```
        public static void main(String args[])
            {
            try
             {
          int x;
          DataInputStream dis=new DataInputStream(System.in);
          System.out.println("Enter a no. ");
          x=Integer.parseInt(dis.readLine());
          if(x<1 || x>10)
                    throw new RangeException();
          else
                    System.out.println("number accepted");
          }
          catch(IOException ioe)
             {
              System.out.println(ioe);
             }
           catch(RangeException re)
             {
             re.disp();
             }
         finally
             {
             System.out.println("I am from finally block");
             }
}
}
```

Applet programming

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 appletviewer or any web browser that supports java.

Difference between applet and application

i) Applet don't use the main() method for initiating the execution of the code. When loaded automatically call certain methods of applet class to start and execute the applet code.

ii) Unlike stand-alone applications, applets can't be run independently. They are run from inside a web page using a special feature known as HTML tag.

iii) Applets can't read from or write to the files in the local computer.

iv) Applets are restricted from using libraries from other languages such as C or C++.

The steps involved in developing and testing in applet are :

1. Building an applet code (.java file)
2. Creating an executable applet (.class file)
3. Designing a web page using HTML Tags.
4. preparing <APPLET> tag.
5. Incorporating <APPLET> tag into the web page.
6. Creating HTML file.
7. Testing the applet code.

Write a simple program in Applet to print "Hello Java"

```
import java.awt.*;
import java.applet.*;
public class app1 extends Applet
  {
  public void paint(Graphics g)
    {
    g.drawString("Hello java",40,60);
    }
  }
```

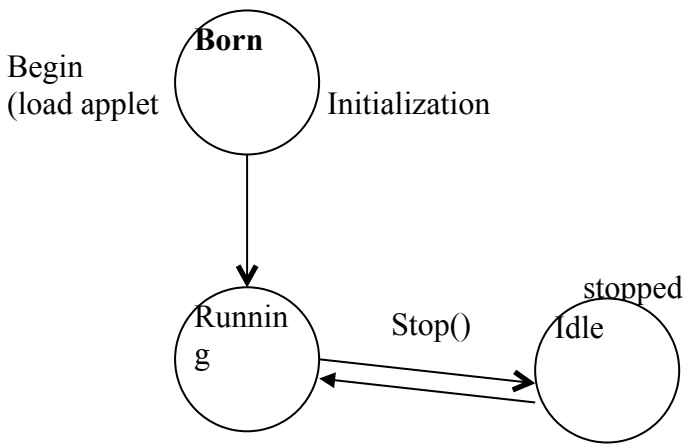
Compile -> javac app1.java

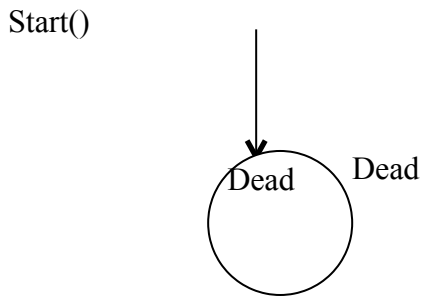
abc.html

```
<html>
<applet code="app1.class" width=400 height=400>
</applet>
</html>
```

For run
appletviewer abc.html

Applet life cycle





i) Initialization state -> Applet enters the initialization state when it is first loaded. This is achieved by calling the init() method of applet class. The applet is born. At this stage, we may do the following if required.
-> create objects needed by the applet
-> set up initial values.
-> load images or fonts.
-> setup colors.

```
public void init()
  {
  ……————
  ……————. (Action)
  }
```

ii) Running state :

Applet enters the running state when the system calls the start() method of applet class. This occurs automatically after the applet is initialized. Starting can also occur if the applet is already in stopped state.

```
public void start()
   {
   ……————..
   ……————
   }
```

iii) Idle or stopped state

An applet becomes idle when it is stopped from running. Stopping occurs automatically when we leave the page containing the currently running applet.

```
public void stop()
   {
   ……————..
   ……————
   }
```

iv) 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.

```
public void destroy()
```

```
{
……..
……..
}
```

v) Display state

```
public void paint(Graphics g)
  {
  ……
  ……
  }
```

Displaying numeric values

In applets we can display numeric values by first converting them into strings and then using the drawstring() method of Graphics class. We can do this by calling the valueOf() method of string class.

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

numvalues.html

```
<html>
<applet code=”numvalues.class” width=400 height=400>
</applet>
</html>
```

Getting input from the user

Applets work in graphical environment. Therefore, applets treat as text strings. We must first create an area of the screen in which user can type and edit input items.

Next step is to retrieve the items from the fields for display of calculations.

Develop an applet that receives three numeric values as input from the user and then displays the largest of the three on the screen. Write an HTML page and test the applet

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

```
import java.awt.*;

public class large extends Applet
        {
        TextField text1,text2,text3;
        public void init()
            {
            text1 = new TextField(10);
            text2 = new TextField(10);
            text3 = new TextField(10);
            add(text1);
            add(text2);
            add(text3);
            }
        public void paint(Graphics g)
             {
             int num1 = 0;
             int num2 = 0;
             int num3 = 0;
             String s1, s2, s3;

g.drawString("Input a number in each box ", 10, 50);
try
   {
   s1 = text1.getText();
   num1 = Integer.parseInt(s1);
   s2 = text2.getText();
   num2 = Integer.parseInt(s2);
   s3= text3.getText();
   num3=Integer.parseInt(s3);
   }
catch(Exception e1)
    {}
   if(num1>num2)
         if(num1>num3)
             {
             String str="The largest is :" +String.valueOf(num1);
             g.drawString (str,100, 125);
             }
        else
             {
   String str1="The largest is :" +String.valueOf(num3);
   g.drawString (str1,100, 125);

   }
   else
        if(num2>num3)
             {
             String str3="The largest is :" +String.valueOf(num2);
             g.drawString (str3,100, 125);
```

```
            }
             else
              {
              String str4="The largest is :"+String.valueOf(num3);
             g.drawString (str4,100, 125);
              }
        }
        public boolean action(Event ev, Object obj)
        {
                repaint();
                return true;
        }
}
```

WAP using Applet to calculate sum of its digit.

```
import java.applet.*;
import java.awt.*;
public class sum extends Applet
  {
  public void paint(Graphics g)
   {
   int num=123,rev,sum=0;
   while(num>0)
    {
    rev=num%10;
    sum=sum+rev;
    num=num/10;
    }
   String str="sum=" +String.valueOf(sum);
   g.drawString(str,100,25);
   }
  }
```

Coordinate System

By Default, the upper left corner of a GUI component (such as applet or window) has the coordinates (0,0). A Coordinate pair is composed of x-coordinate (the horizontal coordinate) and a y-coordinate (the vertical coordinate). The x-coordinate is the horizontal distance moving right from the upper left corner.

The y-coordinate is the vertical distance moving down from the upper left corner. The x-axis describes every horizontal coordinate, and the y-axis describes every vertical coordinate.

Drawing Lines

call
g.drawLine(x1,y1,x2,y2)
method, where `(x1, y1)` and `(x2, y2)` are the endpoints of our lines and g is the `Graphics` object we are drawing with. The following program will result in a line on the applet.

```
import java.applet.*;
import java.awt.*;
public class SimpleLine extends Applet
 {
  public void paint(Graphics g)
    {
     g.drawLine(10, 20, 30, 40);
     }
    }
```

Drawing Rectangles

Drawing rectangles is simple. Start with a `Graphics` object `g` and call its drawRect() method:

```
public void drawRect(int x, int y, int width, int height)
g.drawRoundRect(10,100,80,50,10,10);
```

The first argument `int` is the left hand side of the rectangle, the second is the top of the rectangle, the third is the width and the fourth is the height.

Drawing Ovals and Circles

Java has methods to draw outlined and filled ovals. These methods are called drawOval(), and fillOval() respectively. These two methods are:

```
public void drawOval(int left, int top, int width, int height)
public void fillOval(int left, int top, int width, int height)
```

Instead of dimensions of the oval itself, the dimensions of the smallest rectangle, which can enclose the oval, are specified.

If the width and height are same oval becomes a circle.

```
public void paint(Grpahics g)
   {
   g.drawOval(20,20,200,120);
   g.setColor(Color.green);
   g.fillOval(70,30,100,100); // This is a circle.
  }
```

Drawing Arcs

An arc is a part of an oval. The drawArc() designed to draw arcs takes six arguments. The first four are the same as the arguments for drawOval() method and the last two represent the starting angle of the arc and the number of degrees (sweep) angle around the arc.

```
g.drawArc(60,125,80,40,180,180);

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