Ch-5 Java Applets and Graphics Programming Marks: 10

Applet

- Applet is java program that can be embedded into html pages.
- Applets transported over the internet form one computer to another and run using applet viewer or can run on any java enabled web browser using JVM.

  OR
- An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application. The applet class provides a standard interface between applet and their environment.

Advantages of applet

- It provides GUI, facilitates graphics animation and multimedia.
- Avoids risk and provide 2 way interactions between webpages.
- Applets can work on all versions of Java plugin.

Disadvantages of applet

- Java applets requires JVM so first time it takes significant startup time.
- It is difficult to design and build good user interface in applet as compare to html technology.
- Some organization only allowed software installed by administrator, As a result many users cannot view applet by default.

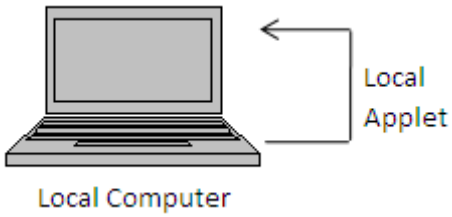
Types of Applets

1. Local Applets

- An applet developed locally and stored in a local system is known as a local applet.
- When a web page is trying to find a local applet, it does not need to use the internet and therefore the local system does not require the internet

connection. It simply searches the directories in the local system; locates and loads the specified applet.

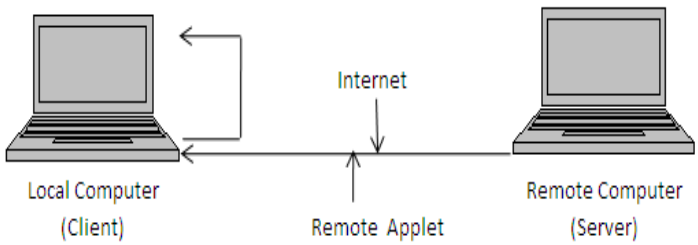
- A local applet is specified by a path name and a file name.



- For e.g. <applet
  code="JPR.class"
  width=120
  height=120>
  </applet>

2. Remote Applet

- A **remote applet** is that which is developed by someone else and stored on a remote computer connected to the internet. If our system is connected to the internet, we can download the remote applet into our system via at the internet and run it.
- Your browser must, of course, be connected to the Internet at the time it needs to display the remote applet.
- For locate and load a remote applet, we must know the applet's address on the web. This address is known as Uniform Resource Location (URL).

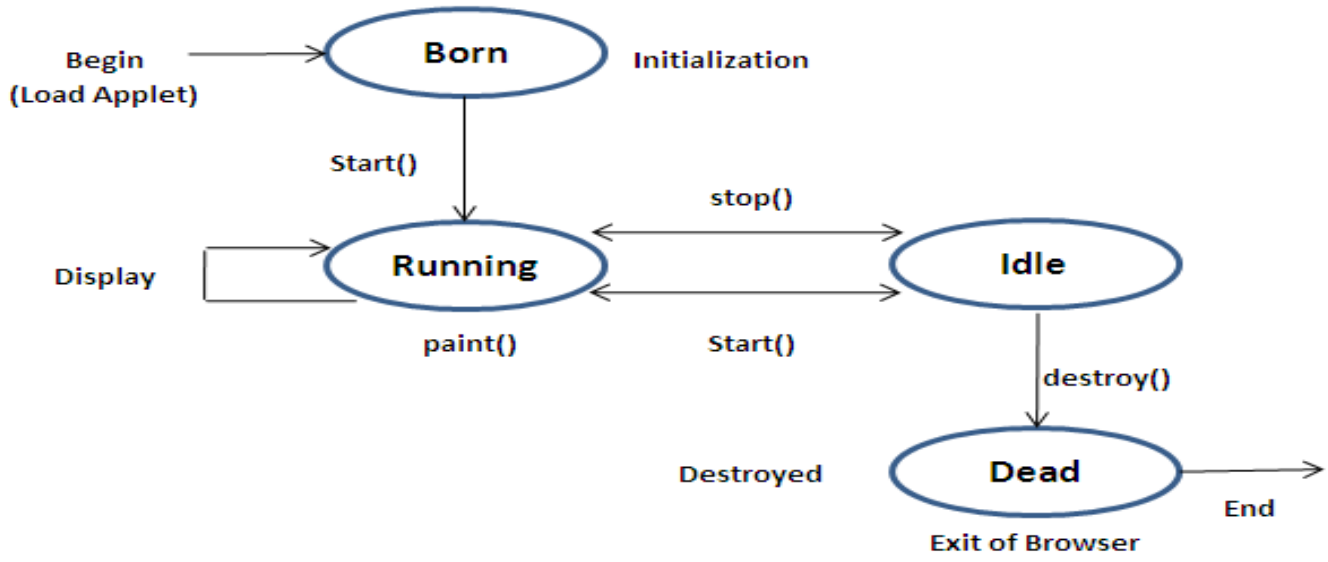


- For e.g.
  <applet
  codebase="http://www.vpt.edu.in/applets/"

code="JPR.class"
width=120
height=120>

Applet	Application
Applet does not use main() method for initiating execution of code	Application use main() method for initiating execution of code
Applet cannot run independently	Application can run independently
Applet cannot read from or write to files in local computer	Application can read from or write to files in local computer
Applet cannot communicate with other servers on network	Application can communicate with other servers on network
Applet cannot run any program from local computer.	Application can run any program from local computer.
Applet are restricted from using libraries from other language such as C or C++	Application are not restricted from using libraries from other language
Applets are event driven.	Applications are control driven.

Life Cycle of Applet



- **Applet Life Cycle:** An Applet has a life cycle, which describes how it starts, how it operates and how it ends. The life cycle consists of four methods:
  1. init()
  2. start()
  3. stop()
  4. destroy()

1. Initialization State (The init() method):

- The life cycle of an Applet is begin on that time when the applet is first loaded into the browser and called the init() method.
- The init() method is called only one time in the life cycle on an Applet. The init() method is basically called to read the "PARAM" tag in the html file.
- The init () method retrieve the passed parameter through the "PARAM" tag of html file using getParameter() method. All the initialization such as initialization of variables and the objects like image, sound file are loaded in the init() method.
- After the initialization of the init() method user can interact with the Applet and mostly applet contains the init() method.
- Syntax:

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

2. Running State (The start() method):

- The start method of an Applet is called after the initialization method init(). This method may be called multiples time when the Applet needs to be started or restarted.
- For Example if the user wants to return to the Applet, in this situation the start() method of an Applet will be called by the web browser and the user will be back on the applet. In the start method user can interact within the applet.

- Syntax:- public void start()
  {
  ……..
  ……..
  }

3. Idle (The Stop() method):

- An applet becomes idle when it is stopped from running. The stop() method stops the applet and makes it invisible.
- Stopping occurs automatically when we leave the page containing the currently running applet. We can also do so by calling the stop() method explicitly.
- The stop() method can be called multiple times in the life cycle of applet like the start () method or should be called at least one time.
- For example the stop() method is called by the web browser on that time When the user leaves one applet to go another applet and the start() method is called on that time when the user wants to go back into the first program or Applet.
- **Syntax:-** public void stop()
  {
  ……..
  ……..
  }

4. Dead State (The destroy() method):

- The destroy() method is called to terminate an Applet. 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. It is useful for clean-up actions, such as releasing memory after the applet is removed, killing off threads and closing network/database connections.
- Thus this method releases all the resources that were initialized during an applet's initialization.
- Syntax:-public void destroy()
  {
  ……..

……..

}

5. Display State (The paint() method):

- The paint() method is used for applet display on the screen. The display includes text, images, graphics and background.
- This happens immediately after the applet enters into the running state. Almost every applet will have a paint() method and can be called several times during an applet's life cycle.
- The paint() method is called whenever a window is required to paint or repaint the applet.
- Syntax:-public void paint(Graphics g)

  {

  ……..

  ……..

  }

Write a simple applet which display message 'Welcome to Java'.

Program:

```
/*<applet code= WelcomeJava width= 300 height=300>
</applet>*/
import java. applet.*;
import java.awt.*;
public class WelcomeJava extends Applet
{
public void paint( Graphics g)
{
g.drawString("Welcome to java",25,50);
}
}
```

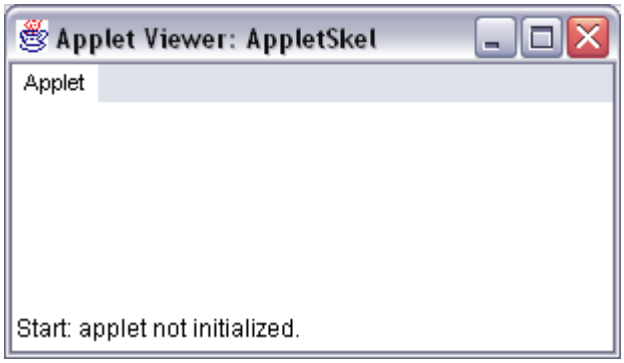
Applet Skeleton

- Applets override a set of methods that provides the basic mechanism by which the browser or appletviewer interfaces to the applet and controls its execution.
- Four of these methods - init (), start (), stop () and destroy ()-are defined by Applet. Paint() method is defined by the AWT component class. Default implementation for all of these methods is provided.
- These five methods can be assembled into the skeleton as shown below:

```
// An Applet skeleton.
import java.awt.*;
import java.applet.*;
/*
<applet code="AppletSkel" width=300 height=100>
</applet>
*/
public class AppletSkel extends Applet
{  // Called first.
  public void init()
 {    // initialization
 }

  /* Called second, after init().  Also called whenever   the applet is restarted.
*/
  public void start()
{    // start or resume execution
 }

  // Called when the applet is stopped.
  public void stop()
{    // suspends execution
 }

  /* Called when applet is terminated.  This is the last   method executed. */
 public void destroy()
```

```
{   // perform shutdown activities
 }

  // Called when an applet's window must be restored.
 public void paint(Graphics g)
   {   // redisplay contents of window
 }
}
```

The skeleton does not do anything; it can be compiled and run. When run, it generates the following output.



Adding Applet to the HTML file

Steps to add an applet in HTML document

1. Insert an <APPLET> tag at an appropriate place in the web page i.e. in the body section of HTML file.
2. Specify the name of the applet's .class file.
3. If the .class file is not in the current directory then use the codebase parameter to specify:-
   a. the relative path if file is on the local system, or
   b. the uniform resource locator(URL) of the directory containing the file if it is on a remote computer.
4. Specify the space required for display of the applet in terms of width and height in pixels.
5. Add any user-defined parameters using <param> tags
6. Add alternate HTML text to be displayed when a non-java browser is used.
7. Close the applet declaration with the </APPLET> tag.

Open notepad and type the following source code and save it into file name "Hellojava.java"

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

Use the java compiler to compile the applet "Hellojava.java" file.

C:\jdk> javac Hellojava.java

After compilation "Hellojava.class" file will be created. Executable applet is nothing but the .class file of the applet, which is obtained by compiling the source code of the applet. If any error message is received, then check the errors, correct them and compile the applet again.

We must have the following files in our current directory.

- Hellojava.java
- Hellojava.class
- HelloJava.html

If we use a java enabled web browser, we will be able to see the entire web page containing the applet.

We have included a pair of <APPLET..> and </APPLET> tags in the HTML body section. The <APPLET…> tag supplies the name of the applet to be loaded and tells the browser how much space the applet requires. The <APPLET> tag given below specifies the minimum requirements to place the HelloJava applet on a web page. The display area for the applet output as 300 pixels width and 200 pixels height. CENTER tags are used to display area in the center of the screen.

<APPLET CODE = hellojava.class WIDTH = 400 HEIGHT = 200 >
</APPLET>

Example: Adding applet to HTML file:

Create Hellojava.html file with following code:

<HTML>

<! This page includes welcome title in the title bar and displays a welcome message. Then it specifies the applet to be loaded and executed.

>

<HEAD> <TITLE> Welcome to Java Applet </TITLE> </HEAD>

<BODY> <CENTER> <H1> Welcome to the world of Applets </H1> </CENTER>

<CENTER>

<APPLET CODE=HelloJava.class WIDTH = 400 HEIGHT = 200 >
</APPLET>

</CENTER>

</BODY>

</HTML>

<PARAM> Tag or passing parameters to applet

- To pass parameters to an applet <PARAM… > tag is used.
- Each <PARAM…> tag has a name attribute and a value attribute.
- Inside the applet code, the applet can refer to that parameter by name to find its value.
- The syntax of <PARAM…> tag is as follows

<PARAM NAME = name1 VALUE = value1>

To set up and handle parameters, two things must be done.

1. Include appropriate <PARAM..> tags in the HTML document.
2. Provide code in the applet to parse these parameters.
   Parameters are passed on an applet when it is loaded. Generally init() method in the applet is used to get hold of the parameters defined in the <PARAM…> tag. The getParameter() method, which takes one string argument representing the name of the parameter and returns a string containing the value of that

parameter.

Example

```
import java.awt.*;
import java.applet.*;
public class hellouser extends Applet
{
String str; public void init()
{
str = getParameter("username");
str = "Hello "+ str;
}
public void paint(Graphics g)
{
g.drawString(str,10,100);
}
}
<HTML>
<Applet code = hellouser.class width = 400 height = 400>
<PARAM NAME = "username" VALUE = abc>
</Applet>
</HTML>
```

Attributes of Applet tags

The HTML APPLET Tag and attributes

The APPLET tag is used to start an applet from both an HTML document and from an applet viewer.

The syntax for the standard APPLET tag:

< APPLET

[CODEBASE = *codebaseURL*]

CODE = *appletFile*

[ALT = *alternateText*]

[NAME = *appletInstanceName*]

WIDTH = *pixels*
HEIGHT = *pixels*
[ALIGN = *alignment*]
[VSPACE = *pixels*]
[HSPACE = *pixels*]>
[< PARAM NAME = *AttributeName* VALUE = *AttributeValue*>]
[< PARAM NAME = *AttributeName2* VALUE = *AttributeValue*>]
. . .
</APPLET>

- **CODEBASE** is an optional attribute that specifies the base URL of the applet code or the directory that will be searched for the applets executable class file.
- **CODE** is a required attribute that give the name of the file containing your applet's compiled class file which will be run by web browser or appletviewer.
- **ALT:** Alternate Text. The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser cannot run java applets.
- **NAME** is an optional attribute used to specifies a name for the applet instance.
- **WIDTH AND HEIGHT** are required attributes that give the size (in pixels) of the applet display area.
- **ALIGN** is an optional attribute that specifies the alignment of the applet. The possible value is: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.
- **VSPACE AND HSPACE** attributes are optional, VSPACE specifies the space, in pixels, about and below the applet. HSPACE VSPACE specifies the space, in pixels, on each side of the applet.
- **PARAM NAME AND VALUE:** The PARAM tag allows you to specifies applet-specific arguments in an HTML page applets access there attributes with the getParameter()method.

Graphics programming, Class and Methods

- Java graphics class includes methods for drawing many different types of shapes, from simple line to polygon to text in variety of form.
- Graphics is class present in java.awt package in which is used to perform all the graphical operation in applet window.
- There are different methods available in graph class which are explain as follows:

---

1. Display Text:-

drawString() method is used to display the string in an applet window

Syntax: void drawString(String message, int x, int y);

where message is the string to be displayed beginning at x, y

Example: g.drawString("WELCOME", 10, 10);

---

2. drawOval( )

Drawing Ellipses and circles: To draw an Ellipses or circles used drawOval() method can be used.

Syntax: void drawOval(int top, int left, int width, int height)

The ellipse is drawn within a bounding rectangle whose upper-left corner is specified by top and left and whose width and height are specified by width and height to draw a circle or filled circle, specify the same width and height the following program draws several ellipses and circle.

Example: g.drawOval(10,10,50,50);

To draw filled oval use the method g.fillOval(30,140,70,40)

---

3. **drawPolygon()**

drawPolygon() method is used to draw arbitrarily shaped figures.

Syntax: void drawPolygon(int x[], int y[], int numPoints)

The polygons end points are specified by the co-ordinates pairs contained within the x and y arrays. The number of points define by x and y is specified by numPoints.

Example:

```
int xpoints[]={30,200,30,200,30};
int ypoints[]={30,30,200,200,30};
int num=5;
```

g.drawPolygon(xpoints,ypoints,num);

To draw filled polygon fillPolygon() is used.

4. **drawArc( )**

It is used to draw arc.

Syntax: void drawArc(int x, int y, int w, int h, int start_angle, int sweep_angle);

where x, y starting point, w & h are width and height of arc, and start_angle is starting angle of arc, sweep_angle is degree around the arc

Example: g.drawArc(10, 10, 30, 40, 40, 90);

5. **drawRect()**

The drawRect() method display an outlined rectangle.

Syntax: void drawRect(int top,int left,int width,int height)

The upper-left corner of the Rectangle is at top and left. The dimension of the Rectangle is specified by width and height.

Example: g.drawRect(10,10,60,50);

To draw filled rectangle fillRect() is used.

6. **drawLine()**

The **drawLine()** method is used to draw line which take two pair of coordinates, (x1,y1) and (x2,y2) as arguments and draws a line between them. The graphics object g is passed to paint() method.

Syntax: g.drawLine(x1,y1,x2,y2);

Example: g.drawLine(100,100,300,300);

7. drawRoundRect()

This methods draws rounded rectangle, it takes 6 parameters first 4 is same as drawRect()and last are x diameter and y diameter.

Syntax:- g.drawRoundRect(int x, int y, int width, int height, int Xdiameter, int ydiameter);

Example: g.drawRoundRect(30,140,70,40,10,10)

To draw filled rounded rectangle use the method g.fillRoundrect(30,140,70,40,10,10)

8. Setting Color:

Used to set color for graphics.

Syntax: g.setColor(Color.red);

9. Creating font object and Setting font

Syntax:- Font(String FontName, String FontStyle, int FontSize)

Example:-a=("Times New Roman", Font.BOLD+Font.ITALIC, 14)

Font Class

- The Font class states fonts, which are used to render text in a visible way.
- It is used to set or retrieve the screen font.

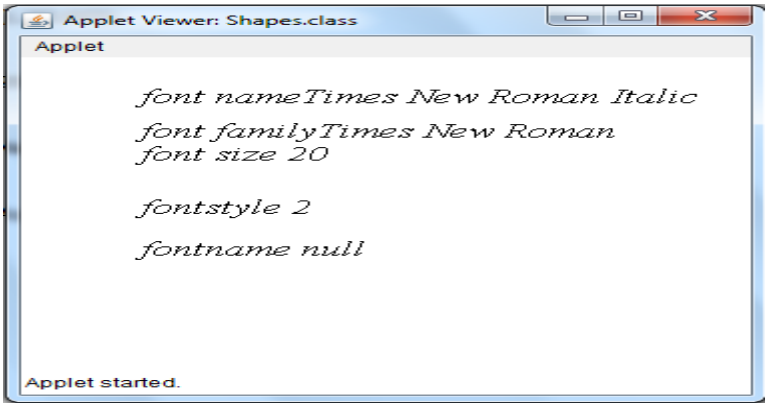
Methods	**Description**
String getFamily( )	Returns the name of the font family to which the invoking font belongs.
static Font getFont(String *property*)	Returns the font associated with the system property specified by *property.* **null** is returned if *property* does not exist.
String getFontName()	Returns the face name of the invoking font.
String getName( )	Returns the logical name of the invoking font.
int getSize( )	Returns the size, in points, of the invoking font.
int getStyle( )	Returns the style values of the invoking font.
int hashCode( )	Returns the hash code associated with the invoking object.
boolean isBold( )	Returns **true** if the font includes the **BOLD** style value. Otherwise, **false** is returned.
boolean isItalic( )	Returns **true** if the font includes the **ITALIC** style value. Otherwise, **false** is returned.
boolean isPlain( )	Returns **true** if the font includes the **PLAIN** style value. Otherwise, **false** is returned.

Example:

```
import java.awt.*;
import java.applet.*;
public class Shapes extends Applet
```

```
{
Font f,f1;
String s,msg;
String fname;
String ffamily;
int size;
int style;
public void init()
{
f= new Font("times new roman",Font.ITALIC,20);
setFont(f);
msg="is interesting";
s="java programming";
fname=f.getFontName();
ffamily=f.getFamily();
size=f.getSize();
style=f.getStyle();
String f1=f.getName();
}
public void paint(Graphics g)
{
g.drawString("font name"+fname,60,44);
g.drawString("font family"+ffamily,60,77);
g.drawString("font size "+size,60,99);
g.drawString("fontstyle "+style,60,150);
g.drawString("fontname "+f1,60,190);
}
}
/*<applet code=Shapes.class height=300 width=300></applet>*/
```

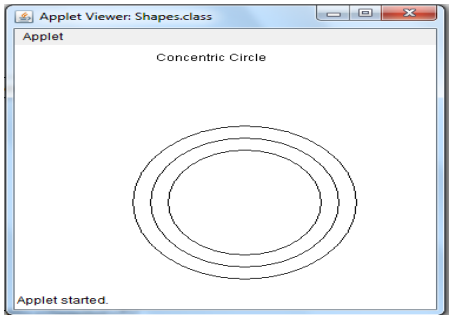
Output:



Q. Write a simple applet program which display tree concentric circle.

```
import java.awt.*;
import java.applet.*;
public class Shapes extends Applet
{
public void paint (Graphics g)
{
g.drawString("Concentric Circle",120,20);
g.drawOval(100,100,190,190);
g.drawOval(115,115,160,160);
g.drawOval(130,130,130,130);
}
}
/*<applet code="Shapes.class" height=300 width=200> </applet>*/
```

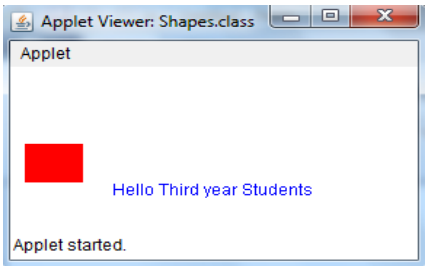
Output:



Q. Design an Applet program which displays a rectangle filled with red color and message as "Hello Third year Students" in blue color.

```
import java.awt.*;
import java.applet.*;
public class Shapes extends Applet
{
public void paint (Graphics g)
{
g.setColor(Color.red);
g.fillRect(10,60,40,30);
g.setColor(Color.blue);
g.drawString("Hello Third year Students",70,100);
}
}
/*<applet code="Shapes.class" height=300 width=200> </applet>*/
```

Output:

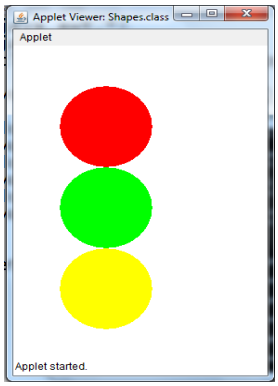


Q. Write a program to design an applet to display three circles filled with three different colors on screen.

```
import java.awt.*;
import java.applet.*;
public class Shapes extends Applet
{
public void paint (Graphics g)
{
g.setColor(Color.red);
g.fillOval(50,50,100,100);
```

```
g.setColor(Color.green);
g.fillOval(50,150,100,100);
g.setColor(Color.yellow);
g.fillOval(50,250,100,100);
}
}
/*<applet code="Shapes.class" height=300 width=200> </applet>*/
```

Output:



Q. Write an applet to accept a user name in the form of parameter and print „Hello<User Name>“.

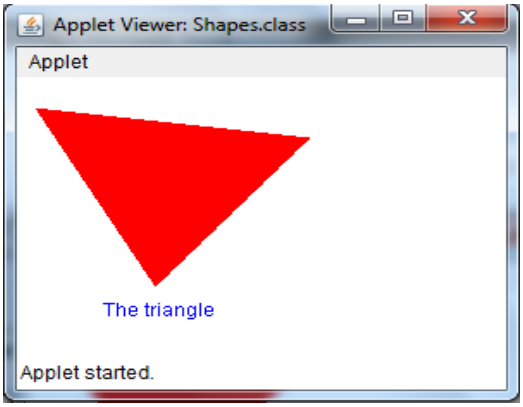
```
import java.lang.*;
import java.awt.*;
import java.applet.*;
public class Hello extends Applet
{
String message;
public void init()
{
message=getParameter("username");
message="Hello" + message;
}
public void paint(Graphics g)
{
g.drawString(message, 100,100);
```

```
}
}
/*<applet code=Hello.class height=300 width=300>
<param name="username" value="Abc">
</applet> */
```

Q. Design an applet which display a triangle filled with red colour and a message as "The triangle" in blue below it.

```
import java.awt.*;
import java.applet.*;
public class DrawTriangle extends Applet
{
public void paint(Graphics g)
{
int xPoints[] = { 10, 170, 80, 10 };
int yPoints[] = { 20, 40, 140, 20 };
int nPoints = xPoints.length;
g.set Color(Color.red);
g.fillPolygon(xPoints, yPoints, nPoints);
g.setColor(Color.blue);
g.drawString("The triangle",50,160);
}
}
/* <applet code="DrawTriangle.class" width="350" height="300">
</applet> */
```

Output:



Q. Write a program to draw a bar chart for plotting students passing percentage in last 5 years.

Consider the following data

Year	2001	2002	2003	2004
% result of subject	96	85	70	98

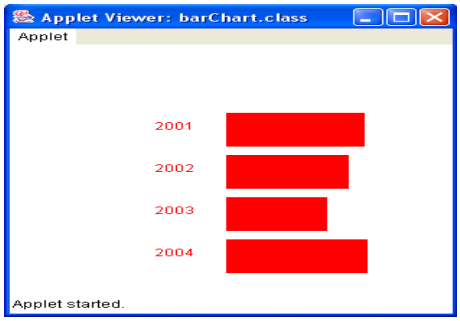
```
import java.awt.*;
import java.applet.*;
public class barChart extends Applet
{
String label[];
int  values[];
int n=0;
public void init()
{
try
{
n=Integer.parseInt(getParameter("cols"));
label=new String[n];
values=new int[n];
label[0]=getParameter("label1");
label[1]=getParameter("label2");
label[2]=getParameter("label3");
label[3]=getParameter("label4");
```

```
values[0]=Integer.parseInt(getParameter("values1"));
values[1]=Integer.parseInt(getParameter("values2"));
values[2]=Integer.parseInt(getParameter("values3"));
values[3]=Integer.parseInt(getParameter("values4"));
}
catch(NumberFormatException e){  }
}
public void paint(Graphics g)
{
g.setColor(Color.red);
for(int i=0;i<n;i++)
{
g.drawString(label[i],100,100+i*50);
g.fillRect(150,80+i*50,values[i],40);
}
}
}
```

Code for html file

```
<html>
<applet code="barChart.class" height=700 width=700>
        <param name="cols" value="4">

        <param name="label1" value="2001">
        <param name="label2" value="2002">
        <param name="label3" value="2003">
        <param name="label4" value="2004">

        <param name="values1" value="96">
        <param name="values2" value="85">
        <param name="values3" value="70">
        <param name="values4" value="98">
    </applet>
</html>
```

Output:



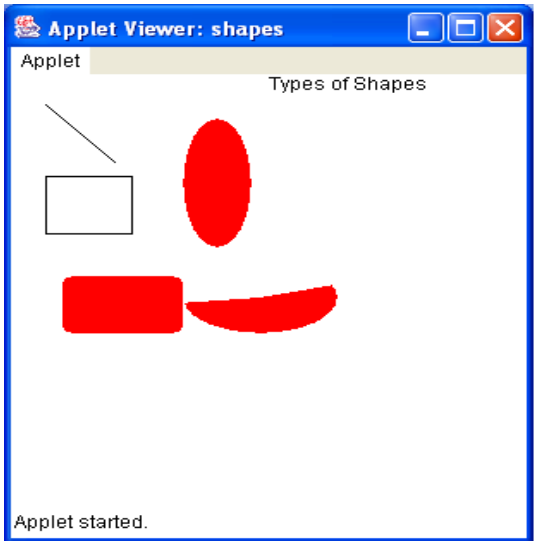
Q. Write a program to create an applet for displaying different shapes.

```
import java.awt.*;
import java.applet.*;
public class shapes extends Applet
{
public void paint(Graphics g)
{
g.setFont(new Font("Arial",10, 12));
g.drawString("Types of Shapes",150,10);
g.drawLine(20,20,60,60);
g.drawRect(20,70,50,40);
g.setColor(Color.red);
g.fillOval(100,30,40,90);
g.fillRoundRect(30,140,70,40,10,10);
g.fillArc(100,130,90,50,190,190);
}
}

/*<applet code=shapes height=300 width=300>
</applet>
*/
```

Output:

Javac shapes.java

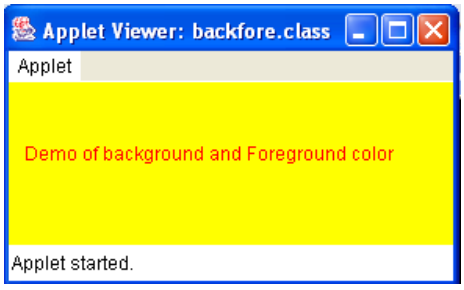
Appeltviewer shapes.java



Q. Write a program to set background and foreground color using applet

```
import java.awt.*;
import java.applet.*;
public class backfore extends Applet
{
public void paint(Graphics g)
{
setBackground(Color.yellow);

g.setColor(Color.red);
g.drawString("Demo of background and Foreground color",10,50);
}
}
/*<applet code="backfore.class" width=100 height=100>
</applet>
*/
```

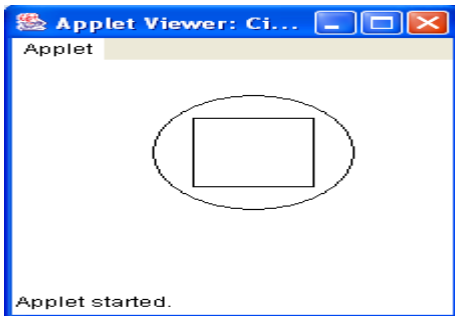
Output:



Q. Write the applets to draw square inside a circle

```
import java.awt.*;
import java.applet.*;
public class CirSqr extends Applet
{
public void paint(Graphics g)
{
g.drawOval(70,30,100,100);
g.drawRect(90,50,60,60);
}
}
/*<applet code="CirSqr.class" height=200 width=200>
</applet> */
```

Output:



Q. Write the applets to set background with red color and foreground with blue color.

```
/*<applet code="backfore.class" width=400 height=400>
</applet>*/
import java.awt.*;
import java.applet.*;
public class backfore extends Applet
{
public void init()
```

```
{
setBackground(Color.red);
}
public void paint(Graphics g)
{
g.setColor(Color.blue);
g.drawString("Welcome to graphics programming", 40, 70);
}
}
```