effect for the window determines how the components are sized and positioned inside the window.

List of Layout Managers

The java.awt package provides the following predefined layout managers that implement the java.awt.LayoutManager interface. Every Abstract Window Toolkit (AWT) and Swing container has a predefined layout manager as its default. It is easy to use the container.setLayout method to change the layout manager, and you can define your own layout manager by implementing the `java.awt.LayoutManager` interface. This article describes the predefined AWT layout managers in this list.

- java.awt.BorderLayout
- java.awt.FlowLayout
- java.awt.CardLayout
- java.awt.GridLayout
- java.awt.GridBagLayout

BorderLayout

The border layout is the default layout manager for all Window objects. It consists of five fixed areas: North, South, East, West, and Center. You do not have to put a component in every area of the border layout. The demonstration puts the label "Border Layout" in the North area, the OK button in the South area, and the List of fruit trees in the Center area. The East (right) and West (left) areas are empty.

FlowLayout

The flow layout is the default layout manager for all Panel objects and applets. It places the panel components in rows according to the width of the panel and the number and size of the components. FlowLayout implements a simple layout style, which is similar to how words flow in a text editor. Components are laid out from the upper-left corner, left to right and top to bottom. When no more components fit on a line, the next one appears on the next line.

GridLayout

The Grid layout arranges components into a grid of rows and columns. we specify the number of rows and columns, the number of rows only and let the layout manager determine the numer of columns, or the number of columns only and let the layout manager determine the number of rows.

GRAPHICS CONTEXTS AND GRAPHICS OBJECTS

A Graphics object manages a graphics context by controlling how objects are drawn. Graphics objects contain methods for drawing, font manipulation, color manipulations, and the other related operations. It has been developed using the Graphics object g (the argument to the applet's paint method) to manage the applet's graphics context. In other words, wou can say that Java's Graphics is capable of:

• Drawing 2D shapes
• Controlling colors

• Controlling fonts
• Providing Java 2D API
• Using More sophisticated graphics capabilities
• Drawing custom 2D shapes
• Filling shapes with colors and patterns.

Graphics Context and Graphics Class

• Enables drawing on screen
• Graphics object manages graphics context
• Controls how objects are drawn
• Class Graphics is abstract
• Cannot be instantiated
• Contributes to Java's portability
• Class Component method paint takes Graphics object.

The `Graphics` class is the abstract base class for all graphics contexts. It allows an application to draw onto components that are realized on various devices, as well as on to off-screen images.

Graphics Objects
In Java, all drawing takes place via a `Graphics` object. This is an instance of the class `java.awt.Graphics`.
Initially the Graphics object we use will be passed as an argument to an applet's paint() method. The drawing can be done by using Applet Panels, Frames, Buttons, Canvases, etc.
Each `Graphics` object has its own coordinate system, and methods for drawing strings, lines, rectangles, circles, polygons, etc. Drawing in Java starts with particular `Graphics` object. we get access to the Graphics object through the *paint(Graphics g)* method of your applet.
Each draw method call will look like
g.drawString("Hello World", 0, 50);
where g is the particular Graphics object with which you're drawing.

Color Control

It is known that Color enhances the appearance of a program and helps in conveying meanings. To provide color to our objects use class Color, which defines methods and constants for the manipulation of colors. Colors are created from **red**, **green** and **blue** components **RGB** values.

All three RGB components can be integers in the range 0 to 255, or floating point values in the range 0.0 to 1.0
The first part defines the amount of *red*, the second defines the amount of *green* and the third defines the amount of *blue*. So, if we want to give a dark red color to our graphics we will have to give the first parameter value 255 and two parameter zero.

Some of the most common colors are available by name and their RGB values.

Table 1: Colors and their RGB values Color Constant	Color	RGB Values
Public final static Color ORANGE	Orange	255, 200, 0
Public final static Color PINK	Pink	255, 175, 175
Public final static Color CYAN	Cyan	0, 255, 255
Public final static Color MAGENTA	Magenta	255, 0, 255
Public final static Color YELLOW	Yellow	255, 255, 0
Public final static Color BLACK	Black	0, 0, 0
Public final static Color WHITE	White	255, 255, 255
Public final static Color GRAY	Gray	128, 128, 128
Public final static Color LIGHT_GRAY	light gray	192, 192, 192
Public final static Color DARK_GRAY	dark gray	64, 64, 64
Public final static Color RED	Red	255, 0, 0
Public final static Color GREEN	Green	0, 255, 0
Public final static Color BLUE	Blue	0, 0, 255

As we do with any variable, we should preferably give our colors descriptive names. For instance

```
Color medGray = new Color(127, 127, 127);
Color cream = new Color(255, 231, 187);
Color lightGreen = new Color(0, 55, 0);
```

we should note that Color is not a property of a particular rectangle, string or other object we may draw. Color is a part of the `Graphics` object that does the drawing. we change the color of our `Graphics` object and everything we draw from that point forward will be in the new color, at least until we change it again.

When an applet starts running , its color is set to *black by default*. we can change this to red by calling `g.setColor(Color.red)`. we can change it back to black by calling `g.setColor(Color.black)`.

Fonts

Unlike HTML Java allows you to choose your fonts. Java implementations are guaranteed to have a *serif font* like *Times* that can be accessed with the name *"Serif"*, a monospaced font like *courier* that can be accessed with the name *"Mono",* and a *sans serif* font like *Helvetica* that can be accessed with the name *"SansSerif"*.

GridBagLayout

The Grid bag layout (like grid layout) arranges components into a grid of rows and columns, but lets you specify a number of settings to fine-tune how the components are sized and positioned within the cells. Unlike the grid layout, the rows and columns are not constrained to be a uniform size. For example, a component can be set to span multiple rows or columns, or you can change its position on the grid.

GridBagConstraints gbc = new GridBagConstraints();

gridx and gridy

The `gridx` and `gridy` fields specify the x and y coordinates of the cell at the upper left of the `Component's` display area. The upper-left-most cell has coordinates (0, 0). The mnemonic constant `GridBagConstraints.RELATIVE` specifies that the Component is placed immediately to the right of (gridx), or immediately below `(gridy)` the previous Component added to this container.

Gridwidth and Gridheight

The `gridwidth` and `gridheight` fields specify the number of cells in a row `(gridwidth)` or column `(gridheight)` in the Component's display area. The mnemonic constant `GridBagConstraints.REMAINDER` specifies that the `Component` should use all remaining cells in its row `(for gridwidth)` or column (for `gridheight`). The mnemonic constant `GridBagConstraints`.RELATIVE specifies that the Component should fill all but the last cell in its row `(gridwidth)` or column `(gridheight)`.

Fill

The `GridBagConstraints` `fill` field determines whether and how a component is resized if the component's display area is larger than the component itself. The mnemonic constants you use to set this variable are
`GridBagConstraints.NONE` : Don't resize the component
`GridBagConstraints.HORIZONTAL:` Make the component wide enough to fill the display area, but don't change its height.
`GridBagConstraints.VERTICAL:` Make the component tall enough to fill its display area, but don't change its width.
`GridBagConstraints.BOTH:` Resize the component enough to completely fill its display area both vertically and horizontally.

Ipadx and Ipady

Each component has a minimum width and a minimum height, smaller than which it will not be. If the component's minimum size is smaller than the component's display area, then only part of the component will be shown.

The `ipadx` and `ipady` fields let you increase this minimum size by padding the edges of the component with extra pixels. For instance setting `ipadx` to 2 will guarantee that the component is at least 4 pixels wider than its normal minimum. (`ipadx` adds 2 pixels to each side.)

Insets

The `insets` field is an instance of the `java.awt.Insets` class. It specifies the padding between the component and the edges of its display area.

weightx and weighty

The `weightx` and `weighty` fields determine how the cells are distributed in the container when the total size of the cells is less than the size of the container. With weights of zero (the default), the cells all have the minimum size they need, and everything clumps together in the center. All the extra space is pushed to the edges of the container.

Swing

The Swing toolkit includes a rich set of components for building GUIs and adding interactivity to Java applications. Swing includes all the components you would expect from a modern toolkit: table controls, list controls, tree controls, buttons, and labels.

Swing is part of the Java Foundation Classes (JFC). The JFC also include other features important to a GUI program, such as the ability to add rich graphics functionality and the ability to create a program that can work in different languages and by users with different input devices.
The following list shows some of the features that Swing and the Java Foundation Classes provide.

Swing GUI Components

The Swing toolkit includes a rich array of components: from basic components, such as buttons and check boxes, to rich and complex components, such as tables and text. Even deceptively simple components, such as text fields, offer sophisticated functionality, such as formatted text input or password field behavior. There are file browsers and dialogs to suit most needs, and if not, customization is possible. If none of Swing's provided components are exactly what you need, you can leverage the basic Swing component functionality to create your own.

Pluggable Look-and-Feel Support

Any program that uses Swing components has a choice of look and feel. The JFC classes shipped by Sun and Apple provide a look and feel that matches that of the platform. The Synth package allows you to create your own look and feel. The GTK+ look and feel makes hundreds of existing look and feels available to Swing programs.

Difference between AWT and Swing

AWT - Heavy weight component. Every graphical units it will invoke native methods.
SWING - Light weight component. It doesn't invoke native methods.

Swing is based on MVC architetcure(Model view Controller) and thats why its look and feel is independent of hardware and OS...whereas AWT look and feel depends upon the platform...

Swing are the extension of Awt.
Awt have some disadvanage that are removed by he swing .for example
awt programs are not machine independent.
it causes some problem on different machine
whereas his does not occur in swing.

Swing provides some advanced compoent like JTree JTabbedPane JList and many more

Swing GUI components are packaged into Package javax.swing. In the Java class hierarchy there is a class
Class Component which contains a method paint for drawing Component onscreen
Class Container which is a collection of related components and contains method add for adding components and Class JComponent which has
Pluggable look and feel for customizing look and feel
Shortcut keys (*mnemonics*)
Common event-handling capabilities
The Hierarchy is as follows:

Object----- Component-- Container--- JComponent

In Swings, we have classes prefixed with the letter 'J' like

JLabel -> Displays single line of read only text

JTextField -> Displays or accepts input in a single line

JTextArea -> Displays or accepts input in multiple lines

JCheckBox ->Gives choices for multiple options

JButton -> Accepts command and does the action

JList - > Gives multiple choices and display for selection

JRadioButton - > Gives choices for multiple option, but can select one at a time.

Registration.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
class Registration1 extends JFrame implements ActionListener,ItemListener
{
public Registration1()
        {
        display();
        }
        JPanel        panel;
```

```
        JLabel          label1,label2,label3,label4,label5,label6,
                      label7,label8,label9,label10,label11,
                      label12,
                       label13,label14,label15,label16;
        JTextField     text1,text2,text3,text4,text5,text6,text7,
                       text8,text9,text10,text11,text12,text13,
                       text14;
        JButton                bt1,bt2,bt3,bt4,bt5;
        JComboBox  cb1,cb2;
        public void display()
                {
                setTitle("Registration From...");
                setSize(1024,600);
                setVisible(true);
                panel=new JPanel();
                getContentPane().add(panel);
                panel.setLayout(null);
//--------------------------PATIENT ID-------------------------------------
label1=new JLabel();
label1.setText("Patient ID");
panel.add(label1).setBounds(100,50,160,35);
label1.setFont(new Font("courier new",Font.BOLD,18));
label1.setForeground(new Color(50,48,20));
//--------------------------TEXT1-------------------------------------
text1=new JTextField(20);
text1.setBackground(new Color(10,255,160));
panel.add(text1).setBounds(300,50,150,30);
//--------------------------REG...CATAGARY-------------------------------------
label2=new JLabel();
label2.setText("Reg...Catagary");
panel.add(label2).setBounds(100,85,160,35);
label2.setFont(new Font("courier new",Font.BOLD,18));
label2.setForeground(new Color(50,48,20));
//--------------------------TEXT2-------------------------------------
text2=new JTextField(20);
text2.setBackground(new Color(10,255,160));
panel.add(text2).setBounds(300,85,150,30);
//--------------------------PATIENT TYPE-------------------------------------
label3=new JLabel();
label3.setText("Patient Type");
panel.add(label3).setBounds(100,120,160,35);
label3.setFont(new Font("courier new",Font.BOLD,18));
label3.setForeground(new Color(50,48,20));
//--------------------------TEXT3-------------------------------------
text3=new JTextField(20);
text3.setBackground(new Color(10,255,160));
panel.add(text3).setBounds(300,120,150,30);
//------------------------DEPARTMENT----------------------------------
label4=new JLabel();
label4.setText("Department");
```

```
panel.add(label4).setBounds(100,155,160,35);
label4.setFont(new Font("courier new",Font.BOLD,18));
label4.setForeground(new Color(50,48,20));
//--------------------------TEXT4--------------------------------------
text4=new JTextField(20);
text4.setBackground(new Color(10,255,160));
panel.add(text4).setBounds(300,155,150,30);
//--------------------------CONSULTANT ID-------------------------------------
label5=new JLabel();
label5.setText("Consultant ID");
panel.add(label5).setBounds(100,190,160,35);
label5.setFont(new Font("courier new",Font.BOLD,18));
label5.setForeground(new Color(50,48,20));
//--------------------------TEXT5--------------------------------------
text5=new JTextField(20);
text5.setBackground(new Color(10,255,160));
panel.add(text5).setBounds(300,190,150,30);
//--------------------------FIRST NAME-------------------------------------
label6=new JLabel();
label6.setText("First Name");
panel.add(label6).setBounds(100,225,160,35);
label6.setFont(new Font("courier new",Font.BOLD,18));
label6.setForeground(new Color(50,48,20));
//--------------------------TEXT6--------------------------------------
text6=new JTextField(20);
text6.setBackground(new Color(10,255,160));
panel.add(text6).setBounds(300,225,150,30);
//--------------------------MIDDLE NAME-------------------------------------
label7=new JLabel();
label7.setText("Middle Name");
panel.add(label7).setBounds(100,260,160,35);
label7.setFont(new Font("courier new",Font.BOLD,18));
label7.setForeground(new Color(50,48,20));
//--------------------------TEXT7--------------------------------------
text7=new JTextField(20);
text7.setBackground(new Color(10,255,160));
panel.add(text7).setBounds(300,260,150,30);
//--------------------------LAST NAME-------------------------------------
label8=new JLabel();
label8.setText("Last Name");
panel.add(label8).setBounds(100,295,160,35);
label8.setFont(new Font("courier new",Font.BOLD,18));
label8.setForeground(new Color(50,48,20));
//--------------------------TEXT8--------------------------------------
text8=new JTextField(20);
text8.setBackground(new Color(10,255,160));
panel.add(text8).setBounds(300,295,150,30);
//--------------------------DATE-------------------------------------
label9=new JLabel();
label9.setText("Date");
```

```
panel.add(label9).setBounds(550,50,160,35);
label9.setFont(new Font("courier new",Font.BOLD,18));
label9.setForeground(new Color(50,48,20));
//--------------------------TEXT9--------------------------------------
text9=new JTextField(20);
text9.setBackground(new Color(10,255,160));
panel.add(text9).setBounds(740,50,150,30);
//--------------------------D.O.B-------------------------------------
label10=new JLabel();
label10.setText("D.O.B");
panel.add(label10).setBounds(550,85,160,35);
label10.setFont(new Font("courier new",Font.BOLD,18));
label10.setForeground(new Color(50,48,20));
//--------------------------TEXT10-------------------------------------
text10=new JTextField(20);
text10.setBackground(new Color(10,255,160));
panel.add(text10).setBounds(740,85,150,30);
//--------------------------GENDER-------------------------------------
label11=new JLabel();
label11.setText("Gender");
panel.add(label11).setBounds(550,120,160,35);
label11.setFont(new Font("courier new",Font.BOLD,18));
label11.setForeground(new Color(50,48,20));
//----------------------------COMBOBOX1-----------------------------------
cb1=new JComboBox();
cb1.addItem("Male");
cb1.addItem("Female");
cb1.addItemListener(this);
cb1.setFont(new Font("courier new",Font.BOLD,16));
cb1.setBackground(new Color(10,255,160));
panel.add(cb1).setBounds(740,120,150,30);
//------------------------------MARITAL
STATUS-------------------------------------
label12=new JLabel();
label12.setText("Marital Status");
panel.add(label12).setBounds(550,155,160,35);
label12.setFont(new Font("courier new",Font.BOLD,18));
label12.setForeground(new Color(50,48,20));
//----------------------------COMBOBOX2-----------------------------------
cb2=new JComboBox();
cb2.addItem("Married");
cb2.addItem("Unmarried");
cb2.addItemListener(this);
cb2.setFont(new Font("courier new",Font.BOLD,16));
cb2.setBackground(new Color(10,255,160));
panel.add(cb2).setBounds(740,155,150,30);

//--------------------------OCCUPATION-------------------------------------
label13=new JLabel();
label13.setText("Occupation");
```

```
panel.add(label13).setBounds(550,190,140,35);
label13.setFont(new Font("courier new",Font.BOLD,18));
label13.setForeground(new Color(50,48,20));
//-----------------------TEXT11--------------------------------------
text11=new JTextField(20);
text11.setBackground(new Color(10,255,160));
panel.add(text11).setBounds(740,190,150,30);
//-------------------------ADDRESS-------------------------------------
label14=new JLabel();
label14.setText("Address");
panel.add(label14).setBounds(550,225,140,35);
label14.setFont(new Font("courier new",Font.BOLD,18));
label14.setForeground(new Color(50,48,20));
//-------------------------TEXT12--------------------------------------
text12=new JTextField(20);
text12.setBackground(new Color(10,255,160));
panel.add(text12).setBounds(740,225,150,30);
//-------------------------MOBILE--------------------------------------
label15=new JLabel();
label15.setText("Mobile");
panel.add(label15).setBounds(550,260,140,35);
label15.setFont(new Font("courier new",Font.BOLD,18));
label15.setForeground(new Color(50,48,20));
//-------------------------TEXT13--------------------------------------
text13=new JTextField(20);
text13.setBackground(new Color(10,255,160));
panel.add(text13).setBounds(740,260,150,30);
//-------------------------REMARKS-------------------------------------
label16=new JLabel();
label16.setText("Remarks");
panel.add(label16).setBounds(550,295,140,35);
label16.setFont(new Font("courier new",Font.BOLD,18));
label16.setForeground(new Color(50,48,20));
//-------------------------TEXT14--------------------------------------
text14=new JTextField(20);
text14.setBackground(new Color(10,255,160));
panel.add(text14).setBounds(740,295,150,30);
//-------------------------ADD BUTTON----------------------------------
bt1=new JButton("ADD");
panel.add(bt1).setBounds(100,450,110,40);
bt1.setFont(new Font("courier new",Font.BOLD,18));
bt1.setForeground(new Color(255,78,90));
bt1.setBackground(new Color(50,48,20));
//----------------------------CLEAR BUTTON-------------------------------------
bt2=new JButton("CLEAR");
panel.add(bt2).setBounds(250,450,110,40);
bt2.setFont(new Font("courier new",Font.BOLD,18));
bt2.setForeground(new Color(255,78,90));
bt2.setBackground(new Color(50,48,20));
//-------------------------SEARCH BUTTON-------------------------------------
```

```
bt3=new JButton("SEARCH");
panel.add(bt3).setBounds(400,450,110,40);
bt3.setFont(new Font("courier new",Font.BOLD,18));
bt3.setForeground(new Color(255,78,90));
bt3.setBackground(new Color(50,48,20));
//-------------------------EDIT BUTTON-------------------------------------
bt4=new JButton("EDIT");
panel.add(bt4).setBounds(550,450,110,40);
bt4.setFont(new Font("courier new",Font.BOLD,18));
bt4.setForeground(new Color(255,78,90));
bt4.setBackground(new Color(50,48,20));
//-------------------------DELETE BUTTON-------------------------------------
bt5=new JButton("DELETE");
panel.add(bt5).setBounds(700,450,110,40);
bt5.setFont(new Font("courier new",Font.BOLD,18));
bt5.setForeground(new Color(255,78,90));
bt5.setBackground(new Color(50,48,20));
setDefaultCloseOperation(EXIT_ON_CLOSE);
}
        public void actionPerformed(ActionEvent ae)
        {
        }
      public void itemStateChanged(ItemEvent ie)
      {
      }
        public static void main(String arg[])
        {
        Registration1 from=new Registration1();
        from.display();
        }
}
```