Lecture 4 - Java Graphical User Interface (GUI): Java Swing

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

CSC-1004: Computational Laboratory Using Java
Course Page: [Click]

Outline

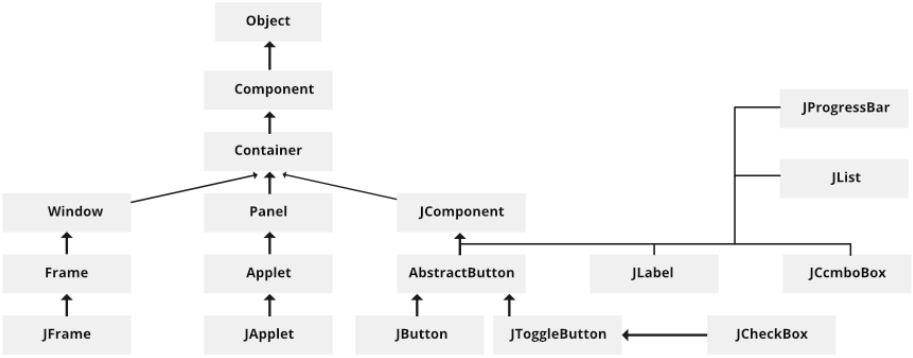
- Java Abstract Window Toolkit (AWT)
- Java Swing
- Java FX

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java Swing

- Java Swing is an extension of the AWT and is entirely written in java.
- Unlike AWT, Java Swing provides platform-independent and lightweight components.
- Swing offers much-improved functionality over AWT, new components, expanded components features, and excellent event handling with drag-and-drop support.

Java Swing


Object
Component
Container
Window
Panel
JComponent
JProgressBar
JList
Frame
Applet
AbstractButton
JLabel
JCcmboBox
JFrame
JApplet
JButton
JToggleButton
JCheckBox

Java Swing Example

A swing example where we create one button and add it to the JFrame object inside the main() method.

```
import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame

JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height

f.add(b);//adding button in JFrame

f.setSize(400,500);//400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
```

Java Swing JLabel

The object of JLabel class is a component for placing text in a container. It is used to display a single line of read-only text. The text can be changed by an application but a user cannot edit it directly.

```
import javax.swing.*;
class LabelExample
{
public static void main(String args[])
  {
  JFrame f= new JFrame("Label Example");
  JLabel l1,l2;
  l1=new JLabel("First Label.");
  l1.setBounds(50,50, 100,30);
  l2=new JLabel("Second Label.");
  l2.setBounds(50,100, 100,30);
  f.add(l1); f.add(l2);
  f.setSize(300,300);
  f.setLayout(null);
  f.setVisible(true);
  }
  }
```

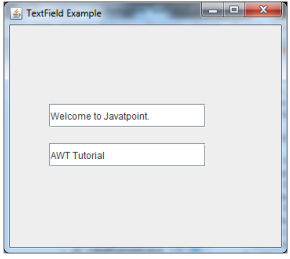
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java Swing JTextField

The object of JTextField class is a text component that allows the editing of a single-line text.

```
import javax.swing.*;
class TextFieldExample
{
public static void main(String args[])
  {
  JFrame f= new JFrame("TextField Example");
  JTextField t1,t2;
  t1=new JTextField("Welcome to Javatpoint.");
  t1.setBounds(50,100, 200,30);
```

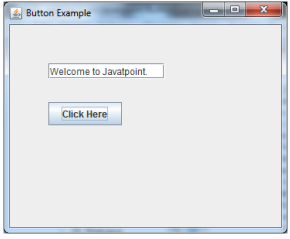
```
  t2=new JTextField("AWT Tutorial");
  t2.setBounds(50,150, 200,30);
  f.add(t1); f.add(t2);
  f.setSize(400,400);
  f.setLayout(null);
  f.setVisible(true);
  }
  }
```



Java Swing Button

The JButton class is used to create a labeled button that has a platform-independent implementation. The application results in some action when the button is pushed. It inherits AbstractButton class.

```
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    final JTextField tf=new JTextField();
    tf.setBounds(50,50, 150,20);
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    b.addActionListener(new ActionListener(){
public void actionPerformed(ActionEvent e){
        tf.setText("Welcome to Javatpoint.");
      }
    });
    f.add(b);f.add(tf);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```

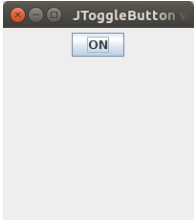


Java Swing JToggleButton

JToggleButton is used to create a toggle button, it is a two-state button to switch on or off.

```
public class JToggleButtonExample extends JFrame implements ItemListener {
  public static void main(String[] args) {
    new JToggleButtonExample();
  }
  private JToggleButton button;
  JToggleButtonExample() {
    setTitle("JToggleButton with ItemListener Example");
    setLayout(new FlowLayout());
    setJToggleButton();
    setAction();
    setSize(200, 200);
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
```

```
  private void setJToggleButton() {
    button = new JToggleButton("ON");
    add(button);
  }
  private void setAction() {
    button.addItemListener(this);
  }
  public void itemStateChanged(ItemEvent eve) {
    if (button.isSelected())
      button.setText("OFF");
    else
      button.setText("ON");
  }
}
```



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java Swing JPasswordField

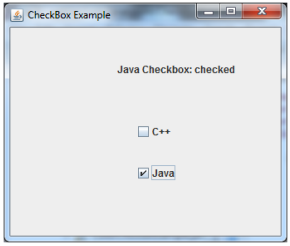
The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

```
import javax.swing.*;
public class PasswordFieldExample {
   public static void main(String[] args) {
   JFrame f=new JFrame("Password Field Example");
    JPasswordField value = new JPasswordField();
    JLabel l1=new JLabel("Password:");
      l1.setBounds(20,100, 80,30);
       value.setBounds(100,100,100,30);
         f.add(value);  f.add(l1);
         f.setSize(300,300);
         f.setLayout(null);
         f.setVisible(true);
}
}
```

Java Swing JCheckBox

The JCheckBox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on ".It inherits JToggleButton class.

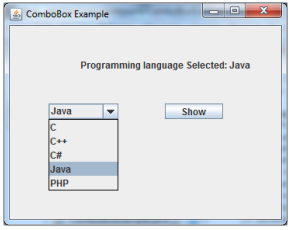
```
public class CheckBoxExample
{
    CheckBoxExample(){
      JFrame f= new JFrame("CheckBox Example");
      final JLabel label = new JLabel();
      label.setHorizontalAlignment(JLabel.CENTER);
      label.setSize(400,100);
      JCheckBox checkbox1 = new JCheckBox("C++");
      checkbox1.setBounds(150,100, 50,50);
      JCheckBox checkbox2 = new JCheckBox("Java");
      checkbox2.setBounds(150,150, 50,50);
      f.add(checkbox1); f.add(checkbox2); f.add(label);
      checkbox1.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
          label.setText("C++ Checkbox: "
          + (e.getStateChange()==1?"checked":"unchecked"));
        }
      });
      checkbox2.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
          label.setText("Java Checkbox: "
          + (e.getStateChange()==1?"checked":"unchecked"));
        }
      });
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
    }
public static void main(String args[])
{
    new CheckBoxExample();
}
}
```



Java Swing JComboBox

The object of the Choice class is used to show a popup menu of choices. The choice selected by the user is shown at the top of the menu. It inherits JComponent class.

```
public class ComboBoxExample {
JFrame f;
ComboBoxExample(){
  f=new JFrame("ComboBox Example");
  final JLabel label = new JLabel();
  label.setHorizontalAlignment(JLabel.CENTER);
  label.setSize(400,100);
  JButton b=new JButton("Show");
  b.setBounds(200,100,75,20);
  String languages[]={"C","C++","C#","Java","PHP"};
  final JComboBox cb=new JComboBox(languages);
  cb.setBounds(50, 100,90,20);
  f.add(cb); f.add(label); f.add(b);
  f.setLayout(null);
  f.setSize(350,350);
  f.setVisible(true);
  b.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
String data = "Programming language Selected: "
  + cb.getItemAt(cb.getSelectedIndex());
label.setText(data);
}
});
}
public static void main(String[] args) {
  new ComboBoxExample();
}
}
```

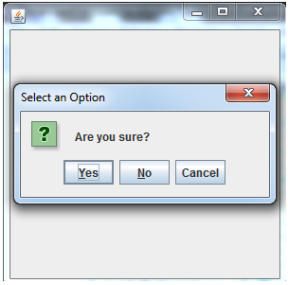


Java Swing JOptionPane

The JOptionPane class is used to provide standard dialog boxes such as the message dialog box, confirm dialog box, and input dialog box. The JOptionPane class inherits JComponent class.

```
public class OptionPaneExample extends WindowAdapter{
JFrame f;
OptionPaneExample(){
   f=new JFrame();
   f.addWindowListener(this);
   f.setSize(300, 300);
   f.setLayout(null);
   f.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
   f.setVisible(true);
}
```

```
public void windowClosing(WindowEvent e) {
   int a=JOptionPane.showConfirmDialog(f,"Are you sure?");
if(a==JOptionPane.YES_OPTION){
   f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
public static void main(String[] args) {
   new  OptionPaneExample();
}
}
```

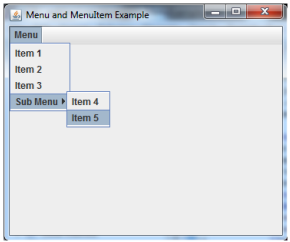


Java Swing JMenuBar, JMenu and JMenuItem

The JMenuBar class is used to display menubar on the window or frame. The object of JMenu class is a pull-down menu component that is displayed from the menu bar. The object of JMenuItem class adds a simple labeled menu item.

```
import javax.swing.*;
class MenuExample
{
    JMenu menu, submenu;
    JMenuItem i1, i2, i3, i4, i5;
    MenuExample(){
    JFrame f= new JFrame("Menu and MenuItem Example");
    JMenuBar mb=new JMenuBar();
    menu=new JMenu("Menu");
    submenu=new JMenu("Sub Menu");
    i1=new JMenuItem("Item 1");
    i2=new JMenuItem("Item 2");
    i3=new JMenuItem("Item 3");
    i4=new JMenuItem("Item 4");
    i5=new JMenuItem("Item 5");
```

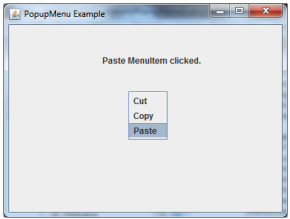
```
    menu.add(i1); menu.add(i2); menu.add(i3);
    submenu.add(i4); submenu.add(i5);
    menu.add(submenu);
    mb.add(menu);
    f.setJMenuBar(mb);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
public static void main(String args[])
{
new MenuExample();
}}
```



Java JPopupMenu

PopupMenu can be dynamically popped up at specific position within a component. It inherits the JComponent class.

```
import javax.swing.*;
import java.awt.event.*;
class PopupMenuExample
{
    PopupMenuExample(){
      final JFrame f= new JFrame("PopupMenu Example");
      final JLabel label = new JLabel();
      label.setHorizontalAlignment(JLabel.CENTER);
      label.setSize(400,100);
      final JPopupMenu popupmenu = new JPopupMenu("Edit");
      JMenuItem cut = new JMenuItem("Cut");
      JMenuItem copy = new JMenuItem("Copy");
      JMenuItem paste = new JMenuItem("Paste");
      popupmenu.add(cut); popupmenu.add(copy); popupmenu.add(paste);
      f.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent e) {
          popupmenu.show(f , e.getX(), e.getY());
        }
      });
      cut.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e) {
        label.setText("cut MenuItem clicked.");
      }
      });
      copy.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
          label.setText("copy MenuItem clicked.");
        }
        });
      paste.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
          label.setText("paste MenuItem clicked.");
        }
        });
      f.add(label); f.add(popupmenu);
      f.setSize(400,400);
      f.setLayout(null);
      f.setVisible(true);
    }
public static void main(String args[])
{
    new PopupMenuExample();
}
}
```



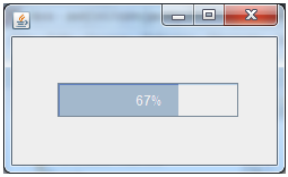
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java Swing JProgressBar

The JProgressBar class is used to display the progress of the task.

```
import javax.swing.*;
public class ProgressBarExample extends JFrame{
JProgressBar jb;
int i=0,num=0;
ProgressBarExample(){
jb=new JProgressBar(0,2000);
jb.setBounds(40,40,160,30);
jb.setValue(0);
jb.setStringPainted(true);
add(jb);
setSize(250,150);
setLayout(null);
}
```

```
public void iterate(){
while(i<=2000){
 jb.setValue(i);
 i=i+20;
 try{Thread.sleep(150);}catch(Exception e){}
}
}
public static void main(String[] args) {
   ProgressBarExample m=new ProgressBarExample();
   m.setVisible(true);
   m.iterate();
}
}
```



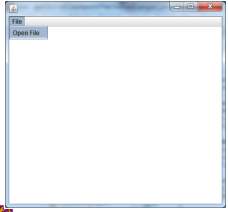
Java Swing JFileChooser

The JFileChooser class represents a dialog window from which the user can select a file.

```
public class FileChooserExample extends JFrame implements ActionListener{
JMenuBar mb;
JMenu file;
JMenuItem open;
JTextArea ta;
FileChooserExample(){
open=new JMenuItem("Open File");
open.addActionListener(this);
file=new JMenu("File");
file.add(open);
mb=new JMenuBar();
mb.setBounds(0,0,800,20);
mb.add(file);
ta=new JTextArea(800,800);
ta.setBounds(0,20,800,800);
add(mb);
add(ta);
}
```

```
public void actionPerformed(ActionEvent e) {
if(e.getSource()==open){
  JFileChooser fc=new JFileChooser();
  int i=fc.showOpenDialog(this);
  if(i==JFileChooser.APPROVE_OPTION){
    File f=fc.getSelectedFile();
    String filepath=f.getPath();
    try{
    BufferedReader br=new BufferedReader(new FileReader(filepath));
    String s1="",s2="";
    while((s1=br.readLine())!=null){
    s2+=s1+"\n";
    }
    ta.setText(s2);
    br.close();
    }catch (Exception ex) {ex.printStackTrace(); }
  }
}
}
```

```
public static void main(String[] args) {
  FileChooserExample om=new FileChooserExample();
        om.setSize(500,500);
        om.setLayout(null);
        om.setVisible(true);
        om.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}
```



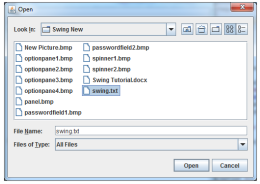
Java Swing JFileChooser

The JFileChooser class represents a dialog window from which the user can select a file.

```
public class FileChooserExample extends JFrame implements ActionListener{
JMenuBar mb;
JMenu file;
JMenuItem open;
JTextArea ta;
FileChooserExample(){
open=new JMenuItem("Open File");
open.addActionListener(this);
file=new JMenu("File");
file.add(open);
mb=new JMenuBar();
mb.setBounds(0,0,800,20);
mb.add(file);
ta=new JTextArea(800,800);
ta.setBounds(0,20,800,800);
add(mb);
add(ta);
}
```

```
public void actionPerformed(ActionEvent e) {
if(e.getSource()==open){
  JFileChooser fc=new JFileChooser();
  int i=fc.showOpenDialog(this);
  if(i==JFileChooser.APPROVE_OPTION){
    File f=fc.getSelectedFile();
    String filepath=f.getPath();
    try{
    BufferedReader br=new BufferedReader(new FileReader(filepath));
    String s1="",s2="";
    while((s1=br.readLine())!=null){
    s2+=s1+"\n";
    }
    ta.setText(s2);
    br.close();
    }catch (Exception ex) {ex.printStackTrace(); }
  }
}
}
```

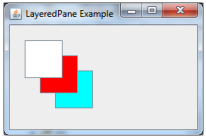
```
public static void main(String[] args) {
  FileChooserExample om=new FileChooserExample();
        om.setSize(500,500);
        om.setLayout(null);
        om.setVisible(true);
        om.setDefaultCloseOperation(EXIT_ON_CLOSE);
}
}
```



Java JLayeredPane

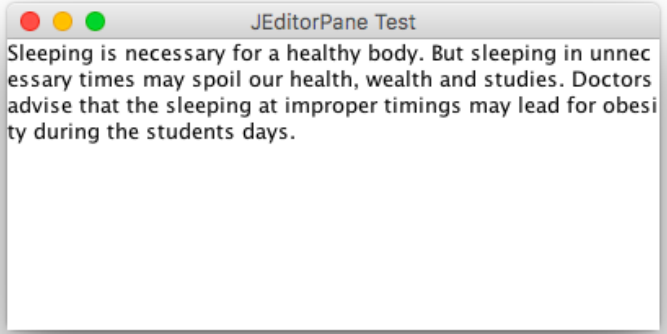
The JLayeredPane class is used to add depth to swing container. It is used to provide a third dimension for positioning components and dividing layers.

```
import javax.swing.*;
import java.awt.*;
public class LayeredPaneExample extends JFrame {
 public LayeredPaneExample() {
  super("LayeredPane Example");
  setSize(200, 200);
  JLayeredPane pane = getLayeredPane();
  //creating buttons
  JButton top = new JButton();
  top.setBackground(Color.white);
  top.setBounds(20, 20, 50, 50);
  JButton middle = new JButton();
  middle.setBackground(Color.red);
  middle.setBounds(40, 40, 50, 50);
  JButton bottom = new JButton();
  bottom.setBackground(Color.cyan);
  bottom.setBounds(60, 60, 50, 50);
  //adding buttons on pane
  pane.add(bottom, new Integer(1));
  pane.add(middle, new Integer(2));
  pane.add(top, new Integer(3));
 }
 public static void main(String[] args) {
   LayeredPaneExample panel = new  LayeredPaneExample();
   panel.setVisible(true);
 }
}
```



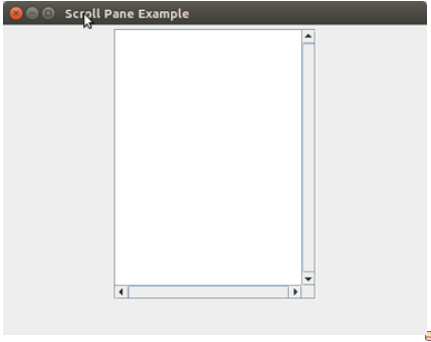
JJava JEditorPane

JEditorPane class is used to create a simple text editor window. This class has setContentType() and setText() methods.



Java JScrollPane

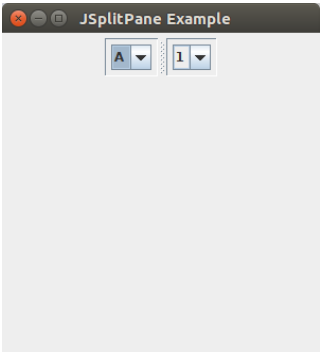
A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java JSplitPane

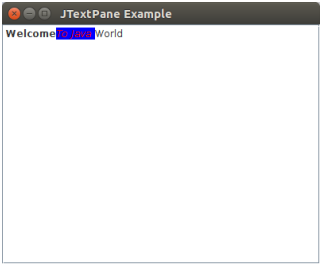
JSplitPane is used to divide two components. The two components are divided based on the look and feel implementation, and they can be resized by the user. If the minimum size of the two components is greater than the size of the split pane, the divider will not allow you to resize it.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java JTextPane

JTextPane is a subclass of JEditorPane class. JTextPane is used for styled documents with embedded images and components. It is a text component that can be marked up with attributes that are represented graphically. JTextPane uses a DefaultStyledDocument as its default model.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java BorderLayout

The BorderLayout is used to arrange the components in five regions: north, south, east, west, and center. Each region (area) may contain one component only.

```
public class Border
{
JFrame f;
Border()
{
   f = new JFrame();

    // creating buttons
   JButton b1 = new JButton("NORTH");;; // the button will be labeled as NORTH
   JButton b2 = new JButton("SOUTH");;; // the button will be labeled as SOUTH
   JButton b3 = new JButton("EAST");;; // the button will be labeled as EAST
   JButton b4 = new JButton("WEST");;; // the button will be labeled as WEST
   JButton b5 = new JButton("CENTER");;; // the button will be labeled as CENTER
```

```
   f.add(b1, BorderLayout.NORTH); // b1 will be placed in the North Direction
   f.add(b2, BorderLayout.SOUTH);  // b2 will be placed in the South Direction
   f.add(b3, BorderLayout.EAST);  // b2 will be placed in the East Direction
   f.add(b4, BorderLayout.WEST);  // b2 will be placed in the West Direction
   f.add(b5, BorderLayout.CENTER);  // b2 will be placed in the Center

   f.setSize(300, 300);
   f.setVisible(true);
}
public static void main(String[] args) {
   new Border();
}
}
```

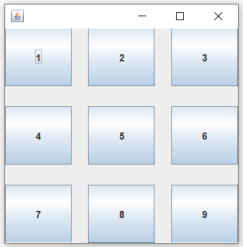
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java GridLayout

The Java GridLayout class is used to arrange the components in a rectangular grid.

```
// import statements
import java.awt.*;
import javax.swing.*;

public class GridLayoutExample1
{

JFrame frameObj;

// constructor
GridLayoutExample1()
{
frameObj = new JFrame();

// creating 9 buttons
JButton btn1 = new JButton("1");
JButton btn2 = new JButton("2");
JButton btn3 = new JButton("3");
JButton btn4 = new JButton("4");
JButton btn5 = new JButton("5");
JButton btn6 = new JButton("6");
JButton btn7 = new JButton("7");
JButton btn8 = new JButton("8");
JButton btn9 = new JButton("9");
```

```
// adding buttons to the frame
// since, we are using the parameterless constructor, therefore;
// the number of columns is equal to the number of buttons we
// are adding to the frame. The row count remains one.
frameObj.add(btn1); frameObj.add(btn2); frameObj.add(btn3);
frameObj.add(btn4); frameObj.add(btn5); frameObj.add(btn6);
frameObj.add(btn7); frameObj.add(btn8); frameObj.add(btn9);
// setting the grid layout
// a 3 * 3 grid is created with the horizontal gap 20
// and vertical gap 25
frameObj.setLayout(new GridLayout(3, 3, 20, 25));
frameObj.setSize(300, 300);
frameObj.setVisible(true);
}
// main method
public static void main(String argvs[])
{
new GridLayoutExample();
}
}
```



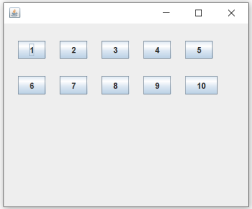
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java FlowLayout

The Java FlowLayout class is used to arrange the components in a line.

```
public class FlowLayoutExample1
{
JFrame frameObj;

// constructor
FlowLayoutExample1()
{
   // creating a frame object
   frameObj = new JFrame();

    // creating the buttons
   JButton b1 = new JButton("1");
   JButton b2 = new JButton("2");
   JButton b3 = new JButton("3");
   JButton b4 = new JButton("4");
   JButton b5 = new JButton("5");
   JButton b6 = new JButton("6");
   JButton b7 = new JButton("7");
   JButton b8 = new JButton("8");
   JButton b9 = new JButton("9");
   JButton b10 = new JButton("10");
```

```
   // adding the buttons to frame
   frameObj.add(b1); frameObj.add(b2); frameObj.add(b3); frameObj.add(b4);
   frameObj.add(b5); frameObj.add(b6);  frameObj.add(b7);  frameObj.add(b8);
   frameObj.add(b9);  frameObj.add(b10);

   // parameterized constructor is used
   // where alignment is left
   // horizontal gap is 20 units and vertical gap is 25 units.
   frameObj.setLayout(new FlowLayout(FlowLayout.LEFT, 20, 25));


   frameObj.setSize(300, 300);
   frameObj.setVisible(true);
}
// main method
public static void main(String argvs[])
{
   new FlowLayoutExample1();
}
}
```



Java GridBagLayout

The Java GridBagLayout class is used to align components vertically, horizontally or along their baseline.

```
import java.awt.Button;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import javax.swing.*;
public class GridBagLayoutExample extends JFrame{
  public static void main(String[] args) {
       GridBagLayoutExample a = new GridBagLayoutExample();
    }
    public GridBagLayoutExample() {
  GridBagLayoutgrid = new GridBagLayout();
       GridBagConstraints gbc = new GridBagConstraints();
       setLayout(grid);
       setTitle("GridBag Layout Example");
       GridBagLayout layout = new GridBagLayout();
  this.setLayout(layout);
  gbc.fill = GridBagConstraints.HORIZONTAL;
  gbc.gridx = 0;
  gbc.gridy = 0;
  this.add(new Button("Button One"), gbc);
  gbc.gridx = 1;
  gbc.gridy = 0;
  this.add(new Button("Button two"), gbc);
  gbc.fill = GridBagConstraints.HORIZONTAL;
  gbc.ipady = 20;
  gbc.gridx = 0;
  gbc.gridy = 1;
  this.add(new Button("Button Three"), gbc);
  gbc.gridx = 1;
  gbc.gridy = 1;
  this.add(new Button("Button Four"), gbc);
  gbc.gridx = 0;
  gbc.gridy = 2;
  gbc.fill = GridBagConstraints.HORIZONTAL;
  gbc.gridwidth = 2;
  this.add(new Button("Button Five"), gbc);
       setSize(300, 300);
       setPreferredSize(getSize());
       setVisible(true);
       setDefaultCloseOperation(EXIT_ON_CLOSE);

    }

}
```

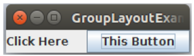
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java GroupLayout

GroupLayout groups its components and places them in a Container hierarchically. The grouping is done by instances of the Group class.

```
// import statements
import java.awt.*;
import javax.swing.*;
public class GroupExample {
   public static void main(String[] args) {
      JFrame frame = new JFrame("GroupLayoutExample");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Container contentPanel = frame.getContentPane();
      GroupLayout groupLayout = new GroupLayout(contentPanel);

      contentPanel.setLayout(groupLayout);

      JLabel clickMe = new JLabel("Click Here");
      JButton button = new JButton("This Button");
```

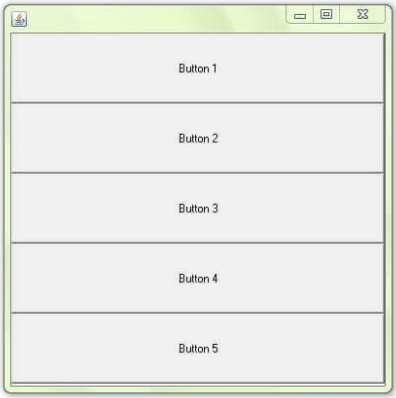
```
      groupLayout.setHorizontalGroup(
            groupLayout.createSequentialGroup()
                  .addComponent(clickMe)
                  .addGap(10, 20, 100)
                  .addComponent(button));
      groupLayout.setVerticalGroup(
             groupLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
                  .addComponent(clickMe)
                  .addComponent(button));
      frame.pack();
      frame.setVisible(true);
   }
}
```



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java BoxLayout

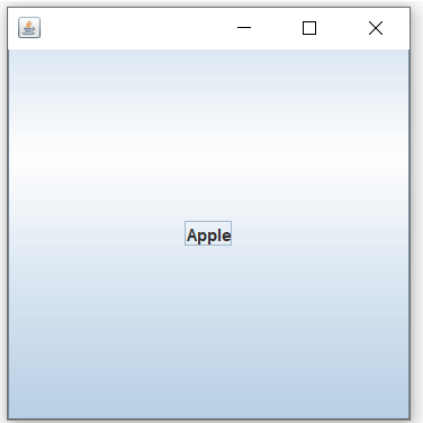
The Java BoxLayout class is used to arrange the components either vertically or horizontally.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java CardLayout

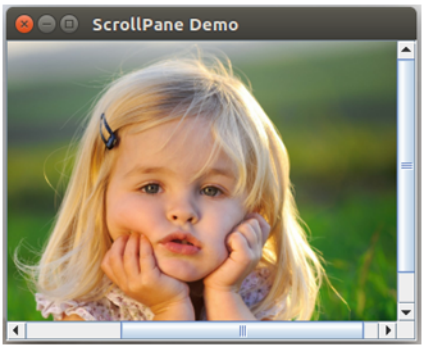
The Java CardLayout class manages the components in such a manner that only one component is visible at a time.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Java ScrollPaneLayout

The layout manager is used by JScrollPane. JScrollPaneLayout is responsible for nine components: a viewport, two scrollbars, a row header, a column header, and four "corner" components.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Real Examples

The real examples:

Note that I realize our school's network won't allow you to visit the website, please try your mobile signal or even use the school's VPN Service.

- Notepad.
- Calculator.
- IP-finder.
- Word-count.
- Folder-explorer.
- Puzzle-Game.
- Online-Exam.

香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen