UNIT V – MULTITHREADING, GUI PROGRAMMING & EVENT HANDLING

Layout Managers:

- In Java AWT and Swing, a Layout Manager is an object that controls the size, position, and arrangement of components (like buttons, labels, text fields) inside a container (like Frame, Panel, JFrame, or JPanel).
- Instead of manually setting positions using setBounds(x, y, width, height), layout managers automatically adjust component placement based on container size, screen resolution, and resizing.
- Defined in the package: java.awt
- Default:
  - ✓ For Frame/JFrame → BorderLayout.
  - ✓ For Panel/JPanel → FlowLayout

Complete List of Layout Managers in Java

1. FlowLayout
2. BorderLayout
3. GridLayout
4. CardLayout
5. GridBagLayout
6. BoxLayout (Swing only)
7. GroupLayout (Swing only, Java SE 6+)
8. SpringLayout (Swing only)
9. Null Layout (absolute positioning)

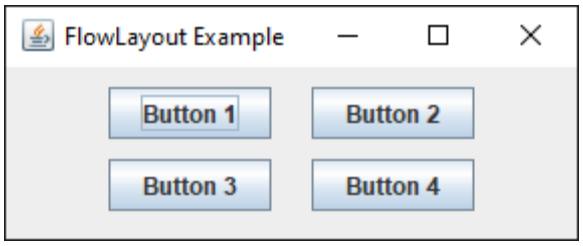
- ✓ For simple apps → FlowLayout, BorderLayout, GridLayout.
- ✓ For professional apps → GridBagLayout, BoxLayout, GroupLayout.
- ✓ For special cases → CardLayout (wizards), SpringLayout (constraints), Null Layout (games).

1. FlowLayout:

- Arranges components in a row, left to right.
- When space is insufficient, moves components to the next line.
- Alignments: LEFT, CENTER (default), RIGHT.

```
import java.awt.*;
import javax.swing.*;
public class FlowLayoutExample
{
```

```
    public static void main(String[] args)
    {
        JFrame f = new JFrame("FlowLayout Example");
        // align=center, hgap=20, vgap=10
        f.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 10));
        f.add(new JButton("Button 1"));
        f.add(new JButton("Button 2"));
        f.add(new JButton("Button 3"));
        f.add(new JButton("Button 4"));
        f.setSize(200, 200);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

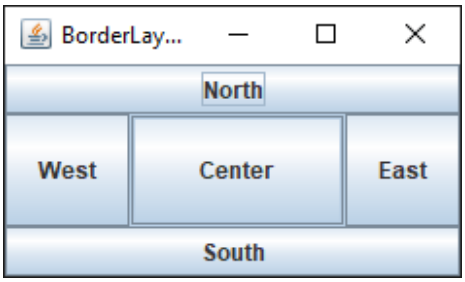


2. BorderLayout:

- Divides the container into 5 regions: NORTH, SOUTH, EAST, WEST, CENTER
- Only one component per region.
- CENTER expands to fill unused space.

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutExample
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame("BorderLayout Example");
        f.setLayout(new BorderLayout());
```

```
        f.add(new JButton("North"), BorderLayout.NORTH);
        f.add(new JButton("South"), BorderLayout.SOUTH);
        f.add(new JButton("East"), BorderLayout.EAST);
        f.add(new JButton("West"), BorderLayout.WEST);
        f.add(new JButton("Center"), BorderLayout.CENTER);
        f.setSize(250, 250);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```



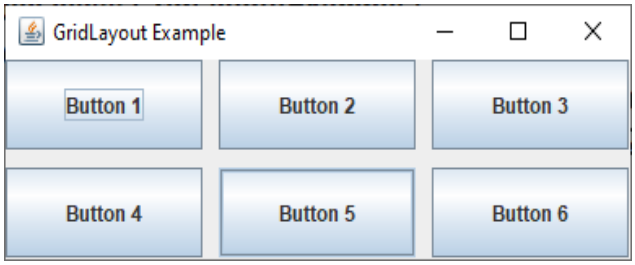
3. GridLayout:

- Divides container into a grid of rows and columns.
- All cells are of equal

size. import

```
java.awt.*;
import javax.swing.*;
public class GridLayoutExample
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame("GridLayout Example");
        f.setLayout(new GridLayout(2, 3, 10, 10)); // 2 rows, 3 columns,
        gaps for (int i = 1; i <= 6; i++)
        {
            f.add(new JButton("Button " + i));
        }
        f.setSize(400, 150);
```

```
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

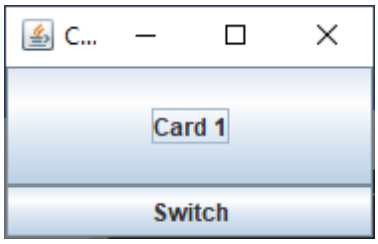


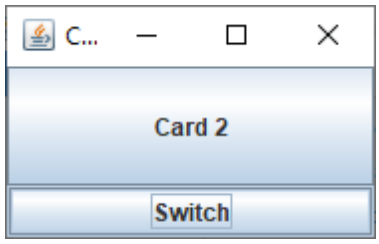
4. CardLayout

- Stacks components like a deck of cards.
- Only one component is visible at a time.
- Useful for wizards or tabbed UI.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
public class CardLayoutExample
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame("CardLayout Example");
        CardLayout card = new CardLayout();
        JPanel panel = new JPanel(card);
        panel.add(new JButton("Card 1"), "card1");
        panel.add(new JButton("Card 2"), "card2");
        JButton switchBtn = new JButton("Switch");
        switchBtn.addActionListener(e -> card.next(panel));
        f.add(panel, BorderLayout.CENTER);
        f.add(switchBtn, BorderLayout.SOUTH);
        f.setSize(200, 125);
        f.setVisible(true);
```

```
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

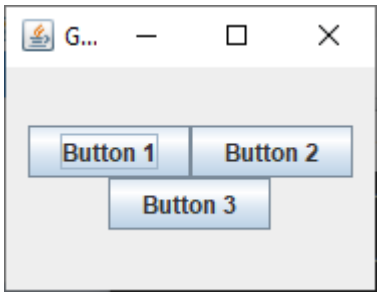




5. GridBagLayout:

- The most flexible and complex layout manager.
- Places components in a grid, but unlike GridLayout, cells can have different sizes, and components can span multiple rows/columns.
- Controlled using GridBagConstraints.

```
import java.awt.*;
import javax.swing.*;
public class GridBagLayoutExample
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame("GridBagLayout Example");
        f.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        JButton b1 = new JButton("Button 1");
        JButton b2 = new JButton("Button 2");
        JButton b3 = new JButton("Button 3");
        gbc.gridx = 0;
        gbc.gridy = 0;                                  gbc.gridy = 0;
        f.add(b1, gbc);                                 f.add(b2, gbc);
        gbc.gridx = 1;                                  gbc.gridx = 0;
```




```
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        f.add(b3, gbc);
        f.setSize(200, 150);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

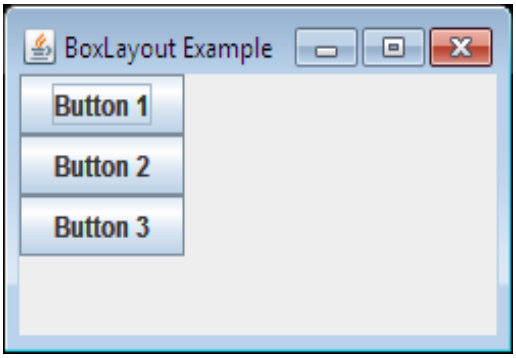
6. BoxLayout (Swing only)

- Arranges components either vertically (Y-axis) or horizontally (X-axis) in a single row or column.
- Unlike FlowLayout, components do not wrap to the next line.
- Belongs to javax.swing package.

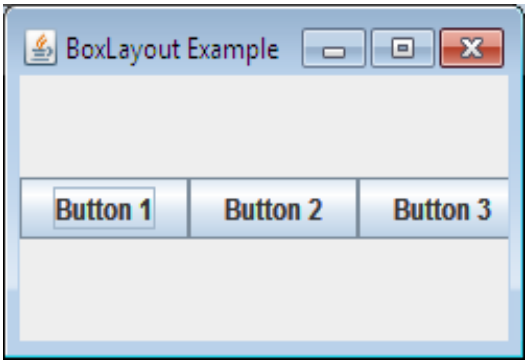
```
import javax.swing.*;
import java.awt.*;
public class BoxLayoutExample
{
    public static void main(String[] args)
    {
        JFrame f = new JFrame("BoxLayout
        Example"); JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
        panel.add(new JButton("Button 1"));
        panel.add(new JButton("Button 2"));
```

```
        panel.add(new JButton("Button 3"));
        f.add(panel);
        f.setSize(250, 150);
        f.setVisible(true);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

Y-axis:



X-axis:



7. GroupLayout (Swing only, Java SE 6+)

- Introduced in Java SE 6.
- Used mainly by GUI builders (like NetBeans).
- Allows placement of components in hierarchical groups, both horizontally and vertically.
- Provides very precise alignment control.

8. SpringLayout (Swing only):

- Very flexible but low-level.
- Defines relationships (springs and constraints) between edges of components.
- Example: "Button A should always be 20 pixels to the right of Button B".
- Rarely used manually → mostly used by IDE GUI builders.

9. Null Layout (absolute positioning):

- Not technically a layout manager (it means no layout manager).
- We can use setLayout(null) and place components manually with

setBounds(x, y, width, height). Not portable across screen resolutions.

- Not resizable. But sometimes used for custom UIs and games.