Java Programming - II
Unit-1 Graphics Programming

Frame:

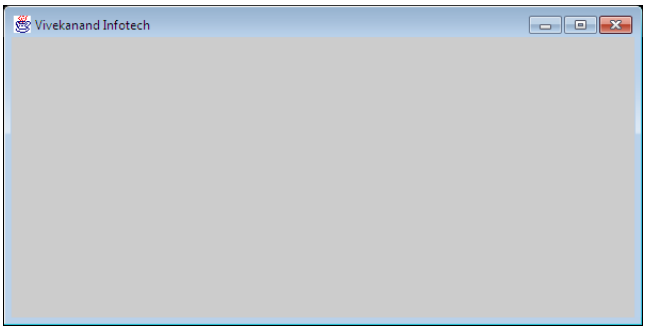
A top-level window (a window that is not contained inside another window) is called a Frame in Java. The AWT library has a class, called **Frame**. The Swing version of this class is called **JFrame**. JFrame extends the Frame class and is one of the few Swing components.

Frames are examples of containers. This means that a frame can contain other user interface components such as buttons and text fields. The java.awt.Frame component is a Windows graphics system which has a title bar and borders, behaving like a normal GUI window.

When working with Frame objects, the following steps are basically followed to get a window to appear on the screen.

1) Create an object of type Frame.
2) Give the Frame object a size using setSize () method.
3) Make the Frame object appear on the screen by calling setVisible () method.
4) In order to close the window by clicking the close(X) button, you will have to insert the code for window closing event.

```
import javax.swing.*;
import java.awt.*;
class myframe extends JFrame
{
        myframe()
        {
                setSize(600,300);
                setTitle("Vivekanand Infotech");
                setLocation(200,100);
        }
}
class Prog1
{
        public static void main(String args[])
        {
                myframe m=new myframe();
                m.show();
                m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```



Displaying information in a frame:

Now we will see how to display information inside a frame. It is possible to draw the message string directly onto a frame, but that is not considered good programming. Frames are really designed to be containers for components such as a menu bar and other user interface elements. We normally draw on another component called a panel, which we add to the frame.

The structure of a JFrame is surprisingly complex. It is four panes are layered in a JFrame. The root pane, layered pane & glass pane are of the no interest to us. They are required to organize the menu bar. We focus on content pane to implement the look & feel method. When designing a frame, we add components into the content pane, by using following code-

```
mypanel mp=new mypanel();
Container cp=getContentPane();
cp.add(mp);
```

To add a simple panel to the content pane on to which we will draw our message. Panels are implemented by the JPanel class.

Graphics objects & Paint component method:

The **PaintComponent()** method is actually in JComponent the parent class for all non windows Swing components. It takes on parameter of the type Graphics. A graphics objects remembers a collection of setting for drawing images & text, such as the font we set or the current color. All drawing in Java must go through a Graphics object. It has methods that draw patterns, images & text. The **PaintComponent()** method is called automatic whenever a part of your application needs to be redrawn.

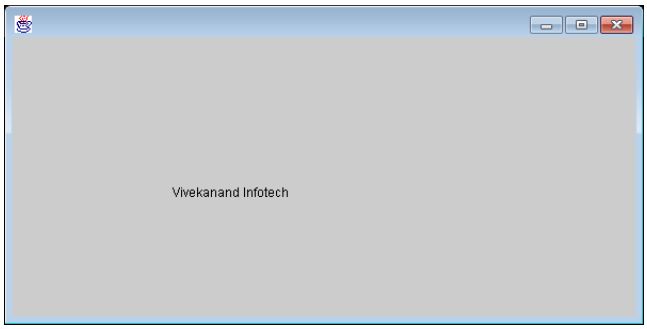
To displaying the text, is considered a special kind of drawing. The Graphics class has a drawString() method that has the following syntax-

g.drawString(text, x, y);

e.g.: g.drawString("Vivekanand", 100, 150);

Program:

```
import javax.swing.*;
import java.awt.*;
class mypanel extends JPanel
{
        public void paintComponent(Graphics g)
        {
                super.paintComponent(g);
                g.drawString("Vivekanand Infotech",150,150);
        }
}
class myframe extends JFrame
{
    myframe()
    {
        setSize(600,300);
        setLocation(300,300);
        mypanel mp=new mypanel();
        Container cp=getContentPane();
        cp.add(mp);
    }
}
class prog2
{
    public static void main(String args[])
    {
        myframe m=new myframe();
        m.show();
        m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



Text and Fonts:

We want to show text in a different font. We specify a font by its font face name. A font face name is compared of a font family name, such as "Arial". These font names are always mapped to fonts that actually exist on the client machine. To draw characters in a font, we must first create an object of Font class, then specify the font name, the font style and the font size.

e.g.: Font f1 = new Font("Arial", Font.BOLD, 14);

We can use the logical font name in the place of a font face name as first parameter. We specify the style as follows as second parameter. The third parameter is the font size.

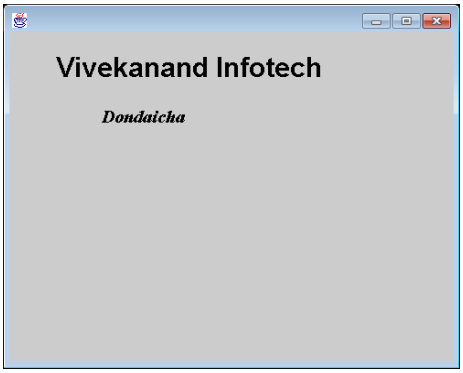
Styles: **Font.PLAIN, Font.BOLD, Font.ITALIC, Font.BOLD+Font.ITALIC**

Program:

```
import java.io.*;
import javax.swing.*;
import java.awt.*;
class mypanel extends JPanel
{
	public void paintComponent(Graphics g)
	{
		super.paintComponent(g);

		Font f1=new Font("Arial",Font.BOLD,30);
		g.setFont(f1);
		g.drawString("Vivekanand Infotech",50,50);

		Font f2=new Font("Times New Roman",Font.BOLD+Font.ITALIC,20);
		g.setFont(f2);
		g.drawString("Dondaicha",100,100);
	}
}
class myframe extends JFrame
{
	myframe()
	{
		setSize(500,400);
		setLocation(150,150);
		mypanel mp=new mypanel();
		Container cp=getContentPane();
		cp.add(mp);
	}
}
class prog3
{
	public static void main(String args[])
	{
		myframe m=new myframe();
		m.show();
		m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
}
```



Colors:

The **setPaint()** method of the **Graphics2D** class use for select a color that is used for all subsequent drawing contents on the graphics context or component.

We define a color with the Color class. The **java.awt.Color** class offers pre-defined constants for the 13 standard colors listed in below –

black	green	red	blue
lightgray/darkgray	white	cyan	magenta
yellow	orange	gray	pink

e.g.:
```
g2.setPaint(Color.red);
g2.drawString("Vivekanand", 50, 50);
```

We can specify a custom color by creating a color object by its red, green and blue components. Using a scale of 0-255 for redness, blueness and greenness.

e.g.: g2.setPaint(new Color(255,0,0));
g2.drawString("Vivekanand", 50, 50);

To set background color, we use the **setBackground()** method of the Component class.

e.g.: setBackground(Color.yellow);

Program:

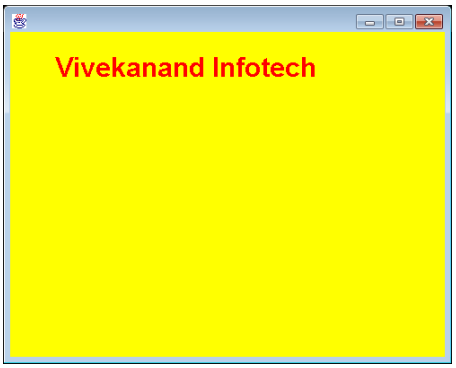
```
import javax.swing.*;
import java.awt.*;
class mypanel extends JPanel
{
        public void paintComponent(Graphics g)
        {
                Graphics2D g2=(Graphics2D)g;
                super.paintComponent(g2);

                setBackground(Color.yellow);

                Font f1=new Font("Arial",Font.BOLD,30);
                g2.setFont(f1);

                //g2.setPaint(Color.blue);
                g2.setPaint(new Color(255,0,0));
                g2.drawString("Vivekanand Infotech",50,50);
        }
}
class myframe extends JFrame
{
        myframe()
        {
                setSize(500,400);
                setLocation(150,150);

                mypanel mp=new mypanel();
                Container cp=getContentPane();
                cp.add(mp);
        }
}
class prog4
{
        public static void main(String args[])
        {
                myframe m=new myframe();
                m.show();
                m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```



Drawing Shapes:

The Java2D library which implements a very powerful set graphical operation. To draw shapes in the Java2D library, we need to obtain an object of the Graphics2D class. This is a subclass of the Graphics class. If we use a version of the SDK that is Java2D enabled, methods such as painComponent() automatically receive an object of the Graphics 2D class. Simply use a cast, as follows –

```
public void paintComponent(Graphics g)
{
        Graphics2D  g2 = (Graphics2D)g;
}
```

The Java2D library organizes geometric shapes in an object oriented fashion. There are classes to represent lines, rectangle and ellipse –

Line2D, Rectangle2D, Ellipse2D

1) Draw Line:

To draw a line, we supply the start and end points as pair of numbers.

Syntax: Line2D line = new Line2D.Double(startx, starty. endx, endy);

e.g.: Line2D line = new Line2D.Double(100.0,130.0,450.0,350.0);

2) Rectangle Shape:

The Rectangle2D class is an abstract class with two concrete subclasses, which are also static inner classes –

Rectangle2D.Float

Rectangle2D.Double

Both Rectangle2D.Float and Rectangle2D.Double class extends the common Rectangle2D class & the methods is the subclass simply override methods in the Rectangle2D super class.

When we construct a Rectangle2D.Float object, we supply the co-ordinates as float number. For a Rectangle2D.Double object, we supply then as double numbers.

e.g.: Rectangle2D.Float frect = new Rectangle2D.Float(100.0F,130.0F,450.0F,350.0F);

Rectangle2D.Double drect = new Rectangle2D.Double(100.0,130.0,450.0,350.0);

The constructor parameters denote the top-left corner, width & height of the rectangle.

3) Ellipse Shape:

To draw the shape of the ellipse we have to use Ellipse2D class, which are inherit from a common super class Rectangular shape. The ellipse draw process is same as rectangle shape but not a rectangle, only the have a bounding rectangle.

Ellipse2D object are simple to construct. We need to specify, the x & y co-ordinates of the top-left corner, width & height.

e.g.: Ellipse2D elp=new Ellipse2D.Double(200.0,150.0,200.0,100.0);

Filling Shapes:

We can fill the interiors of closed shapes (such as rectangle or ellipses) with a color or more generally the paint setting by using fill instead of draw.

```
e.g.: g2.setPaint(Color.red);
      Ellipse2D elp=new Ellipse2D.Double(200.0,150.0,200.0,100.0);
      g2.fill(elp);
```

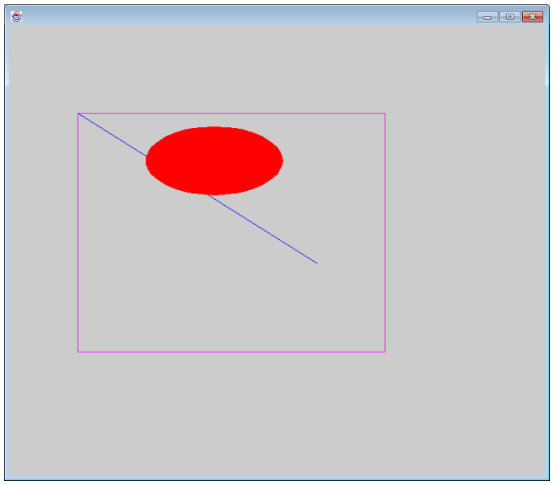
Program:

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;
class mypanel extends JPanel
{
        public void paintComponent(Graphics g)
        {
                Graphics2D g2=(Graphics2D)g;
                super.paintComponent(g2);
```

```
                g2.setPaint(Color.blue);
                Line2D line=new Line2D.Double(100.0,130.0,450.0,350.0);
                g2.draw(line);

                g2.setPaint(Color.magenta);
                Rectangle2D.Double rect=new Rectangle2D.Double(100.0,130.0,450.0,350.0);
                g2.draw(rect);

                g2.setPaint(Color.red);
                Ellipse2D elp=new Ellipse2D.Double(200.0,150.0,200.0,100.0);
                g2.fill(elp);
                //g2.draw(elp);
        }
}
class myframe extends JFrame
{
        myframe()
        {
                setSize(800,700);
                setLocation(150,150);
                mypanel mp=new mypanel();
                Container cp=getContentPane();
                cp.add(mp);
        }
}
class prog5
{
        public static void main(String args[])
        {
                myframe m=new myframe();
                m.show();
                m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```



Paint Mode:

There are two painting or drawing modes for the Graphics class:- paint mode which is the default mode and XOR mode. In paint mode, anything we draw replaces whatever is already on the screen. i.e. it overwrites any preexisting contents. If you draw a green circle, you get a green circle, no matter what was underneath. The paint mode determine how objects are drawn in a window. It is possible to have new objects XORed on to the window by using setXORMode().

Syntax: **void setXORMode(Color XORColor)**

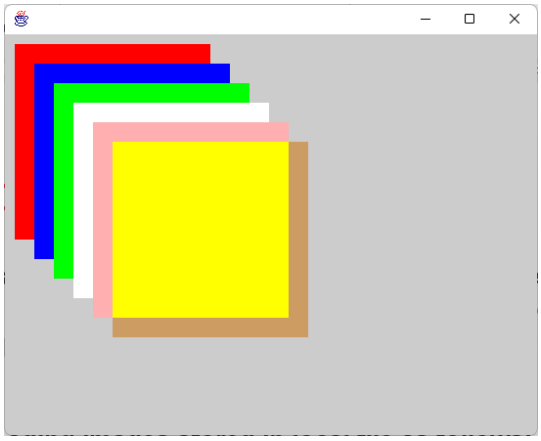
XORColor:- color that will be XORed to the window when an object is drawn.

The advantage of XORMode is that the new object is always guaranteed to be visible no matter what color the object is drawn over.

Program:

```
import java.awt.*;
import javax.swing.*;
class mypanel extends JPanel
{
        public void paintComponent(Graphics g)
        {
                        g.setColor(Color.red);
```

```
                    g.fillRect(10,10,200,200);
                    g.setColor(Color.blue);
                    g.fillRect(30,30,200,200);
                    g.setColor(Color.green);
                    g.fillRect(50,50,200,200);
                    g.setColor(Color.white);
                    g.fillRect(70,70,200,200);
                    g.setColor(Color.pink);
                    g.fillRect(90,90,200,200);
                    g.setXORMode(Color.yellow);
                    g.fillRect(110,110,200,200);
        }
}
class myFrame extends JFrame
{
        myFrame()
        {
                setSize(800,600);
                setLocation(100,100);
                mypanel mp=new mypanel();
                Container cp=getContentPane();
                cp.add(mp);
        }
}
class Prog07PaintMode
{
        public static void main(String args[])
        {
                myFrame m1= new myFrame();
                m1.show();
                m1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }
}
```



Images:

We have already seen how to create simple image by drawing line & shapes. Complex images such as photographs are generated externally & stored on hard disk or internet. We can read these complex images to the java application and display them on graphics object.

Reading images stored in local file as follows:

```
String fname="D:/vivek.jpg";
img=Toolkit.getDefaultToolkit().getImage(fname);
```

Reading images store in URL as follows:

```
String fname="http://dadasahebrawalcollege.ac.in/Images/2018-10-02
                %20Prize%20Disribution/IMG_20181003_135234.jpg";
URL uname=new URL(fname);
img=Toolkit.getDefaultToolkit().getImage(uname);
```

Here getImage() method throws I/O exception, if the image is not available. So, it better to handle the exception using try & catch block. The image store in img in above example can be drawn by calling **drawImage()** method as follows:

Syntax: g.drawImage(img, x1, y1, x2, y2, observer);

The observer is an object notifying the progress of process usually, It is set to null.

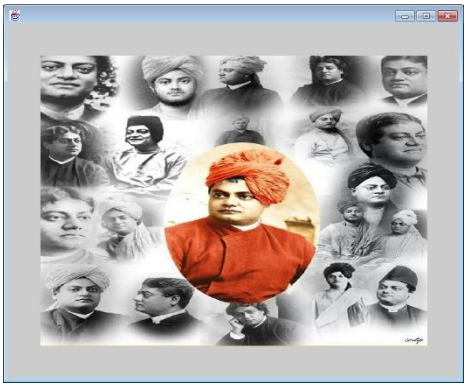
e.g.: g.drawImage(img,50,50,600,450,null);

Program:

```
import javax.swing.*;
import java.awt.*;
import java.net.*;
class mypanel extends JPanel
{
	private Image img;
	mypanel()
	{
		setLayout(null);
		try
		{
			// Reading images stored in local file
			String fname="D:/vivek.jpg";
			img=Toolkit.getDefaultToolkit().getImage(fname);

			// Reading images store in URL
			//String fname = "http://dadasahebrawalcollege.ac.in/Images/
					2018-10-02%20Prize%20Disribution/IMG_20181003_135234.jpg";
			//URL uname=new URL(fname);
			//img=Toolkit.getDefaultToolkit().getImage(uname);
		}
		catch(Exception e)
		{
			System.out.println(e);
		}
	}
	public void paintComponent(Graphics g)
	{
		super.paintComponent(g);
		g.drawImage(img,50,50,600,450,null);
	}
}
class myframe extends JFrame
{
	myframe()
	{
		setSize(800,700);
		setLocation(150,150);
		mypanel mp=new mypanel();
		Container cp=getContentPane();
		cp.add(mp);
	}
}
class prog6
{
	public static void main(String args[])
	{
		myframe m=new myframe();
		m.show();
		m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	}
}
```

Reading images stored in local file:



Reading images store in URL: