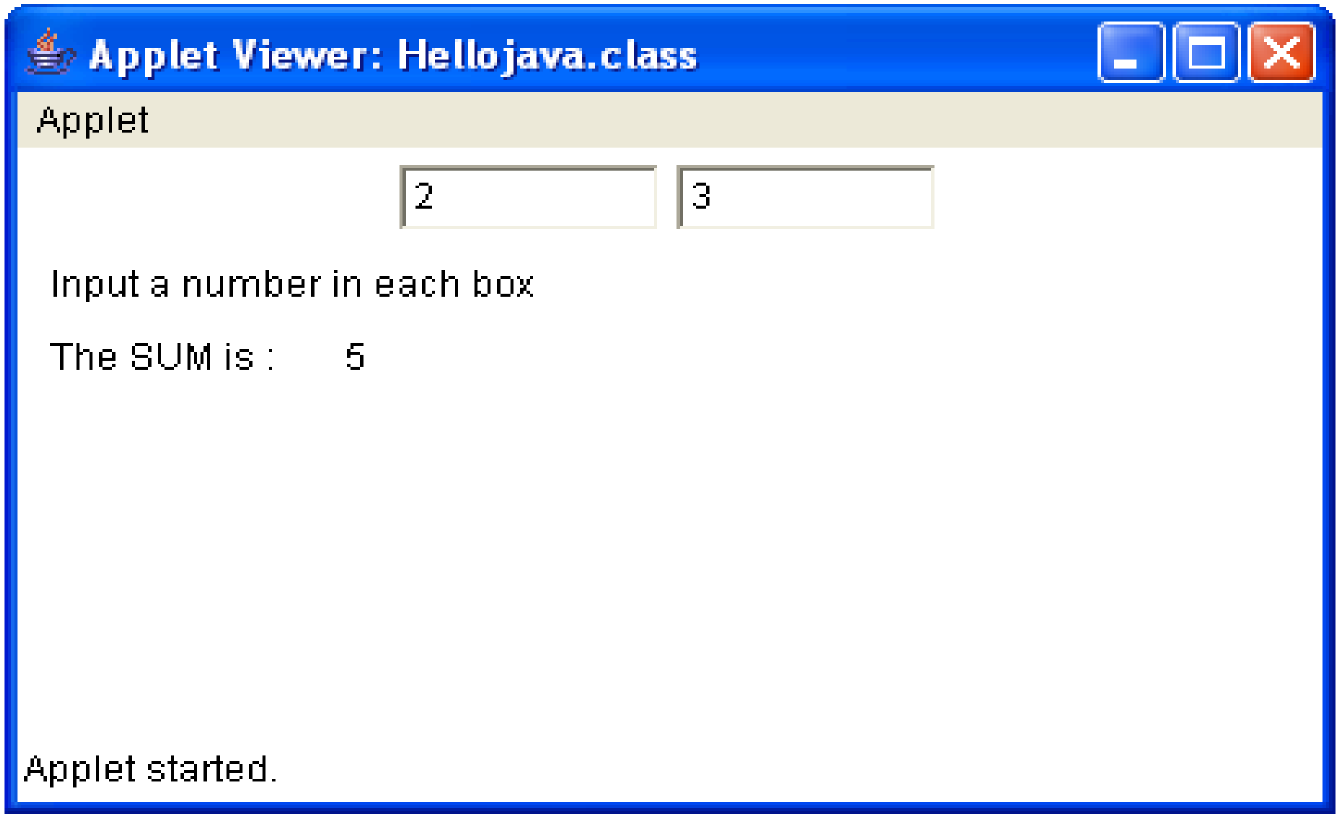
```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet{
    public void paint(Graphics g)
     {
       int value1 =10;
       int value2 = 20;
       int sum =value1+value2;
       String s= "Sum:"+String.valueOf(sum);
       g.drawString(s,10,100);
     }
}
```

Taking input from user

- Applet works in graphical environment. So applet treats the input as text string.
- The TextField class of applet package is used for creating a text field on applet.
- The input from text field is in string form, so it should be changed to right format before any computation.
- For displaying the output the result should be converted to String.

```
import java.awt.*;
import java.applet.*;
public class Hellojava extends Applet
{
   TextField text1,text2;
    public void init()
    {
     text1 = new TextField(8);
     text2 = new TextField(8);
     add(text1);
     add(text2);
     text1.setText("0");
     text2.setText("0");
    }
public void paint(Graphics g)  {
        int x=0,y=0,z=0;
        String s1,s2,s;
        g.drawString("Input a number in each box ",10,50);
        try{
           s1=text1.getText();
           x=Integer.parseInt(s1);
           s2=text2.getText();
           y=Integer.parseInt(s2);  }
        catch(Exception e){  }
        z=x+y;
        s=String.valueOf(z);
        g.drawString("The SUM is :",10,75);
        g.drawString(s,100,75);
      }
   public boolean action(Event event, Object obj)  {
       repaint();
       return true;
      }
  }
```

```
<applet
    code=Hellojava.class
    width=400
    height=200>
</applet>
```



```
2        3

Input a number in each box

The SUM is :     5

Applet started.
```