8.1 Introduction to Graphics Programming

- The graphics programming in Java is supported by the AWT package. The AWT stands for Abstract Window Toolkit.
- The AWT contains large number of classes which help to include various graphical components in the Java program. These graphical components include text box, buttons, labels, radio buttons, list items and so on. We need to import **java.awt** package for using these components.
- These classes are arranged in hierarchical manner which is recognised as AWT hierarchy.

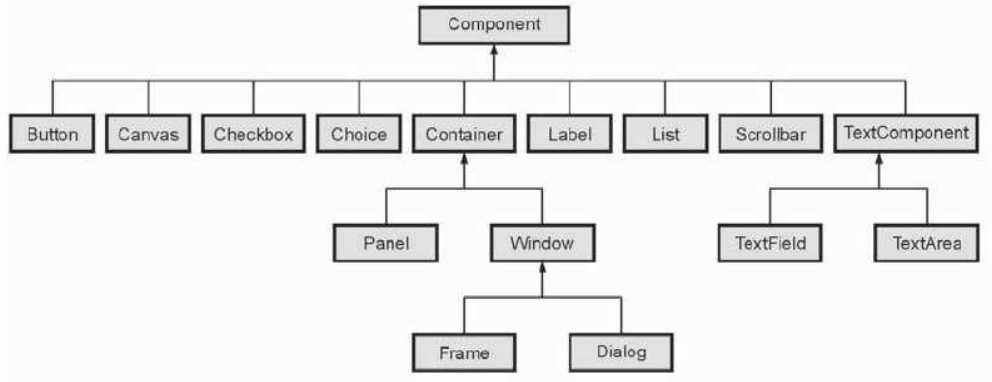


Fig. 8.1.1 AWT hierarchy

- The hierarchy components classes are -
  - **Component :** This is the super class of all the graphical classes from which variety of graphical classes can be derived. It helps in displaying the graphical object on the screen. It handles the mouse and keyboard events.
  - **Container :** This is a graphical component derived from **the component** class. It is responsible for managing the layout and placement of graphical components in the container.
  - **Window :** The top level window without border and without the menu bar is created using the window class. It decides the layout of the window.
  - **Panel :** The panel class is derived from the **container** class. It is just similar to window - without any border and without any menu bar, title bar.
  - **Frame :** This is a top-level window with a border and menu bar. It supports the common window events such as window open, close, activate and deactivate.

8.2 Frame

AU : IT : Dec.-11, Marks 8

- In Java, Frame is a standard graphical window.
- The frame can be displayed using the **Frame** class.
- The frame drawn using this class has standard minimize, maximize and close buttons.
- The syntax of frame class is -

i) Frame()

This creates the new instance of frame which is invisible initially.

ii) Frame(String title)

This creates the new instance of frame which has some title.

- Following table enlists various methods of **Frame class**

Methods	Description
void setResizable(boolean resizable)	Sets frame to resizable
void setTitle(String Title)	Sets the title of the frame
void setSize(int width,int height)	Sets the width and height of a frame
String getTitle()	Obtains the title of the frame
void setVisible(boolean visible)	Set the frame visible or not.

A frame can be created by two ways -

- By **extending the Frame class**
- By creating an **instance of a Frame class.**

Ex. 8.2.1 : Create a java frame by extending the Frame class.

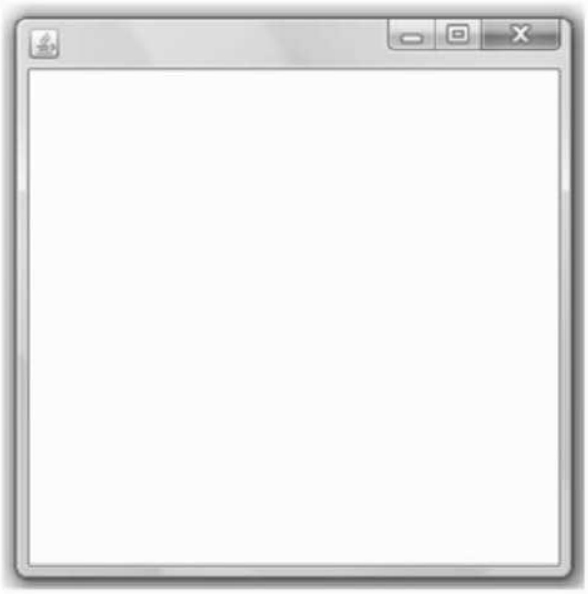
Sol. :

Java Program

```
import java.awt.*;
class FrameDemo extends Frame
{
        public static void main(String[] args)
        {
```

```
        FrameDemo fr=new FrameDemo();
        fr.setSize(300,300);
      fr.setVisible(true);
    }
}
```

Output



Note that the initially the frame will not be visible. Hence we need to set the visibility of the frame.

Ex. 8.2.2 : Create a java frame by using an instance of Frame class.

Sol. :

Java Program

```
import java.awt.*;
class FrameDemo1
{
        public static void main(String[] args)
        {
          Frame fr=new Frame();
          fr.setSize(300,300);
          fr.setVisible(true);
    }
}
```

Output will be the same frame as above.

Review Question

1. *Write the short note on - Frames.*

AU : IT : Dec.-11, Marks 8

8.3 Components

AU : May-19, Marks 9

- There are various graphical components that can be placed on the frame. These components have the classes. These classes have the corresponding methods.
- When we place the components on the frame we need to set the layout of the frame.
- The commonly used layout is **FlowLayout**. The **FlowLayout** means the components in the frame will be placed from left to right in the same manner as they get added.
- Various components that can be placed for designing user interface are -

1. Label	2. Buttons	3. Canvas	4. Scroll bars
5. Text Components	6. Checkbox	7. Choices	8. Lists Panels
9. Dialogs	10. Menubar		

- Let us discuss these components one by one, but before that let us understand how to create a frame on which we can place the components.

8.3.1 Labels

The syntax of this control is

Label (String s)
Label(String s, int style)

where the **s** of String type represent the string contained by the label similarly in the other label function style is a constant used for the style of label. It can be Label. LEFT, Label.RIGHT and Label.CENTER. Here is a JAVA program which makes use Label.

Ex. 8.3.1 : Write a simple Java program to demonstrate the use of label components.

Sol. :

Java Program[Use_Label.java]

```
import java.awt.*;
class Use_Label
{
        public static void main(String[] args)
        {
          Frame fr=new Frame("This Program is for Displaying the Label");
          fr.setSize(400,200);
          fr.setLayout(new FlowLayout());
          fr.setVisible(true);
          Label L1=new Label("OK");
```

```
        Label L2=new Label("CANCEL");
        fr.add(L1);
        fr.add(L2);
    }
}
```

Output

```
C:\>javac Use_label.java
C:\>java Use_label
```

8.3.2 Buttons

- Buttons are sometimes called as **push buttons**. This component contains a label and when it is pressed it generates an event.
- The syntax of this control is
  Button (String s)

Java Program

```
import java.awt.*;
class Use_Button
{
    public static void main(String[] args)
    {
    Frame fr=new Frame("This Program is for Displaying the Button");
    fr.setSize(400,200);
    fr.setLayout(new FlowLayout());
    fr.setVisible(true);
    Button B1=new Button("OK");
    Button B2=new Button("CANCEL");
    fr.add(B1);
    fr.add(B2);
    }
}
```

Output

- We can create an array of buttons. Following is a simple Java program which illustrates this ideas -

Java Program[Use_Button_Arr.java]

```
import java.awt.*;
class Use_Button_Arr
{
        public static void main(String[] args)
        {
                int i;
                Frame fr=new Frame("This Program is for Displaying the Buttons");
                fr.setSize(400,200);
                fr.setLayout(new FlowLayout());
                fr.setVisible(true);
                Button buttons[]=new Button[5];
                String Fruits[]={"Mango","Orange","Banana","Apple","Strawberry"};
                for(i=0;i<5;i++)
                {
                  buttons[i]=new Button(" "+Fruits[i]);
                  fr.add(buttons[i]);
                }
          }
}
```

Output

8.3.3 Canvas

- Canvas is a **special area** created on the frame.
- The canvas is specially used for drawing the graphical components such as oval, rectangle, line and so on.
- Various methods of this class are

Method	Description
void setSize(int width, int height)	This method sets the size of the canvas for given width and height.
void setBackground(Color c)	This method sets the background color of the canvas.
void setForeground(Color c)	This method sets the color of the text.

- Following is a simple Java program which shows the use of canvas -

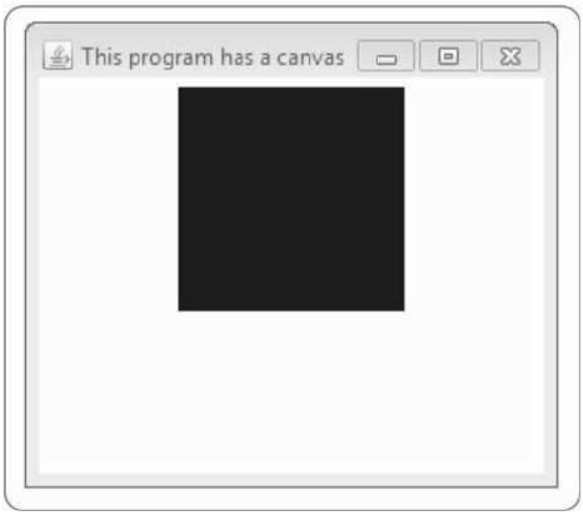
Java Program[Use_Canvas.java]

```
import java.awt.*;
class Use_Canvas
{
  public static void main(String[] args)
  {
    Frame Fr = new Frame("This program has a canvas");
    Canvas C1 = new Canvas();
    C1.setSize(120,120);
    C1.setBackground(Color.blue);
    Fr.setLayout(new FlowLayout());
```

```
        Fr.setSize(250,250);
        Fr.setVisible(true);
        Fr.add(C1);
      }
    }
```

Output

```
C:\>javac Use_Canvas.java
C:\>java Use_Canvas
```



8.3.4 Scrollbars

- Scrollbar can be represented by the slider widgets.
- There are two styles of scroll bars - Horizontal scroll bar and vertical scroll bar.
- Following program shows the use of this component.

Java Program

```
import java.awt.*;
class Use_ScrollBars
{
  public static void main(String[] args)
  {

    Frame Fr = new Frame("This program has a scrollbars");
    Scrollbar HSelector = new Scrollbar(Scrollbar.HORIZONTAL);
    Scrollbar VSelector = new Scrollbar(Scrollbar.VERTICAL);

    Fr.setLayout(new FlowLayout());
```

```
        Fr.setSize(300,300);
        Fr.setVisible(true);
        Fr.add(HSelector);
        Fr.add(VSelector);
      }
    }
```

Output

8.3.5 Text Components

- In Java there are two controls used for text box - One is **TextField** and the other one is **TextArea**.
- The **TextField** is a slot in which one line text can be entered. In the **TextField** we can enter the string, modify it, copy, cut or paste it. The syntax for the text field is

```
int TextField(int n)
```

where n is total number of characters in the string.

- The TextArea control is used to handle multi-line text. The syntax is -

```
TextArea(int n,int m)
```

where n is for number of lines and m is for number of characters.

Following is a Java program which illustrates the use of TextField and TextArea.

Java Program[Use_TxtFld.java]

```
import java.awt.*;
class Use_TxtFld
{
        public static void main(String[] args)
        {
        int i;
        Frame fr=new Frame("This Program is for Displaying the TextField");
        fr.setSize(350,300);
        fr.setLayout(new FlowLayout());
         fr.setVisible(true);
         Label L1=new Label("Enter your name here");
        TextField input1=new TextField(10);
         Label L2=new Label("Enter your address here");
        TextArea input2=new TextArea(10,20);
        fr.add(L1);
         fr.add(input1);
        fr.add(L2);
         fr.add(input2);
    }
}
```

Output

8.3.6 Checkbox

- Checkbox is basically a small box which can be ticked or not ticked.
- In Java we can select particular item using checkbox control.

- This control appears as small box along with label. The label tells us the name of the item to be selected.
- The syntax of checkbox is as given below -

Checkbox(String label)

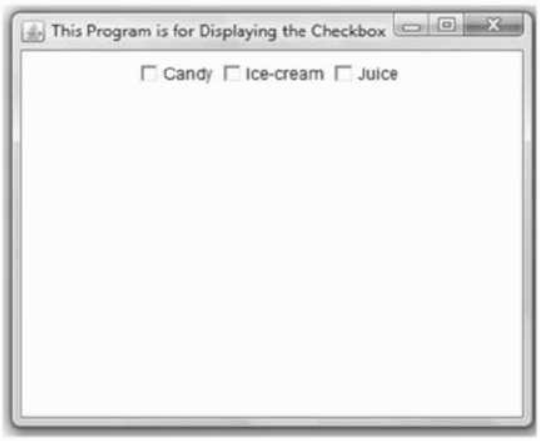
where *label* denotes the label associated with each checkbox.

- To get the state of the checkbox the **getState**() method can be used.

Java Program[Use_ChkBox.java]

```
import java.awt.*;
class Use_ChkBox
{
        public static void main(String[] args)
        {
        int i;
       Frame fr=new Frame("This Program is for Displaying the Checkbox");
        fr.setSize(350,300);
        fr.setLayout(new FlowLayout());
        fr.setVisible(true);
        Checkbox box1=new Checkbox("Candy");
        Checkbox box2=new Checkbox("Ice-cream");
        Checkbox box3=new Checkbox("Juice");
        fr.add(box1);
        fr.add(box2);
        fr.add(box3);
         }
}
```

Output



8.3.7 Checkbox Group

- The Checkbox Group component allows the user to make one and only one selection at a time.
- These checkbox groups is also called as **radio buttons**. The syntax for using checkbox groups is -

Checkbox(String str ,CheckboxGroup cbg , Boolean val);

- Following is a simple Java program which makes use of this control.

Java Program[Use_CheckBoxGr.java]

```
import java.awt.*;
class Use_CheckBoxGr
{
  public static void main(String[] args)
  {

    Frame Fr = new Frame("This program uses checkbox groups");
    Fr.setLayout(new FlowLayout());
    Fr.setSize(300,300);
    Fr.setVisible(true);
    CheckboxGroup cbg=new CheckboxGroup();
    Checkbox box1=new Checkbox("Candy",cbg,true);
    Checkbox box2=new Checkbox("Ice-cream",cbg,false);
    Checkbox box3=new Checkbox("Juice",cbg,false);
```

```
        Fr.add(box1);
        Fr.add(box2);
        Fr.add(box3);
      }
    }
```

Output

8.3.8 Choices

- This is a simple control which allows the popup list for selection.
- We have to create an object of type **choice** as follows -

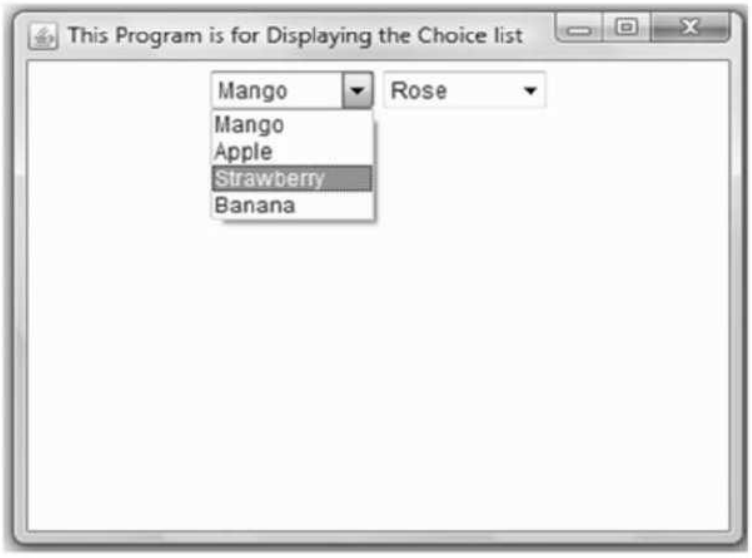
```
Choice obj=new Choice();
```

Java Program[Use_Choice.java]

```
import java.awt.*;
class Use_Choice
{
        public static void main(String[] args)
        {
        int i;
        Frame fr=new Frame("This Program is for Displaying the Choice list");
        fr.setSize(350,300);
        fr.setLayout(new FlowLayout());
        fr.setVisible(true);
        Choice c1=new Choice();
        Choice c2=new Choice();
        c1.add("Mango");
```

```
        c1.add("Apple");
        c1.add("Strawberry");
        c1.add("Banana");

        c2.add("Rose");
        c2.add("Lily");
        c2.add("Lotus");
        fr.add(c1);
        fr.add(c2);


    }
}
```

Output



8.3.9 List Panels

- **List** is a collection of many items.
- By double clicking the desired item we can select it. Following Java program makes use of this control -

Java Program[Use_List.java]

```
import java.awt.*;
class Use_List
{
        public static void main(String[] args)
```

```
        {
          int i;
          Frame fr=new Frame("This Program is for Displaying the List");
          fr.setSize(350,300);
          fr.setLayout(new FlowLayout());
          fr.setVisible(true);
          List flower=new List(4,false);
          flower.add("Rose");
          flower.add("Jasmine");
          flower.add("Lotus");
          flower.add("Lily");
          fr.add(flower);
    }
}
```

Output

Review Question

1. *Write the short note on - Frames.* AU : IT : Dec.-11, Marks 8
2. *Use graphics objects to draw an arc and a semicircle inside a rectangular box.* AU : May-19, Marks 4
3. *Sketch the hierarchy of Java AWT classes and methods. Create a 'checkbox' using these classes and methods.* AU : May-19, Marks 4

8.4 Working with 2D Shapes

AU : CSE : Dec.-10, 12, IT : Dec.-11, May-12, 15, 19, Marks 16

- Java has an ability to draw various graphical shapes. The applet can be used to draw these shapes.
- These objects can be colored using various colors.

- Every applet has its own area which is called **canvas** on which the display can be created.
- The co-ordinate system of Java can be represented by following Fig. 8.4.1.

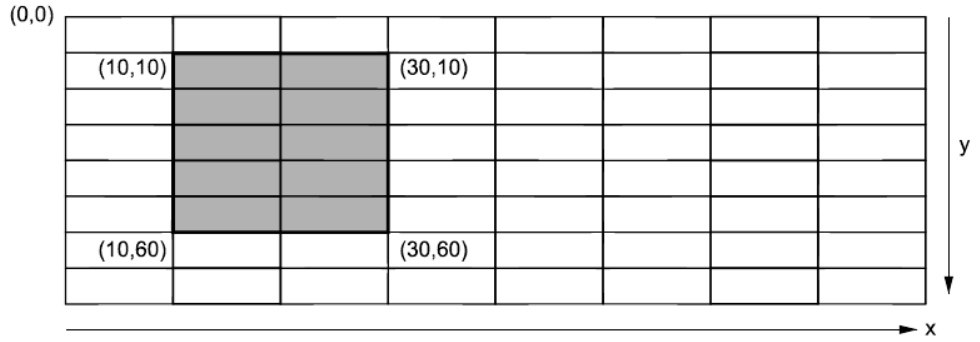


Fig. 8.4.1 Co-ordinate system

- Before drawing the 2D shapes we will need a special area on the frame. This area is called **canvas**. Various methods of this class are -

void setSize(int width,int height) - Sets the size of the canvas of given width and height.

void setBackground(Color c) - Sets the background color of the canvas.

void setForeground(Color c) - Sets the color of the text.

8.4.1 Lines

- Drawing the line is the simplest thing in Java. The syntax is,

```
void drawLine(int x1,int y1,int x2,int y2);
```

Where x1 and y1 represents the starting point of the line and x2 and y2 represents the ending point of the line.

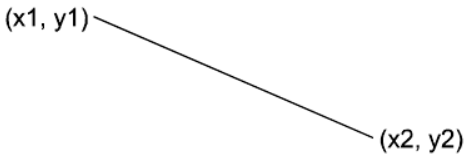


Fig. 8.4.2 Line

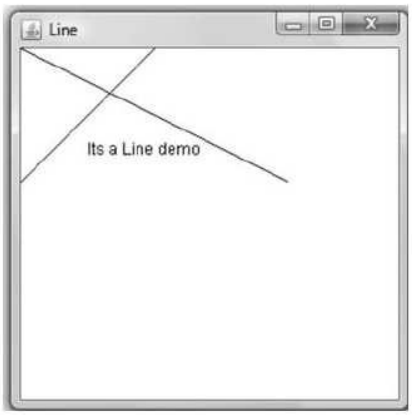
Here is a demonstration -

Java Program[LineDemo.java]

```
import java.awt.*;
class LineDemo extends Canvas
{
 public LineDemo()
```

```
 {
  setSize(200,200);
  setBackground(Color.white);
 }
 public static void main(String[] args)
 {
   LineDemo obj=new LineDemo();
   Frame fr=new Frame("Line");
   fr.setSize(300,300);
   fr.add(obj);
   fr.setVisible(true);
 }
 public void paint(Graphics g)
 {
  g.drawLine(0,0,200,100);//diagonal line from top-left
  g.drawLine(0,100,100,0);//anohter diagonal line
  g.drawString("Its a Line demo",50,80);

 }
}
```

Output



8.4.2 Rectangle

In Java we can draw two types of rectangles: normal rectangle and the rectangle with round corners. The syntax of these methods are -

```
void drawRect(int top,int left,int width,int height)
void drawRoundRect(int top,int left,int width,int height,int xdimeter,int ydimeter)
```

For example

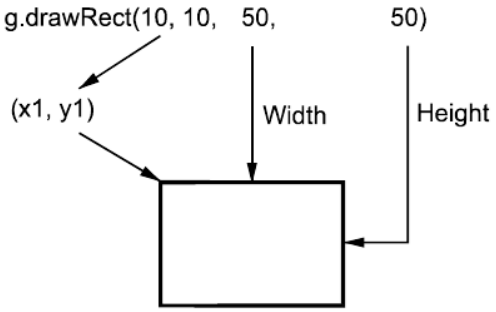


Fig. 8.4.3 Rectangle

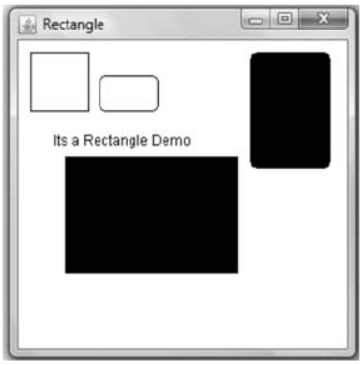
The rectangle can be filled up with following methods,
void fillRect(int top,int left,int width,int height)
void fillRoundRect(int top,int left,int width,int height,int xdimeter,int ydimeter)

Java Program[RectangleDemo.java]

```
import java.awt.*;
class RectangleDemo extends Canvas
{
 public RectangleDemo()
 {
  setSize(200,200);
  setBackground(Color.white);
 }
 public static void main(String[] args)
 {
   RectangleDemo obj=new RectangleDemo();
   Frame fr=new Frame("Rectangle");
   fr.setSize(300,300);
   fr.add(obj);
   fr.setVisible(true);
 }
 public void paint(Graphics g)
 {
  g.drawRect(10,10,50,50);
  g.drawRoundRect(70,30,50,30,10,10);
  g.fillRect(40,100,150,100);
```

```
  g.fillRoundRect(200,10,70,100,10,10);
  g.drawString("Its a Rectangle Demo",30,90);
 }
}
```

Output



8.4.3 Oval

To draw circle and ellipse we can use the function **drawOval()** method. The syntax of this method is ,

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

To fill this oval we use **fillOval()** method. The syntax of this method is,

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

We can specify the color for filling up the rectangle using setColor method. For example

```
g.drawOval(10,10,200,100);
g.setColor(Color.blue);
g.fillOval(30,50,200,100);
```

The *top* and *left* values specify the upper left corner and *width* and *height* is for specifying the width and heights respectively.

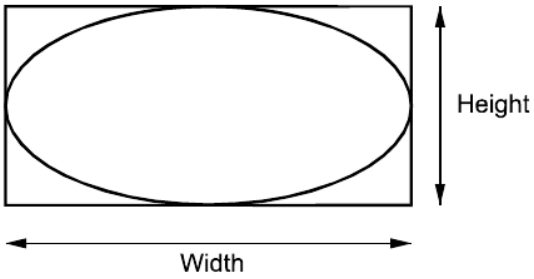


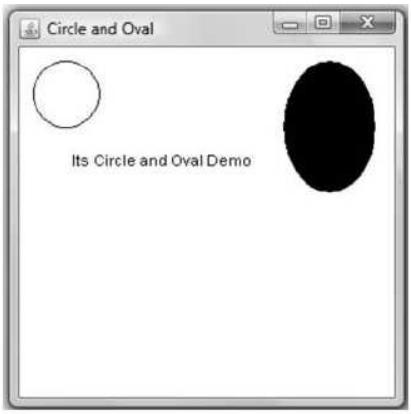
Fig. 8.4.4 Ellipse

Let us understand the functioning of these methods with the help of some example

Java Program[OvalDemo.java]

```
import java.awt.*;
class OvalDemo extends Canvas
{
 public OvalDemo()
 {
  setSize(200,200);
  setBackground(Color.white);
 }
 public static void main(String[] args)
 {
   OvalDemo obj=new OvalDemo();
   Frame fr=new Frame("Circle and Oval");
   fr.setSize(300,300);
   fr.add(obj);
   fr.setVisible(true);
 }
 public void paint(Graphics g)
 {
   g.drawOval(10,10,50,50);
   g.fillOval(200,10,70,100);
   g.drawString("Its Circle and Oval Demo",40,90);
 }
}
```

Output



8.4.4 Arc

To draw an arc the **drawArc()** and to fill an arc **fillArc()** are the functions. The syntax of these methods is as follows -

```
void drawArc(int top,int left,int width,int height,int angle1,int angle2)
void fillArc(int top,int left,int width,int height,int angle1,int angle2)
```

The *angle1* represents the starting angle and the *angle2* represents the angular distance.

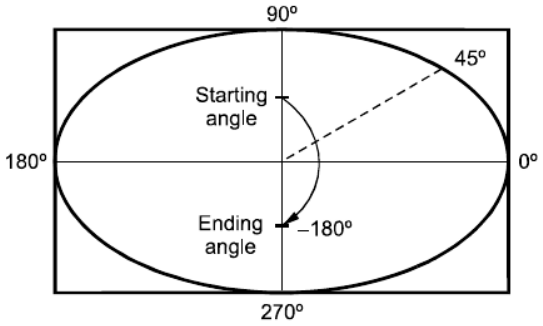


Fig. 8.4.5

Java Program[arcDemo.java]

```
import java.awt.*;
class arcDemo extends Canvas
{
 public arcDemo()
 {
  setSize(200,200);
  setBackground(Color.white);
 }
 public static void main(String[] args)
 {
   arcDemo obj=new arcDemo();
   Frame fr=new Frame("Arc");
   fr.setSize(300,300);
   fr.add(obj);
   fr.setVisible(true);
 }
 public void paint(Graphics g)
 {
  g.drawArc(100,60,100,100,0,90);
  g.setColor(Color.green);//fills the arc with green
  g.fillArc(100,60,55,70,0,90);
  g.setColor(Color.black);
  g.drawArc(100,100,70,90,0,270);
  g.drawString("Its an arc Demo",120,160);
 }
}
```

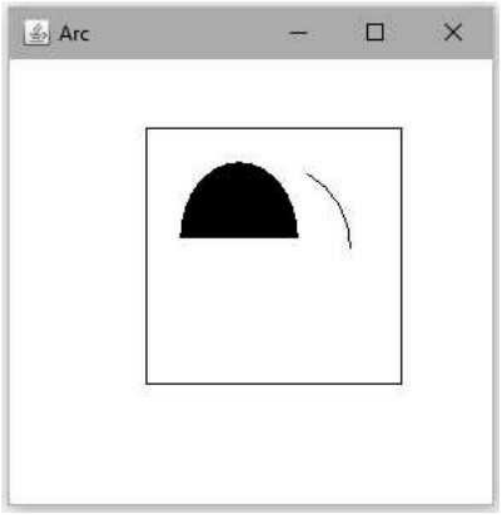
Output

Ex. 8.4.1 : Use graphics object method to draw an arc and semicircle inside a rectangular box AU : May-19, Marks 4

Sol. :

```
import java.awt.*;
class arcDemo extends Canvas
{
 public arcDemo()
 {
  setSize(200,200);
  setBackground(Color.white);
 }
 public static void main(String[] args)
 {
   arcDemo obj=new arcDemo();
   Frame fr=new Frame("Arc");
   fr.setSize(300,300);
   fr.add(obj);
   fr.setVisible(true);
 }
 public void paint(Graphics g)
 {
  g.drawArc(100,60,100,100,0,60);//arc
  g.fillArc(100,60,70,90,0,180);//semicircle
  g.drawRect(80,40,150,150);
 }
}
```

Output



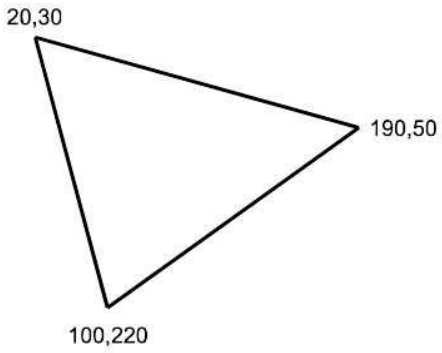
8.4.5 Polygons

In order to draw polygons following function is used

```
Polygon(int[ ] xpoints,int[ ] ypoints,int npoints)
```

The *xpoints* represent the array of *x* co-ordinates. The *ypoints* represent the array of y co-ordinates. And the npoints represents the number of points.

For filling up the polygon the **fillPolygon** method is used.



Java Program[Polyg.java]

```
import java.awt.*;
class Polyg extends Canvas
{
 public Polyg()
 {
  setSize(200,200);
```

```
  setBackground(Color.white);
 }
 public static void main(String[] args)
 {
  Polyg obj=new Polyg();
  Frame fr=new Frame("Polygon");
  fr.setSize(300,300);
  fr.add(obj);
  fr.setVisible(true);
 }
 public void paint(Graphics g)
 {
  int xpt[]={50,20,20,20,130};
 int ypt[]={80,30,200,200,30};
int num=5;
g.drawPolygon(xpt,ypt,num);
g.setColor(Color.magenta);
g.fillPolygon(xpt,ypt,num);
g.setColor(Color.black);
g.drawString("Its a polygon Demo",100,100);
 }
}
```

Output

Review Questions

1. *List the methods available in the draw shapes.* AU : CSE : Dec.-10, Marks 4
2. *Write the short note on - Graphics programming.* AU : IT : Dec.-11, Marks 8, CSE : Dec.-12, Marks 16
3. *How will you draw the following graphics in a window ? i) Arcs ii) Ellipses and circles in Java.* AU : IT : May-12, Marks 6
4. *With an example describe in detail about how to work with 2D Shapes in Java.* AU : May-15, Marks 8

8.5 Applet

- Applets are the small Java programs that can be used in internetworking environment.
- These programs can be transferred over the internet from one computer to another and can be displayed on various web browsers.
- Various **applications of applets** are in performing arithmetic operations, displaying graphics, playing sounds, creating animation and so on.
- Following are the **situations in which we need to use applet -**
  1. For displaying the dynamic web pages we need an applet. The dynamic web page is a kind of web page on which the contents are constantly changing. For example an applet that can represent the sorting process of some numbers. During the process of sorting the positions of all the elements is changing continuously.
  2. If we want some special effects such as sound, animation and much more then the applets are used.
  3. If we want that the particular application should be used by any user who might be located remotely . Then in such situation the applets are embedded into the web pages and can be transferred over the internet.

8.6 Life Cycle of Applet

IT : Dec.-12, Marks 16

There are various methods which are typically used in applet for initialization and termination purpose. These methods are

1. Initialization
2. Running state
3. Idle state
4. Dead or destroyed state (Refer Fig. 8.6.1. for applet's life cycle)

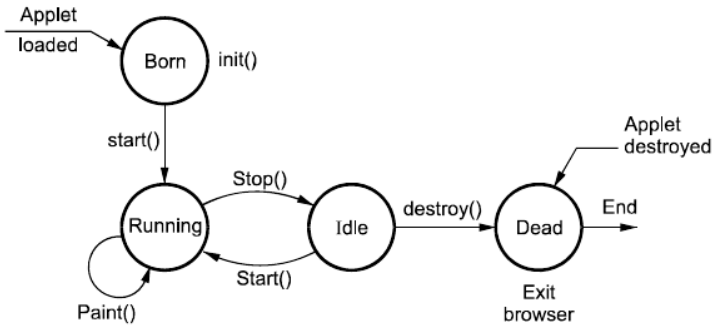


Fig. 8.6.1 Applet's life cycle

When applet begins, the AWT calls following methods in sequence -

a) init() b) start() c) paint()

When applet is terminated following methods are invoked in sequence.

a) stop() b) destroy()

1. Initialization state

When applet gets loaded it enters in the initialization state. For this purpose the **init()** method is used. In this method you can initialize the required variables. This method is called only once initially at the execution of the program. The syntax can be

```
public void init()
{
 //initialization of variables
}
```

In this method various tasks of initialization can be performed such as -

1. Creation of objects needed by applet
2. Setting up of initial values
3. Loading of image
4. Setting up of colors.

2. Running state

When the applet enters in the running state, it invokes the **start()** method of Applet class.

This method is called only after the init method. After stopping the applet when we restart the applet at that time also this method is invoked. The syntax can be

```
public void start()
{
 …
}
```

3. Display state

Applet enters in the display state when it wants to display some output. This may happen when applet enters in the running state. The **paint()** method is for displaying or drawing the contents on the screen. The syntax is

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

An instance of Graphics class has to be passed to this function as an argument. In this method various operations such as display of text, circle, line are invoked.

4. Idle state

This is an idle state in which applet becomes idle. The **stop()** method is invoked when we want to stop the applet .When an applet is running if we go to another page then this method is invoked. The syntax is

```
public void stop()
{
 …
}
```

5. Dead state

When applet is said to be dead then it is removed from memory. The method **destroy()** is invoked when we want to terminate applet completely and want to remove it from the memory.

```
public void destroy()
{
 …
}
```

It is a good practice to call stop prior to destroy method.

Review Question

1. *Explain about applet lifecycle ? How applets are prepared and executed.*

AU : IT : Dec.-12, Marks 16

8.7 Executing an Applet

There are two methods to run the applet

1. Using web browser 2 Using Appletviewer

Let us learn both the methods with necessary illustrations

1. Using web browser

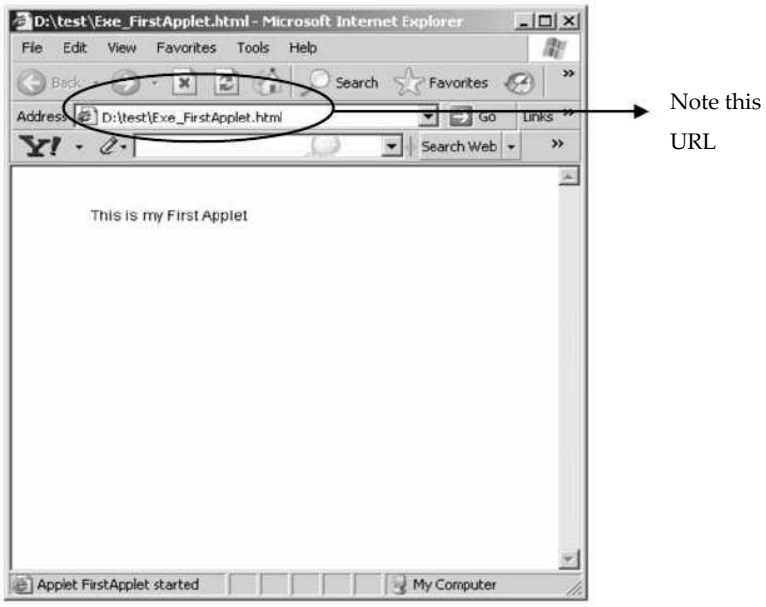
Step 1 : Compile your Applet source program using javac compiler, i.e.

```
D:\test>javac FirstApplet.java
```

Step 2 : Write following code in Notepad/Wordpad and save it with filename and extension **.html.** For example following code is saved as *Exe_FirstApplet.html,* The code is

```
<applet code="FirstApplet" width=300 height=100>
</applet>
```

Step 3 : Load html file with some web browser, This will cause to execute your html file. It will look this -



2. Using Appletviewer

Step 1 : To run the applet without making use of web browser or using command prompt we need to modify the code little bit. This modification is as shown below

```
/*
This is my First Applet program
*/
import java.awt.*;
import java.applet.*;
/*
<applet code="FirstApplet" width=300 height=100>
</applet>
*/
public class FirstApplet extends Applet
{
 public void paint(Graphics g)
 {
  g.drawString("This is my First Applet",50,30);
 }
}
```

In above code we have added one more comment at the beginning of the program
<applet code="FirstApplet" width=300 height=100>

```
</applet>
```

This will help to understand Java that the source program is an applet with the name FirstApplet. By this edition you can run your applet program merely by **Appletviewer** command.

Step 2 :

```
D:\test>javac FirstApplet.java
D:\test>Appletviewer FirstApplet.java
```

And we will get

Note one fact about applet is that: it does not require **main** function.

8.8 Using Color in Applet

AU : CSE : Dec.-10, 11, 13; IT : Dec.-13, Marks 16

When you want to specify the color in order to make your applet colourful, there are some methods supported by AWT system. The syntax of specifying color is

```
Color(int R,int G,int B);
Color(int RGBVal); //here RGBval denotes the value of color
```

Similarly to set the background color of applet window we use

```
void setBackground(Color colorname)
```

where *colorname* denotes the name of the background color. In the same manner we can also specify the foreground color i.e. color of the text by using method

```
void setForeground(Color colorname)
```

We can specify the *colorname* using the **Color** object as follows -

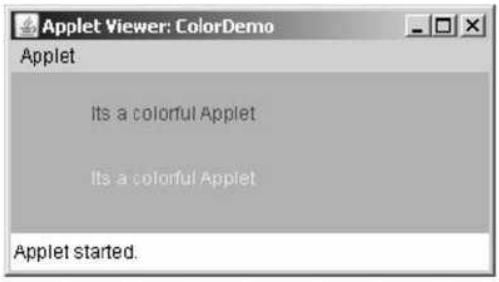
Color.black	Color.lightGray
Color.blue	Color.magenta
Color.cyan	Color.orange
Color.darkGray	Color.pink
Color.gray	Color.red
Color.green	Color.white
	Color.yellow

Here is a simple demo -

Java Program[ColorDemo.java]

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

/*
<applet code="ColorDemo" width=300 height=100>
</applet>
*/
public class ColorDemo extends Applet
{
 public void paint(Graphics g)
 {
  setBackground(Color.cyan);
  setForeground(Color.red);
  g.drawString("Its a colorful Applet",50,30);
  Color newColor=new Color(255,255,0);
   //creating red+green=yellow color
  g.setColor(newColor);
  g.drawString("Its a colorful Applet",50,70);
 }
}
```

Output



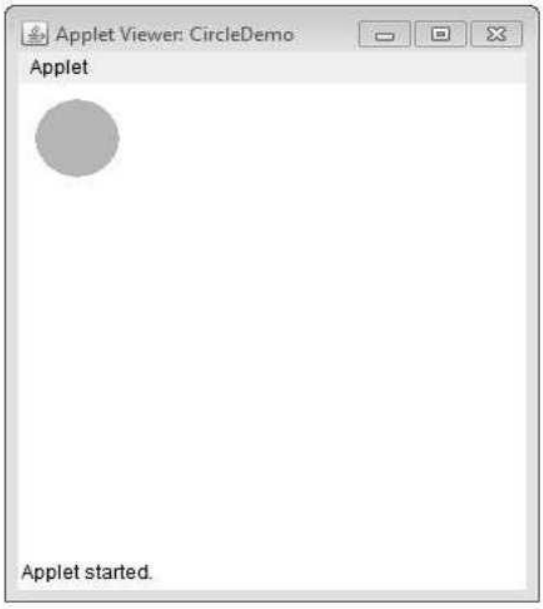
In above program we see one more method related to color and that is **setColor**. To set some specific color this method is used. The color name has to be passed as a parameter to this method. We can create our own color by a method **Color(int R,int G,int B)**. In above program we have created yellow color(Red + Green = Yellow) and then using *newColor* in **setColor** method the string *'It's a colourful applet'* is written on the new line.

Ex. 8.8.1 : Write an applet program for displaying the circle in green color.

Sol. :

```
import java.awt.*;
import java.applet.*;
/*
<applet code="CircleDemo" width=300 height=300>
</applet>
*/
public class CircleDemo extends Applet
{
  public void paint(Graphics g)
  {
   g.setColor(Color.green);
   g.fillOval(10,10,50,50);
  }
}
```

Output



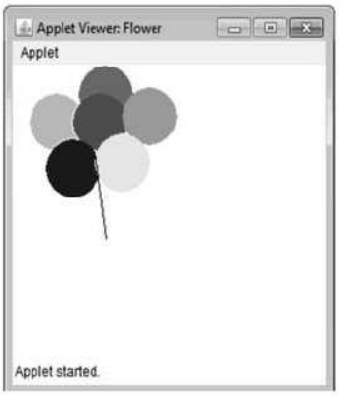
Ex. 8.8.2 : Write an applet program to draw an flower with color packages.

AU : CSE : Dec.-11, Marks 16

Sol. :

```
import java.awt.*;
import java.applet.*;
/*
<applet code="Flower" width=300 height=300>
</applet>
*/
public class Flower extends Applet
{
 public void paint(Graphics g)
 {
  g.setColor(Color.magenta);
  g.fillOval(60,1,50,50);

  g.setColor(Color.pink);
  g.fillOval(15,25,50,50);
  g.setColor(Color.red);
  g.fillOval(55,25,50,50);
  g.setColor(Color.blue);
  g.fillOval(30,65,50,50);
  g.setColor(Color.green);
  g.fillOval(100,20,50,50);
  g.setColor(Color.yellow);
  g.fillOval(75,60,50,50);
  g.setColor(Color.black);
  g.drawLine(75,75,85,150);

 }
}
```

Output



Ex. 8.8.3 : An analysis of examination results at a school gave the following distribution of grades for all subjects taken in one year :

Grade	*Percentage*
A	*10*
B	*25*
C	*45*
D	*20*

Write a java program to represent the distribution of each grade in a pie chart, where each slice of pie is differently colored.

AU : IT : Dec.-13, Marks 10

Sol. :

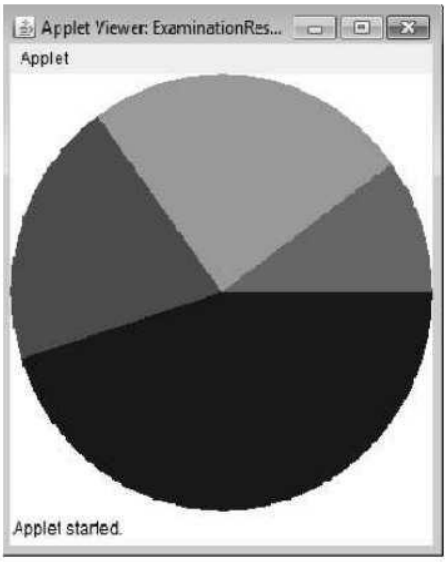
```
import java.awt.*;
import java.applet.*;
/*
<applet code="ExaminationResults" width=300 height=300>
</applet>
*/
class Slice
{
  double value;
  Color color;
  public Slice(double value, Color color)
  {
    this.value = value;
    this.color = color;
  }
}
public class ExaminationResults extends Applet
{
  Slice[] slices = {new Slice(10, Color.magenta),new Slice(25, Color.green),new Slice(20,
Color.red),new Slice(45, Color.blue)};

  public void paint(Graphics g)
  {
    drawPie((Graphics2D) g, getBounds(), slices);
  }

  void drawPie(Graphics2D g, Rectangle area, Slice[] slices)
  {
    double total = 0.0;
    for (int i = 0; i < slices.length; i++)
    {
      total += slices[i].value;
```

Storing grades and colors in an array

Drawing the piechart with each slice of different colors

```
    }
    double Value = 0.0;
    int startAngle = 0;
    for (int i = 0; i < slices.length; i++)
    {
      startAngle = (int) (Value * 360 / total);
      int arcAngle = (int) (slices[i].value * 360 / total);
      g.setColor(slices[i].color);
      g.fillArc(area.x, area.y, area.width, area.height,startAngle, arcAngle);
      Value += slices[i].value;
    }
  }
}
```

Output



Ex. 8.8.4 : Write a Java program to plot the path of small circle moving around the circumference of a larger circle.

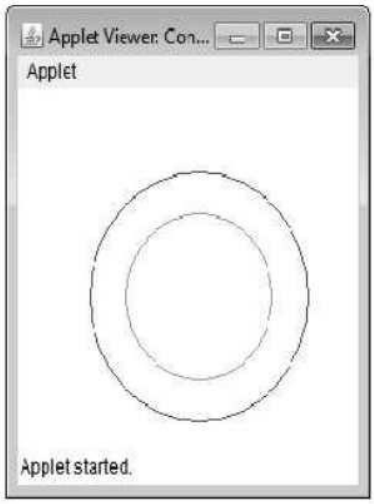
AU : IT : Dec.-13, Marks 16

Sol. :

```
import java.applet.*;
import java.awt.*;
/*<applet code="ConcCircles" height=250 width=300>
  </applet>
 */

public class ConcCircles extends Applet
{
```

```
        public void paint(Graphics g)
        {
                g.setColor(Color.RED);                  //Set the colour to red
                g.drawOval(50,50,150,150);
                g.setColor(Color.GREEN);                // Set the colour to green
                g.drawOval(75,75,100,100);              // 100 pixel diameter circle is drawn

        }
}
```

Output



Review Question

1. *List the methods available in the graphics for COLOR.*

AU : CSE : Dec.-10, Marks 4, Dec.-11, Marks 8

8.9 Using Fonts in Applet

AU : CSE : Dec.-11, Marks 8

We can obtain the list of all the fonts that are present in our computer with the help of **getAvailableFontFamilyNames()**. This method is member of **GraphicsEnvironment**. First of all we need a reference to **GraphicsEnvironment**. For obtaining reference to **GraphicsEnvironment** we can call the **getLocalGraphicsEnvironment( )** method. Then using this reference **getAvailableFontFamilyNames()** is invoked.

Following Java applet is for obtaining the list of available fonts

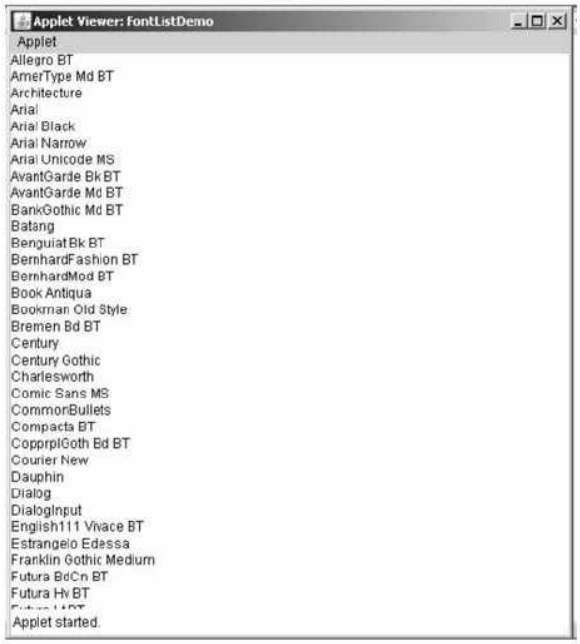
Java Program[FontListDemo.java]

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

```
/*
<applet code="FontListDemo" width=500 height=500>
</applet>
*/
public class FontListDemo extends Applet
{
 public void paint(Graphics g)
 {
  String Font_names="";
  String F[];
  int k=0;
//getting reference to GraphicsEnvironment
 GraphicsEnvironment ge =
GraphicsEnvironment.getLocalGraphicsEnvironment();
  F=ge.getAvailableFontFamilyNames();
  for(int i=0;i<F.length;i++)
  {
   Font_names=F[i];
   g.drawString(Font_names,0,10+k);
   k=k+15;
  }
 }
}
```

To print the message on next line y-co-ordinate is incremented.

Output

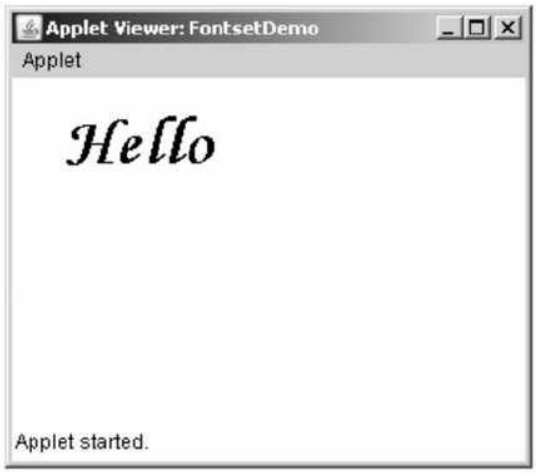


We can set the desired font using **setFont()** function. This function requires a reference parameter which can be created by the method **Font()**. In **Font()** method the first parameter is name of the font, second parameter denotes style of the font which can BOLD, ITALIC or PLAIN and third parameter denotes the size of font. Here is a simple program which is demonstrate the same

Java Program[FontsetDemo.java]

```
import java.awt.*;
import java.applet.*;
/*
<applet code="FontsetDemo" width=310 height=200>
</applet>
*/
public class FontsetDemo extends Applet
{
 String msg=" ";
 public void init()
 {
  Font f;
  f=new Font("Monotype Corsiva",Font.BOLD,40);
  msg="Hello";
  setFont(f);
 }
 public void paint(Graphics g)
 {
  g.drawString(msg,30,50);
 }
}
```

Output

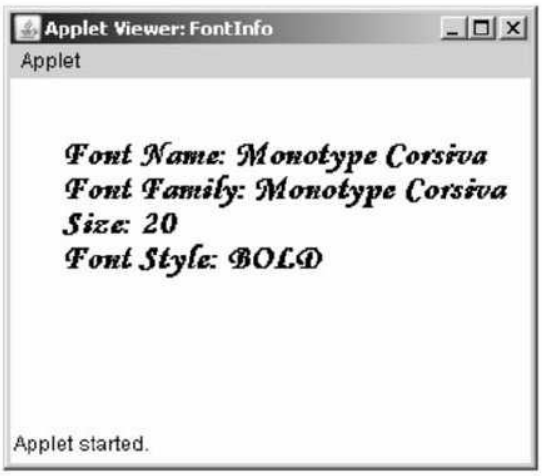


Similarly we can get the information about the font. In the following Java program in the **init** method we have set a specific font and in the **paint** method we can get the font information using **getFont()** method.

Java Program[FontInfo.java]

```
import java.awt.*;
import java.applet.*;
/*
<applet code="FontInfo" width=300 height=200>
</applet>
*/
public class FontInfo extends Applet
{
 String msg=" ";
  Font f;
 public void init()
 {
  f=new Font("Monotype Corsiva",Font.BOLD,20);
  setFont(f);
 }
 public void paint(Graphics g)
 {
   f=g.getFont();
   String name=f.getName();
   msg="Font Name: "+name;
   g.drawString(msg,30,50);
   String family=f.getFamily();
   msg="Font Family: "+family;
   g.drawString(msg,30,70);
   int size= f.getSize();
   msg="Size: "+size;
   g.drawString(msg,30,90);
   int style=f.getStyle();//we int value of font Style
   if((style & Font.PLAIN)==Font.PLAIN)
     msg="Font Style: PLAIN";
   if((style & Font.BOLD)==Font.BOLD)
     msg="Font Style: BOLD";
   if((style & Font.ITALIC)==Font.ITALIC)
     msg="Font Style: ITALIC";
   g.drawString(msg,30,110);
 }
}
```

Output



Review Question

1. *Explain the usage of special fonts for text in Graphics programming.*

AU : CSE : Dec.-11, Marks 8

8.10 Using Images in Applet

- The image can be displayed in applet using **drawImage** method of **Graphics** class. Before using the **drawImage** we must get the image file. This can be obtained using the method **getImage.**
- **Syntax**

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer)

```
public Image getImage(URL u, String image)
{
}
```

- **Example**

```
import java.awt.*;
import java.applet.*;
/*
<applet code="ImageDisplayDemo" width=300 height=300>
</applet>
*/
public class ImageDisplayDemo extends Applet
{
        Image Img;
        public void init()
        {
                Img = getImage(getDocumentBase(),"myimg.jpg");
```

```
        }
        public void paint(Graphics g)
        {
                g.drawImage(Img, 50,50, this);
        }
}
```

Review Question

1. *Explain how to display an image in applet ?*

Two Marks Questions with Answers

Q.1 What are the steps needed to show a Frame ?

AU : CSE : Dec.-10

OR How are frames created in Java ?

AU : CSE : Dec.-11

Ans. : Frame can be created by following two ways -

1) The frame can be created by extending the frame class. For example

```
class FrameDemo extends Frame
{
 public void static main(String[] arg)
 {
    FrameDemo fr=new FrameDemo();
    fr.setSize(300,300);
    fr.setVisible(true);
 }
}
```

2) The frame can be created by using the instance of Frame class -

```
class FrameDemo1
{
 public static void main(String[] arg)
 {
  Frame fr=new Frame();
  fr.setSize(300,300);
  fr.setVisible(true);
 }
}
```

Q.2 What is meant by frame window ?

AU : IT : May-12

Ans. : The frame window is a standard graphical window. It can be displayed by the class named **Frame.** This frame window has standard minimize, maximize and close buttons.

Q.3 What is AWT ?

AU : IT : May-12, Dec.-12

Ans. : The AWT stands for Abstract Window Toolkit. It is a class library that contains various classes and interfaces that are required for graphical programming. The AWT provides the support for drawing graphical shapes, windows, buttons, text boxes, menus and so on.

Q.4 How do you manage the color and font of graphics in applet ?

AU : CSE : May-12

Ans. : The color can be specified as

```
color(int R,int G,int B);
```

The background color can be set by

```
setBackground(Color colorname)
```

The color of the text can be set by

```
setForeground(Color colorname)
```

The font can be set by **setFont()** method. This function requires a reference parameter which can be created by the method **Font().** The Font() method has first parameter as name of the font, second parameter denotes the style of the font which can be BOLD, ITALIC or PLAIN. The third parameter denotes the size of the font.

Q.5 Code a Graphics method in Java to draw the String "Hello World" from the coordinates(100,200)

AU : IT : Dec.-13

Ans. :

```
Import java.applet;
/*
<applet code="MsgDemo" width=300 height=300>
</applet>
*/
public class MsgDemo extends Applet

{
 public void paint(Graphics g)
 {
  g.drawString("Hello World",100,200);
 }
}
```

Q.6 How does radio button in Java differ from check box ?

AU : IT : Dec.-13

Ans. : The radio buttons are mutually exclusive and user must select exactly one choice from the given list. Whereas the check box is for selecting any number options from the given choices.

Q.7 Enumerate features of AWT in Java

AU : CSE : Dec.-18

Ans. : 1) AWT stands for Abstract Window Toolkit. It includes large number of classes to include **graphical components** such as text box, buttons, labels and radio buttons and so on.

2) AWT classes allows to **handle the events** associated with graphical components.

Q.8 Write the class hierarchy for Panel and Frame AU : May 19

Ans. : Refer fig.8.1.1

Q.9 State the purpose of getRed(), getBlue(), and getGreen() methods

AU : May 19

Ans. : The gertRed(), getBlue() and getGreen() returns the red component, blue component or green component of color in the range 0.0 – 1.0

❑❑❑