15.7 Layout Manager

A layout manager is used to control the size and position of components inside a container object.

The layout manager automatically positions all the components within the container. If we do not use layout manager than also the components are positioned by the default layout manager. It is possible to layout the controls by hand but it becomes very difficult because of the following two reasons.

- It is very tedious to handle a large number of controls within the container.
- Oftenly the width and height information of a component is not given when we need to arrange them.

Java provide us with various layout manager to position the controls. The properties like size, shape and arrangement varies from one layout manager to other layout manager. When the size of the applet or the application window changes the size, shape and arrangement of the components also changes in response i.e. the layout managers adapt to the dimensions of appletviwer or the application window.

Def. :

A layout manager is an object that controls the size and the position of components in a container. Therefore, Every Container object has a layout manager object that controls its layout.

AWT has five Layout Manager Classes :

(i) FlowLayout

(ii) GridLayout

(iii) BorderLayout

(iv) CardLayout

(v) GridBagLayout

You can set a new layout manager at any time with the **setLayout( )** method. For example, we set the layout manager of a container to a **BorderLayout** as :

```
setLayout ( new BorderLayout ( )  ) ;
```

Notice that we call the **BorderLayout** constructor

Methods that Result in Calls to the Container's Layout manager are :

1. add( ), remove( ), removeAll ( ) - They add and remove **components** from a **Container**; you can call them at any time.

2. getPreferredSize( ), getMinimumSize( ), getMaximumSize( ) - They return the Container's ideal size, minimum size and maximum size, respectively.

To set up the layout of a Container (such as Frame, Panel) you have to :

(i) Construct an instance of the chosen layout object, via new and constructor, e.g. new **FlowLayout( )**

(ii) Invoke the **setLayout( )** method of the Container, with the layout object created as the argument;

(iii) Place the GUI components into the Container using the **add( )** method in the correct order; or into the correct zones.

15.7.1 FlowLayout

The FlowLayout is used to arrange the components in a line (or row), one after another (in a flow). It is the default layout of applet or panel.

It simply lays out the components from left to right, if first row is complete then starting new rows if necessary. You start at row one, column one, move across the row until it's full and then continue on to the next row.

Constructors of FlowLayout class are :

(i) FlowLayout( )

Creates a flow layout with centered alignment and a default 5 unit horizontal and vertical gap.

(ii) FlowLayout(int align)

Creates a flow layout with the given alignment and a default 5 unit horizontal and vertical gap.

(iii) FlowLayout(int align, int hgap, int vgap)

Creates a flow layout with the given alignment and the given horizontal and vertical gap.

Example

```
import java.awt.*;
class demoflowlayout extends Frame
{
    public static void main(String args[ ])
    {
            Button b1=new Button("First Button");
            Button b2=new Button("Second Button");
            Button b3=new Button("Three");
            Button b4=new Button("Fourth Button");
            Button b5=new Button("Fifth Button");
            Button b6=new Button("Sixth Button");

            demoflowlayout f1 = new demoflowlayout();   //Create a frame
            f1.setTitle("Flow Layout");   //Set title of  frame
            f1.setLayout(new FlowLayout()); //Set Frame Layout to FlowLayout
            f1.add(b1);   //adding a component (button) to the frame
            f1.add(b2);
            f1.add(b3);
            f1.add(b4);
            f1.add(b5);
            f1.add(b6);
            f1.setSize(250, 150);   //set size of frame
            f1.setVisible(true);
    }
}
```

Output :

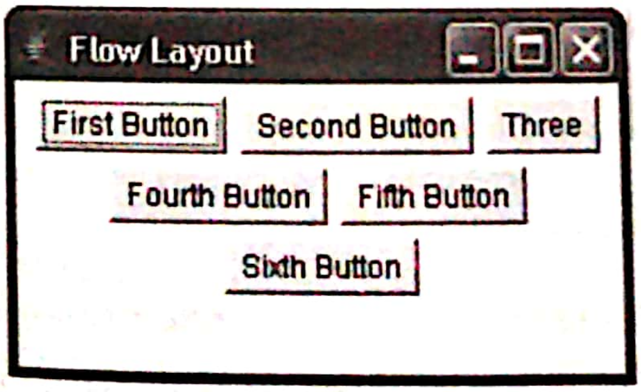

FIGURE 15.15

15.7.2 GridLayout

In java.awt.GridLayout, components are arranged in a grid (matrix) of rows and columns inside the Container. Components are added in a left-to-right, top-to-bottom manner in the order they are added.

Each row and each column in a grid layout will be the same size based on the largest components in the grid. The overall area available to the layout is divided into cells equally between the number of rows and the number of columns.

Constructors of GridLayout class are :

(i)

GridLayout(int rows, int columns)

Creates a new grid layout with specified rows and columns.

(ii) **GridLayout(int rows, int columns, int hgap, int vgap)**

Creates a new grid layout with specified rows, columns, horizontal and vertical gap.

Example

```
import java.awt.*;
class demogridlayout extends Frame
{
    public static void main(String args[ ])
    {
            Button b1=new Button("First Button");
            Button b2=new Button("Second Button");
            Button b3=new Button("Three");
            Button b4=new Button("Fourth Button");
            Button b5=new Button("Fifth Button");
            Button b6=new Button("Sixth Button");

            demogridlayout f1 = new demogridlayout();   //Create a frame
            f1.setTitle("Grid Layout");   //Set title of frame
            f1.setLayout(new GridLayout(3,2,3,2));
                                          // Set Frame Layout to Grid Layout
            f1.add(b1);   //adding a component (button) to the frame
            f1.add(b2);
            f1.add(b3);
            f1.add(b4);
            f1.add(b5);
            f1.add(b6);
            f1.setSize(250, 150);   //set size of frame
            f1.setVisible(true);

      }
}
```

Output :

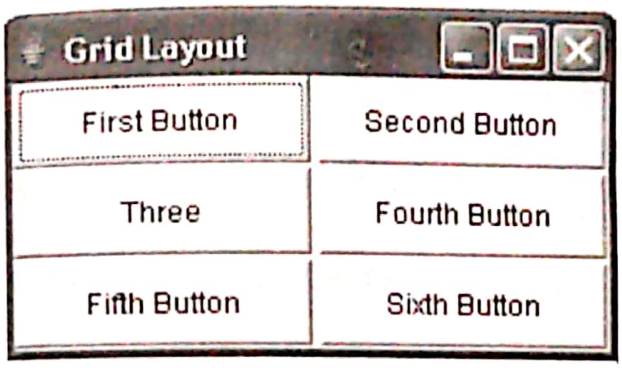

FIGURE 15.16

15.7.3 BorderLayout

In java.awt.BorderLayout, the container is divided into 5 zones; EAST, WEST, SOUTH, NORTH and CENTER. Components are added using method add(component, Zone), where zone can take one of the following values :

BorderLayout.NORTH (or PAGE_START),

BorderLayout.SOUTH (or PAGE_END)

BorderLayout.WEST (or LINE_START)

BorderLayout.EAST (or LINE_END)

BorderLayout.CENTER.

The method **add(Component)** without specifying the zone adds the component to the CENTER.

Constructors of BorderLayout class are :

(i) **BorderLayout( )**

Creates a new border layout.

(ii) **BorderLayout (int hgap, intvgap)**

Create a BorderLayout with a horizontal gap of hgap and vertical gap of vgap.

Example

```
import java.awt.*;
class demoborderlayout extends Frame
{
    public static void main(String args[ ])
    {
            Button b1=new Button("First Button");
            Button b2=new Button("Second Button");
            Button b3=new Button("Three");
            Button b4=new Button("Fourth Button");
            Button b5=new Button("Fifth Button");

            demoborderlayout f1 = new demoborderlayout();   //Create a frame
            f1.setTitle("Border Layout");   //Set title of  frame
            f1.setLayout(new BorderLayout()); // //Set Frame Layout to
            BorderLayout
                                                    Page start
            f1.add(b1,BorderLayout.NORTH);   //adding a component (button)
            to the frame
            f1.add(b2, BorderLayout.SOUTH);          Page end
            f1.add(b3, BorderLayout.CENTER);         center
            f1.add(b4, BorderLayout.EAST);           Line end
            f1.add(b5, BorderLayout.WEST);           Line start

            f1.setSize(250, 150);   //set size of frame
            f1.setVisible(true);
    }
}
```

Output :

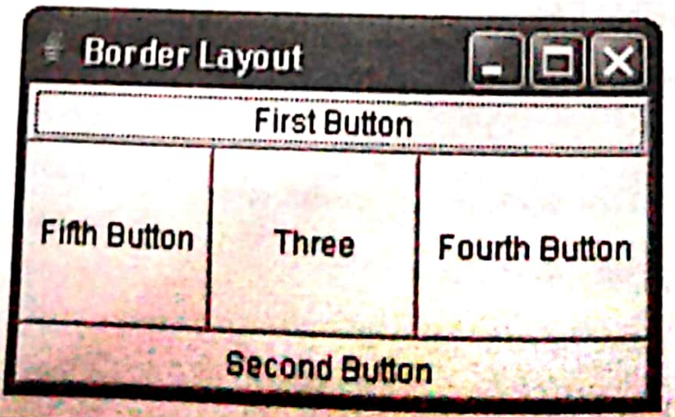

FIGURE 15.17

15.7.4 CardLayout

The class **CardLayout** arranges each component in the container as a card. Only one card is visible at a time, and the container acts as a stack of cards.

CardLayout manager's usage is very need-based and quiet different from others where other layout managers try to display all the components added to the container. But card layout manager displays only one component at a time. Where a tabbed panel, like Page Setup option in MS Word, is required we use card layout manager. Another example is MS Excel workbook in which the selected sheet is opened by clicking over sheet tab.

Constructors of CardLayout class are :

(i) CardLayout( )

Creates a new cardLayout with gaps of size zero.

(ii) CardLayout(int hgap, int vgap)

Creates a new card layout with the specified horizontal and vertical gaps.

Commonly used methods of CardLayout class :

- **public void next(Container parent) :** is used to flip to the next card of the given container.
- **public void previous(Container parent) :** is used to flip to the previous card of the given container.
- **public void first(Container parent) :** is used to flip to the first card of the given container.
- **public void last(Container parent) :** is used to flip to the last card of the given container.
- **public void show(Container parent, String name) :** is used to flip to the specified card with the given name.