Layout Managers

1. What is layout manager?

Or

How flow layout can be handled in Java? Show with an example.

Or

Briefly describe FlowLayout, BorderLayout and GridLayout.

Or

What is layout manager? Describe different types of layout manager.

Or

Differentiate between flow layout and border layout.

Ans:

Layout Manager:

A layout manager is an object that controls the size and position (layout) of components inside a Container object. For example, a window is a container that contains components such as buttons and labels. The layout manager in effect for the window determines how the components are sized and positioned inside the window. A container can contain another container. For example, a window can contain a panel, which is itself a container.

Java has several predefined LayoutManager classes.

- FlowLayout
- BorderLayout
- GridLayout
- GridBagLayout
- CardLayout

FlowLayout:
FlowLayout is the default layout manager. 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. A small space is left between each component, above and below, as well as left and right. The constructors for FlowLayout are shown below:

```
FlowLayout( )
FlowLayout(int how)
FlowLayout(int how, int horz, int vert)
```

The *first form* creates the default layout, which centers components and leaves five pixels of space between each component.
The *second form* lets you specify how each line is aligned. Valid values for how are as follows:

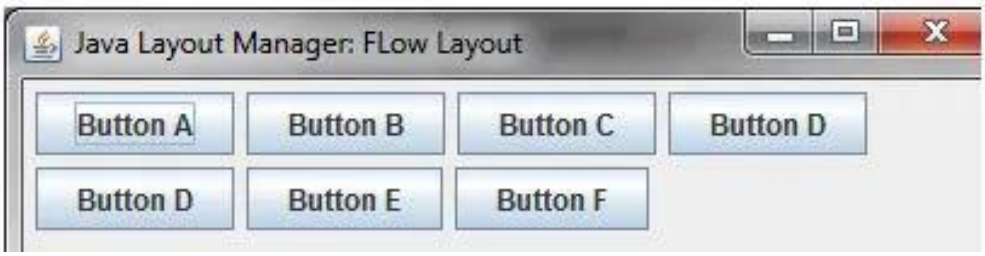
- `FlowLayout.LEFT`
- `FlowLayout.CENTER`
- `FlowLayout.RIGHT`

The *third form* allows specifying the horizontal and vertical space left between components in `horz` and `vert`, respectively.

Example program of FlowLayout:

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

public class FlowRight extends Applet {
     public void init(){
          setLayout(new FlowLayout(FlowLayout.RIGHT));
          for(int i=0; i<4; i++){
               add(new Button("Button #"+i));
          }
     }
}
```



Border Layout:
The BorderLayout class implements a common layout style for top-level windows. It has four narrow, fixed-width components at the edges and one large area in the center. The four sides are referred to as north, south, east, and west. The middle area is called the center. The constructors defined by BorderLayout are shown below:

```
BorderLayout( )
BorderLayout(int horz, int vert)
```

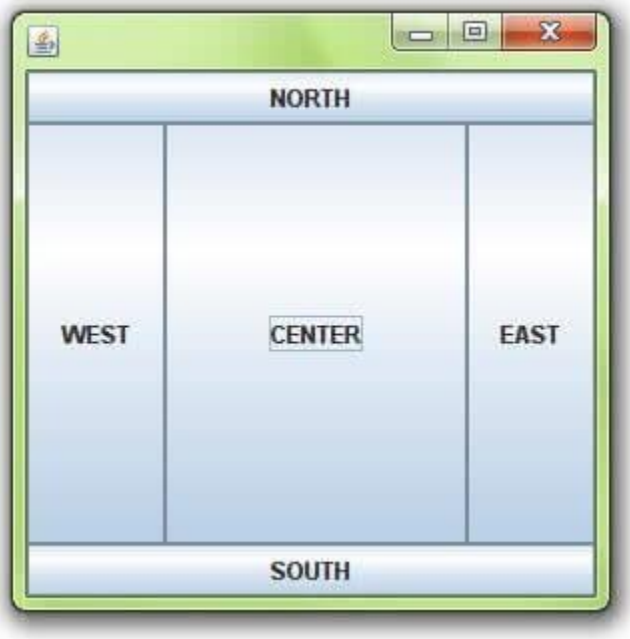
The *first form* creates a default border layout. The *second form* allows you to specify the horizontal and vertical space left between components in `horz` and `vert`, respectively.

Example program of BorderLayout:

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

public class BorderLayoutDemo extends Applet {
     public void init(){
          setLayout(new BorderLayout());

          add(new Button("Button1"), BorderLayout.NORTH);
          add(new Label("Center"), BorderLayout.CENTER);
     }
}
```



GridLayout:
GridLayout lays out components in a two-dimensional grid. When you instantiate a GridLayout, you define the number of rows and columns. The constructors supported by GridLayout are shown below:

```
GridLayout( )
GridLayout(int numRows, int numColumns )
GridLayout(int numRows, int numColumns, int horz,
int vert)
```

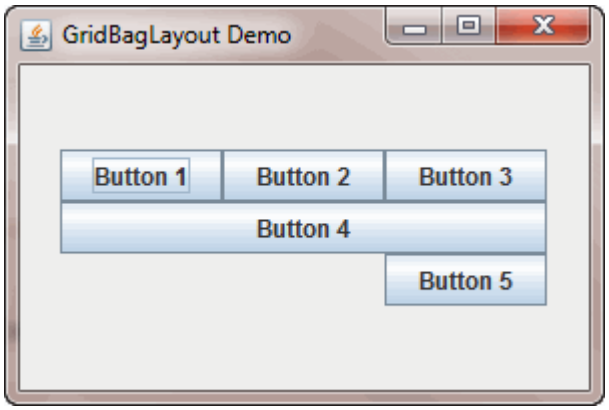
The *first form* creates a single-column grid layout. The *second form* creates a grid layout with the specified number of rows and columns. *The third form* allows you to specify the horizontal and vertical space left between components in horz and vert, respectively.

Example program of GridLayout:

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

public class GridLayoutDemo extends Applet {
     public void init(){
          setLayout(new GridLayout(5,2));
          for(int row=1; row<=5; row++){
               add(new Label("Label"+row));
               add(new Button("Button"+row));
          }
     }
}
```

GridBagLayout:
By using GridBagLayout you can specify the relative placement of components by specifying their positions within cells inside a grid. The key to the grid bag is that each component can be a different size, and each row in the grid can have a different number of columns. This is why the layout is called a grid bag. It's a collection of small grid joined together.



CardLayout:
The CardLayout class is unique among the other layout managers in that it stores several different layouts. Each layout can be thought of as being on a separate index card in a deck that can be shuffled so that any card is on top at a given time. This can be useful for user interfaces with optional components that can be dynamically enabled and disabled upon user input. We can prepare the other layouts and have them hidden, ready to be activated when needed. CardLayout provides the following two constructors:

```
CardLayout( )
CardLayout(int horz, int vert)
```

The *first form* creates a default card layout. The *second form* allows you to specify the horizontal and vertical space left between components in `horz` and `vert`, respectively.