

# Layout Management

- To add components to a container for e.g. a Frame instance,
  - we need to specify the type of component layout scheme to be used i.e.
    - how components within that container should be arranged.
- Layout managers implements a layout policy that defines the spatial relationship between components in a container.
- Layout managers are provided to arrange GUI components on a container for presentation purposes.
- The layout managers provide basic layout capabilities that are easier to use. This enables the programmer to concentrate on the basic “look and feel” and lets the layout managers process most of the layout details.
- Layout managers deals with:
  - location and size of the components.
  - the way to update the interface when user re-sizes the window.

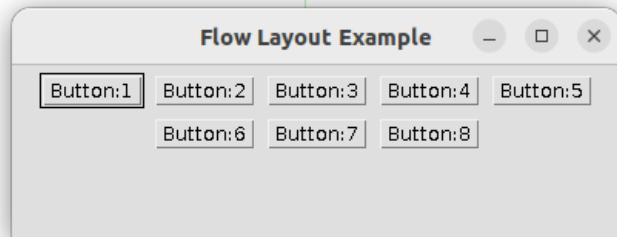
Layout Manager	Constructors
1. <b>FlowLayout</b>	<p>a. <b>FlowLayout()</b> Constructs a new FlowLayout with a centered alignment and a default 5-unit horizontal and vertical gap.</p> <p>b. <b>FlowLayout(int align)</b> Constructs a new FlowLayout with the specified alignment and a default 5-unit horizontal and vertical gap.</p> <p>c. <b>FlowLayout(int align, int hgap, int vgap)</b> Creates a new flow layout manager with the indicated alignment and the indicated horizontal and vertical gaps. Alignment can be LEFT, CENTER and RIGHT.</p>
2. <b>BorderLayout</b>	<p>a. <b>BorderLayout()</b> Constructs a new border layout with no gaps between components.</p> <p>b. <b>BorderLayout(int hgap, int vgap)</b> Constructs a border layout with the specified gaps between components.</p>
3. <b>GridLayout</b>	<p>a. <b>GridLayout():</b> Creates a grid layout with a default of one column per component, in a single row.</p> <p>b. <b>GridLayout(int rows, int cols):</b> Creates a grid layout with the specified number of rows and columns.</p> <p>c. <b>GridLayout(int rows, int cols, int hgap, int vgap):</b> Creates a grid layout with the specified number of rows and columns.</p>
4. <b>CardLayout</b>	<p>a. <b>CardLayout()</b> Creates a new card layout with gaps of size zero.</p> <p>b. <b>CardLayout(int hgap, int vgap)</b> Creates a new card layout with the specified horizontal and vertical gaps.</p>
5. <b>GridBagLayout</b>	<p>a. <b>GridBagLayout():</b> Creates a grid bag layout manager.</p>
6. <b>BoxLayout</b>	<p>a. <b>BoxLayout(Container c, int axis)</b> creates a box layout that arranges the components in X-axis or Y-axis..</p>

## 1. Flowlayout

```

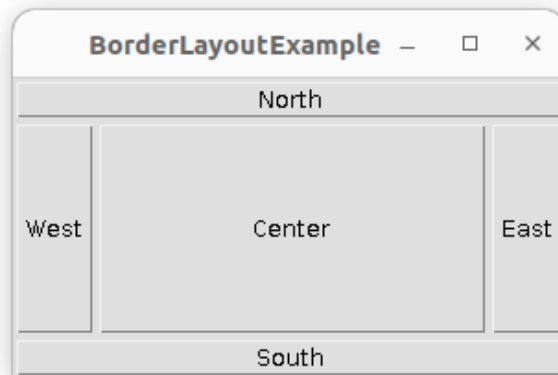
1  // It arranges the components in a line, one after another (in a flow). It is the default layoutof panel.
2
3  import java.awt.*;
4  public class FlowLayoutExample extends Frame{
5      // constructor
6      public FlowLayoutExample(String title)
7      {
8          //It creates the Frame by calling the constructor of Frame class.
9          super(title);
10
11          //Setting up Flow Layout
12          setLayout(new FlowLayout());
13
14          //Creating a button and adding it to the frame
15          Button b1 = new Button("Button:1");
16          add(b1);
17
18          //Adding other components to the Frame
19          Button b2 = new Button("Button:2");
20          add(b2);
21
22          Button b3 = new Button("Button:3");
23          add(b3);
24
25          Button b4 = new Button("Button:4");
26          add(b4);
27
28          Button b5 = new Button("Button:5");
29          add(b5);
30
31          Button b6 = new Button("Button:6");
32          add(b6);
33
34          Button b7 = new Button("Button:7");
35          add(b7);
36
37          Button b8 = new Button("Button:8");
38          add(b8);
39
40      }
41      public static void main(String[] args)
42      {
43          FlowLayoutExample screen = new FlowLayoutExample("Flow Layout Example");
44          screen.setSize(400,150); //sets the width and height of the frame
45          screen.setVisible(true);
46      }
47  }
48

```



## 2. BorderLayout

```
1  import java.awt.*;
2  public class BorderLayoutExample extends Frame {
3      public BorderLayoutExample(String title) {
4          super(title);
5          add("North", new Button("North"));
6          add("South", new Button("South"));
7          add("East", new Button("East"));
8          add("West", new Button("West"));
9          add("Center", new Button("Center"));
10     }
11     public static void main(String[] args) {
12         BorderLayoutExample ble = new BorderLayoutExample("BorderLayoutExample");
13         ble.setSize(300, 200);
14         ble.setVisible(true);
15     }
16 }
17
```

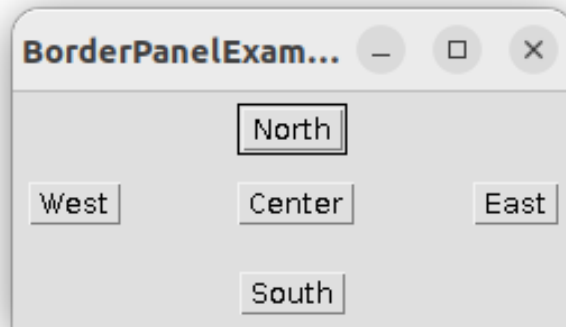


## 3. BorderLayout with Panel

```

1  import java.awt.*;
2  public class BorderLayoutExample extends Frame {
3      public BorderLayoutExample(String title) {
4          super(title);
5          addComponent("North", new Button("North"));
6          addComponent("South", new Button("South"));
7          addComponent("East", new Button("East"));
8          addComponent("West", new Button("West"));
9          addComponent("Center", new Button("Center"));
10     }
11     public void addComponent(String region, Component component) {
12         Panel panel = new Panel();
13         panel.add(component);
14         add(region, panel); // make sure you add the panel!
15     }
16     public static void main(String[] args) {
17         BorderLayoutExample bpe =
18             new BorderLayoutExample("BorderPanelExample");
19         bpe.setSize(200, 150);
20         bpe.setVisible(true);
21     }
22 }

```



## 4. GridLayout

```
1 // arrange the components in a rectangular grid/table
2
3 import java.awt.*;
4
5 public class GridLayoutBasicExample extends Frame {
6     public GridLayoutBasicExample( String title) {
7         setLayout(new GridLayout(3,2));
8         add(new Button("1"));
9         add(new Button("2"));
10        add(new Button("3"));
11        add(new Button("4"));
12        add(new Button("5"));
13        add(new Button("6"));
14    }
15
16    public static void main(String args[]){
17
18        GridLayoutBasicExample gl = new GridLayoutBasicExample("GridLayout Basic Example");
19        gl.setSize(300, 200);
20        gl.setVisible(true);
21    }
22 }
23
24
```

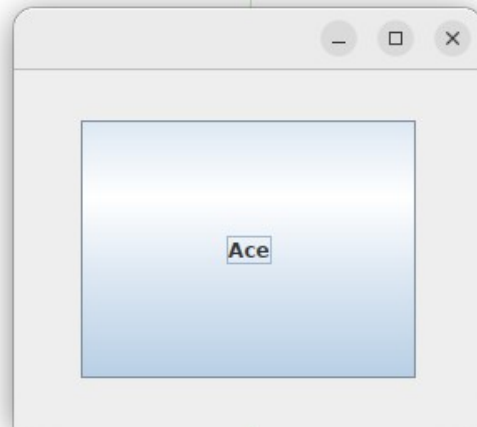


## 5. CardLayout

```

1  import java.awt.*;
2  import java.awt.event.*;
3
4  import javax.swing.*;
5
6  public class CardLayoutBasicExample extends JFrame implements ActionListener{
7      CardLayout card;
8      JButton b1,b2,b3,b4;
9      Container c;
10     CardLayoutBasicExample(){
11
12         c=getContentPane(); // Gets the content pane layer so that we can add elements to it
13         card=new CardLayout(40,30);
14         //create CardLayout object with 40 hor space and 30 ver space
15         c.setLayout(card);
16
17         b1=new JButton("Ace");
18         b2=new JButton("King");
19         b3=new JButton("Queen");
20         b4=new JButton("Jocker");
21
22         b1.addActionListener(this);
23         b2.addActionListener(this);
24         b3.addActionListener(this);
25         b4.addActionListener(this);
26
27         c.add(b1);c.add(b2);c.add(b3); c.add(b4);
28
29     }
30
31     public void actionPerformed(ActionEvent e) {
32         card.next(c);
33     }
34
35     public static void main(String[] args) {
36         CardLayoutBasicExample cl=new CardLayoutBasicExample();
37         cl.setSize(500,400);
38         cl.setVisible(true);
39         cl.setDefaultCloseOperation(EXIT_ON_CLOSE);
40     }
41 }

```

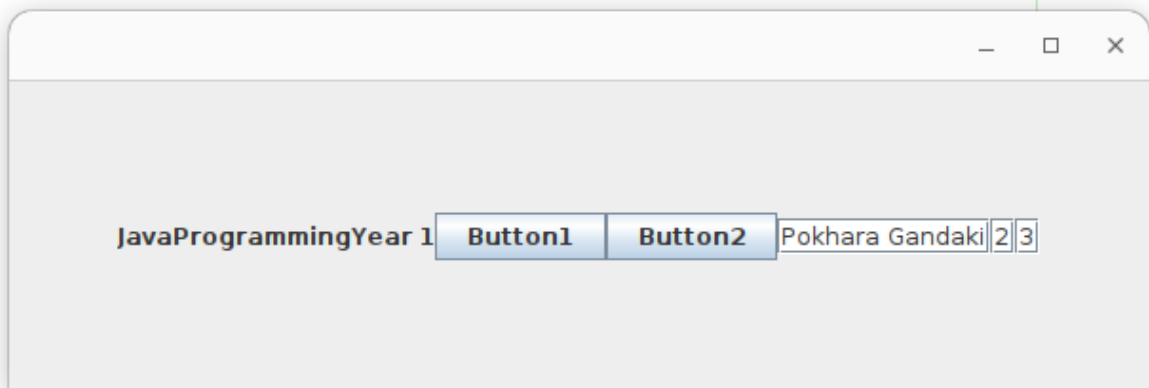


## 6. GridBagLayout

```

1  // can arrange components vertically, horizontally or along their baseline.
2  // The components may not be of the same size. Each GridBagLayout object
3  // maintains a dynamic, rectangular grid of cells.
4  import java.awt.*;
5  import javax.swing.*;
6  public class GridBagBasicExample {
7
8      public static void main(String[] args) {
9          JFrame f = new JFrame();
10         JPanel p = new JPanel();
11
12         p.setLayout(new GridBagLayout());
13         p.add(new JLabel("Java"));
14         p.add(new JLabel("Programming"));
15         p.add(new JLabel("Year 1"));
16         p.add(new JButton("Button1"));
17         p.add(new JButton("Button2"));
18         p.add(new JTextField("Pokhara Gandaki"));
19         p.add(new JTextField("2"));
20         p.add(new JTextField("3"));
21
22         f.getContentPane().add(p);
23         f.setSize(600, 200);
24         f.setVisible(true);
25     }
26 }
27

```





## 7. BoxLayout

```

1  // arrange the components in a rectangular grid/table
2
3  import java.awt.*;
4  import javax.swing.*; // for BoxLayout
5
6
7  public class BoxLayoutBasic extends Frame {
8
9      public BoxLayoutBasic( String title) {
10
11          setLayout (new BoxLayout (this, BoxLayout.X_AXIS)); // Set Box Layout
12          // BoxLayout.X_AXIS : horizontally add buttons
13          // BoxLayout.Y_AXIS : vertically add buttons
14
15          add(new Button("1"));
16          add(new Button("2"));
17          add(new Button("3"));
18          add(new Button("4"));
19          add(new Button("5"));
20          add(new Button("6"));
21      }
22
23      public static void main(String args[]){
24
25          BoxLayoutBasic blb = new BoxLayoutBasic("BoxLayout Basic");
26          blb.setSize(300, 200);
27          blb.setVisible(true);
28      }
29  }
30
31

```

