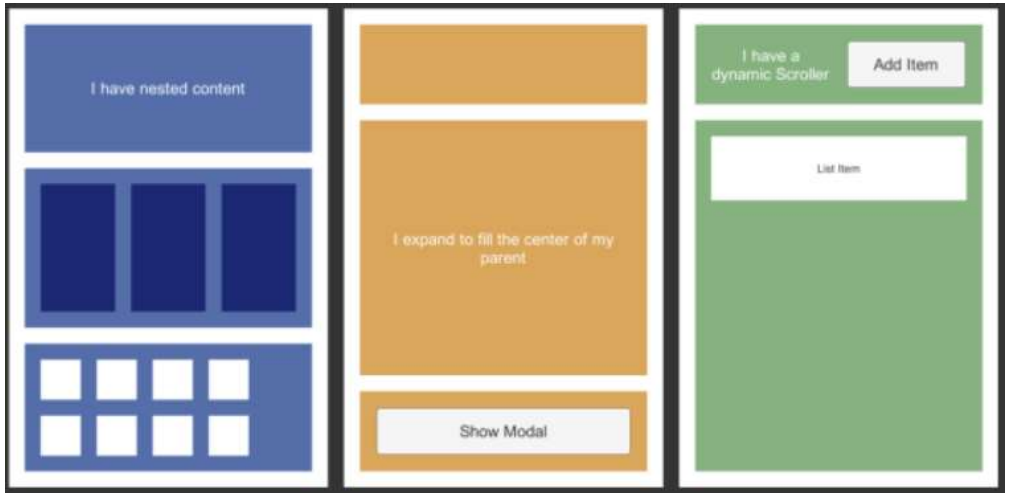
Layout Management (布局管理)

- Layout managers control how to place the GUI components (containers can also be treated as components) in a container for presentation purposes.

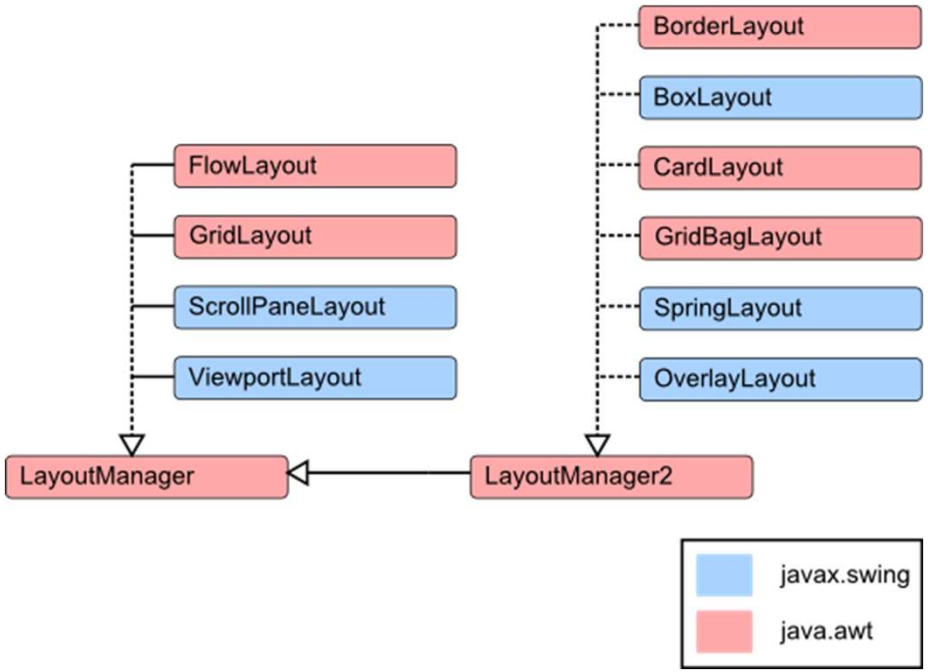
- You can use the layout manager for basic layout capabilities instead of determine every GUI component's exact position and size (which is non-trivial and error-prone)



Layout Management (布局管理)

- All layout managers in Java implement the interface `LayoutManager` (in the package `java.awt`)

- Commonly-used layout managers: `FlowLayout`, `BorderLayout`, `GridLayout`



https://www.mathematik.uni-marburg.de/~thormae/lectures/graphics1/graphics_2_2_eng_web.html#1

FlowLayout

```
public class FlowLayoutDemo extends JFrame {
    private JButton btn1, btn2, btn3, btn4, btn5, btn6;

    public FlowLayoutDemo() {
        super("Flow Layout");
        setLayout(new FlowLayout());
        btn1 = new JButton("Button 1"); add(btn1);
        btn2 = new JButton("This is Button 2"); add(btn2);
        btn3 = new JButton("3"); add(btn3);
        btn4 = new JButton("Another Button 4"); add(btn4);
        btn5 = new JButton("Button 5"); add(btn5);
        btn6 = new JButton("One More Button 6"); add(btn6);
    }

    public static void main(String[] args) { ... }
}
```

FlowLayout

- Default layout manager for the secondary container `javax.swing.JPanel`
- Places components in a straight horizontal line. If there is no enough space to fit all component into one line, simply move the next line

FlowLayout: Alignment


```
setLayout(new FlowLayout(FlowLayout.LEFT));
```


```
setLayout(new FlowLayout(FlowLayout.RIGHT));
```

GridLayout

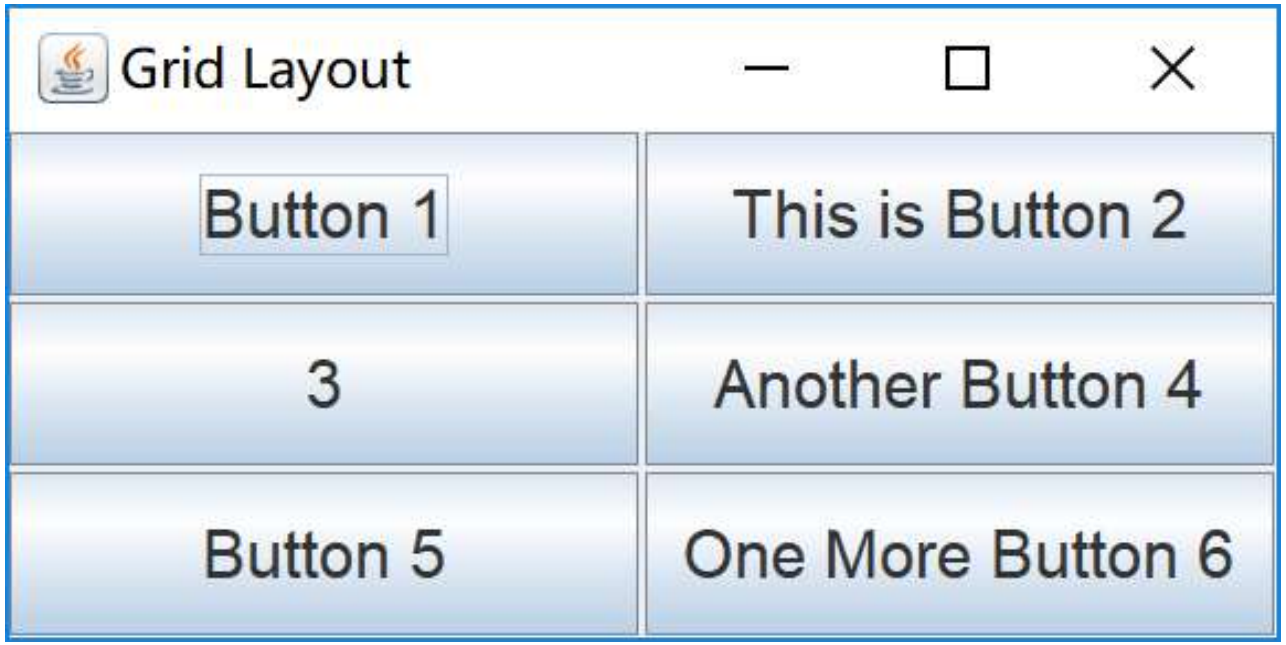
```
public class GridLayoutDemo extends JFrame {
    private JButton btn1, btn2, btn3, btn4, btn5, btn6;

    public GridLayoutDemo() {
        super("Grid Layout");
        setLayout(new GridLayout(3, 2, 3, 3));
        btn1 = new JButton("Button 1"); add(btn1);
        btn2 = new JButton("This is Button 2"); add(btn2);
        btn3 = new JButton("3"); add(btn3);
        btn4 = new JButton("Another Button 4"); add(btn4);
        btn5 = new JButton("Button 5"); add(btn5);
        btn6 = new JButton("One More Button 6"); add(btn6);
    }

    public static void main(String[] args) { ... }
}
```

3 x 2 grid layout (3 rows, 2 columns)
Horizontal and vertical gaps between components: 3 pixels

GridLayout



- Places components into rows and columns

BorderLayout

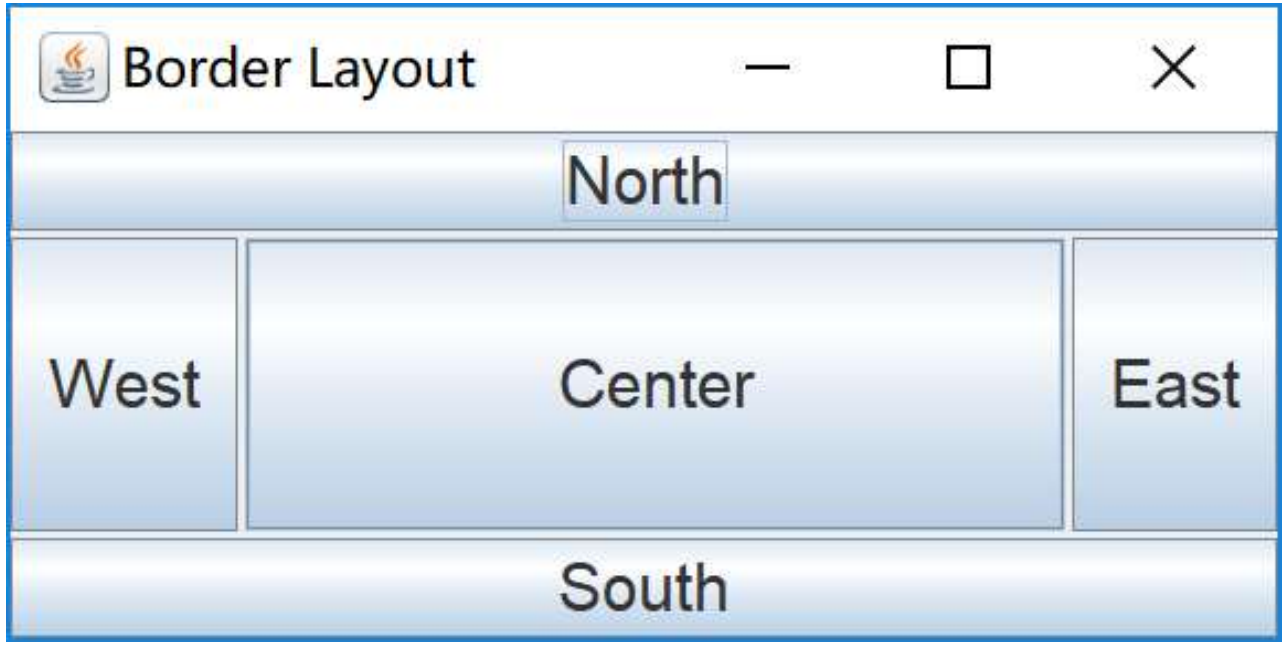
```
public class BorderLayoutDemo extends JFrame {
  private JButton btnNorth, btnSouth, btnCenter, btnEast, btnWest;

  public BorderLayoutDemo() {
    super("Border Layout");
    setLayout(new BorderLayout(3, 3));
    btnNorth = new JButton("North"); add(btnNorth, BorderLayout.NORTH);
    btnSouth = new JButton("South"); add(btnSouth, BorderLayout.SOUTH);
    btnCenter = new JButton("Center"); add(btnCenter, BorderLayout.CENTER);
    btnEast = new JButton("East"); add(btnEast, BorderLayout.EAST);
    btnWest = new JButton("West"); add(btnWest, BorderLayout.WEST);
  }

  public static void main(String[] args) { ... }
}
```

Horizontal and vertical gaps: 3 pixels

BorderLayout



- Default layout manager for the content pane of top level container `javax.swing.JFrame`
- Arranges the GUI components into five pre-defined areas: NORTH, SOUTH, EAST, WEST, CENTER

Using secondary containers for layout management

```
public class LayoutDemo extends JFrame {
    private JButton btn1, btn2, btn3, btn4, btn5, btn6;

    public LayoutDemo() {
        super("Layout demo");
        setLayout(new GridLayout(2, 1));
        JPanel panel1 = new JPanel(new FlowLayout());
        JPanel panel2 = new JPanel(new GridLayout(2, 2, 3, 3));
        add(panel1); add(panel2);
        btn1 = new JButton("Button 1"); panel1.add(btn1);
        btn2 = new JButton("This is Button 2"); panel1.add(btn2);
        btn3 = new JButton("Button 3"); panel2.add(btn3);
        btn4 = new JButton("Button 4"); panel2.add(btn4);
        btn5 = new JButton("Button 5"); panel2.add(btn5);
        btn6 = new JButton("Button 6"); panel2.add(btn6);
    }
    public static void main(String[] args) {...}
}
```

Create two `JPanel`s

Group buttons

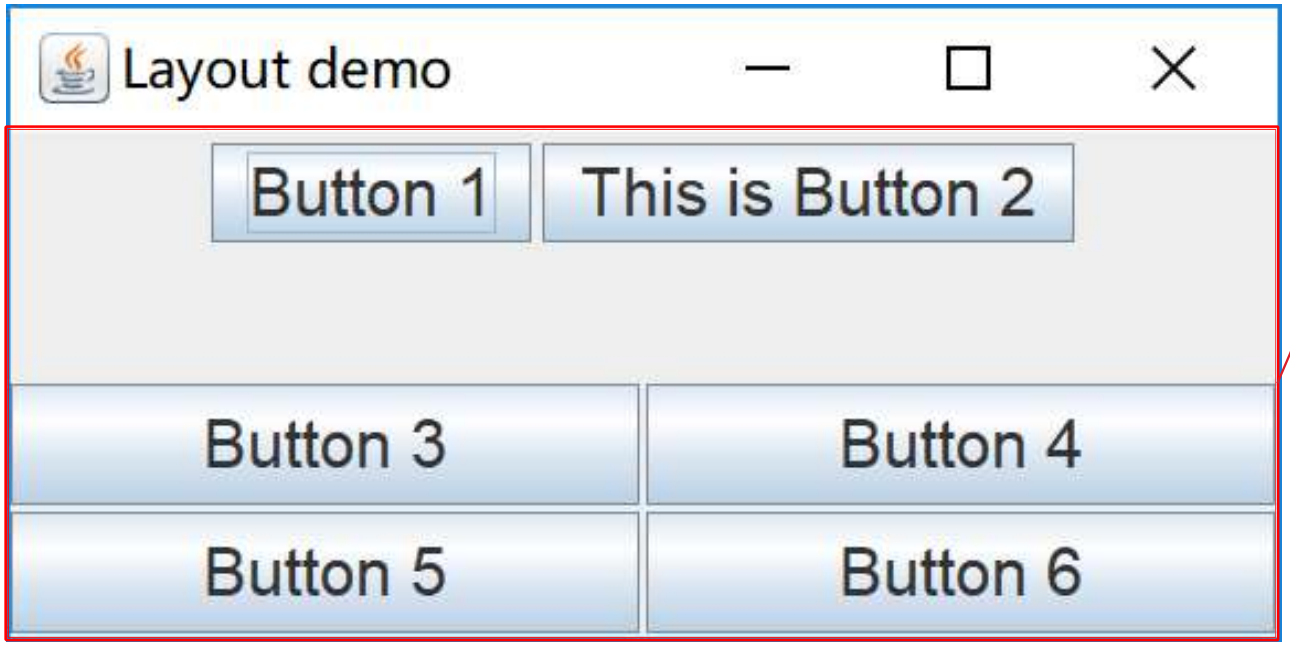
Using secondary containers for layout management

```
public class LayoutDemo extends JFrame {
    private JButton btn1, btn2, btn3, btn4, btn5, btn6;

    public LayoutDemo() {
        super("Layout demo");
        setLayout(new GridLayout(2, 1)); // Set the layout of JFrame's content pane
        JPanel panel1 = new JPanel(new FlowLayout());
        JPanel panel2 = new JPanel(new GridLayout(2, 2, 3, 3));
        add(panel1); add(panel2);  // add the two JPanels to the JFrame
        btn1 = new JButton("Button 1"); panel1.add(btn1);
        btn2 = new JButton("This is Button 2"); panel1.add(btn2);
        btn3 = new JButton("Button 3"); panel2.add(btn3);
        btn4 = new JButton("Button 4"); panel2.add(btn4);
        btn5 = new JButton("Button 5"); panel2.add(btn5);
        btn6 = new JButton("Button 6"); panel2.add(btn6);
    }
    public static void main(String[] args) {...}
}
```

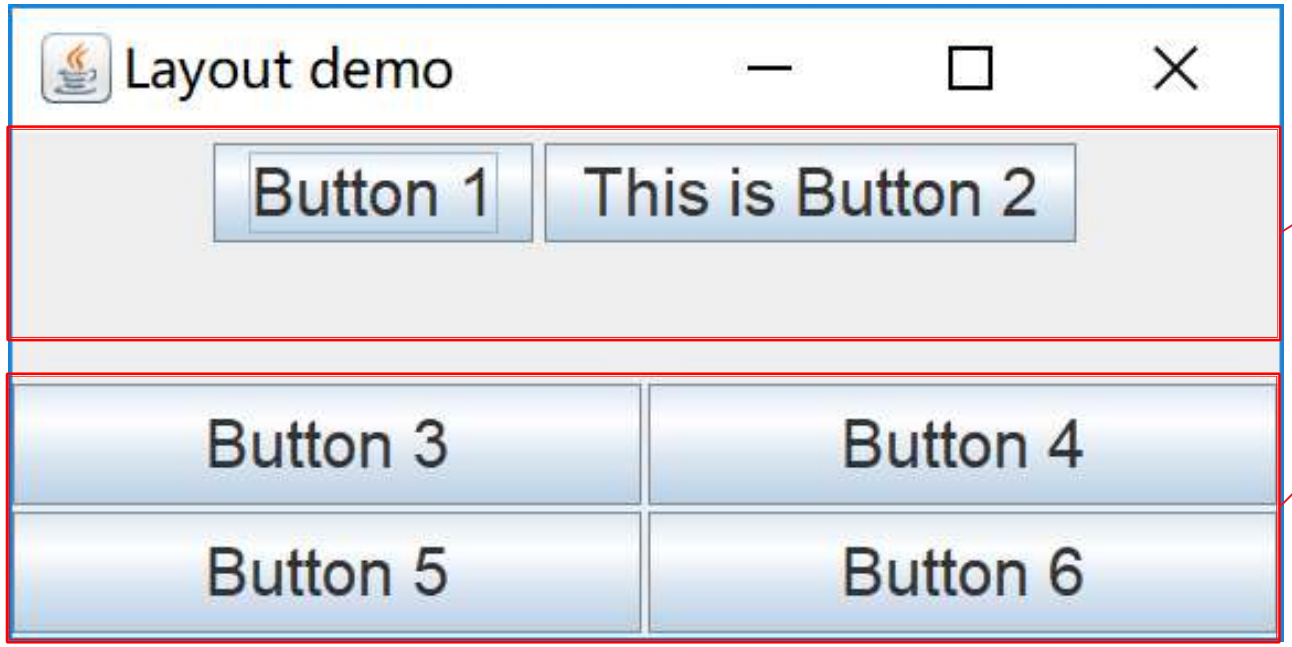
Set layout for the JPanels

Using secondary containers for layout management



JFrame's content pane
(grid layout, 2 rows, 1 col)

Using secondary containers for layout management



JPanel 1 contains two buttons (flow layout)

JPanel 2 contains 4 buttons (grid layout, 2 rows, 2 cols)

Dialogs (对话框)

- A Dialog window is an independent sub window meant to carry temporary notice apart from the main Swing Application Window
- Most Dialogs present an error message or warning to a user, but Dialogs can present images, directory trees, or just about anything compatible with the main Swing Application that manages them.

- To create simple, standard dialogs (标准对话框), you use the `JOptionPane` class
- To create a custom dialog (自定义对话框), use the `JDialog` class directly.

https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html

JOptionPane

- `JOptionPane` is a widely-used Swing class for popping up a dialog box that prompts users for a value or informs them of something.
- Commonly used static methods

Method Name	Description
showConfirmDialog	Asks a confirming question, like yes/no/cancel.
showInputDialog	Prompt for some input.
showMessageDialog	Tell the user about something that has happened.
showOptionDialog	The Grand Unification of the above three.

JOptionPane

- `JOptionPane` is a widely-used Swing class for popping up a dialog box that prompts users for a value or informs them of something.

```
public static void main(String[] args) {
    String str1 = JOptionPane.showInputDialog("Enter 1st integer");
    String str2 = JOptionPane.showInputDialog("Enter 2nd integer");
    int num1 = Integer.parseInt(str1);
    int num2 = Integer.parseInt(str2);
    int sum = num1 + num2;
    JOptionPane.showMessageDialog(null, num1 + " + " + num2 + " = " + sum);
}
```

JOptionPane

- `JOptionPane` is a widely-used Swing class for popping up a dialog box that prompts users for a value or informs them of something.

Static method `showInputDialog()` prompts for user input

```
public static void main(String[] args) {
    String str1 = JOptionPane.showInputDialog("Enter 1st integer");
    String str2 = JOptionPane.showInputDialog("Enter 2nd integer");
    int num1 = Integer.parseInt(str1);
    int num2 = Integer.parseInt(str2);
    int sum = num1 + num2;
    JOptionPane.showMessageDialog(null, num1 + " + " + num2 + " = " + sum);
}
```

`null` will be read

"123" will be read as a string

JOptionPane

- `JOptionPane` is a widely-used Swing class for popping up a dialog box that prompts users for a value or informs them of something.

```
public static void main(String[] args) {
    String str1 = JOptionPane.showInputDialog("Enter 1st integer");
    String str2 = JOptionPane.showInputDialog("Enter 2nd integer");
    int num1 = Integer.parseInt(str1);
    int num2 = Integer.parseInt(str2);
    int sum = num1 + num2;
    JOptionPane.showMessageDialog(null, num1 + " + " + num2 + " = " + sum);
}
```

Static method `showMessageDialog()` tells user about something that has happened