NHẬP MÔN JAVA

BÀI 11

SWING

Các thành phần GUI Swing

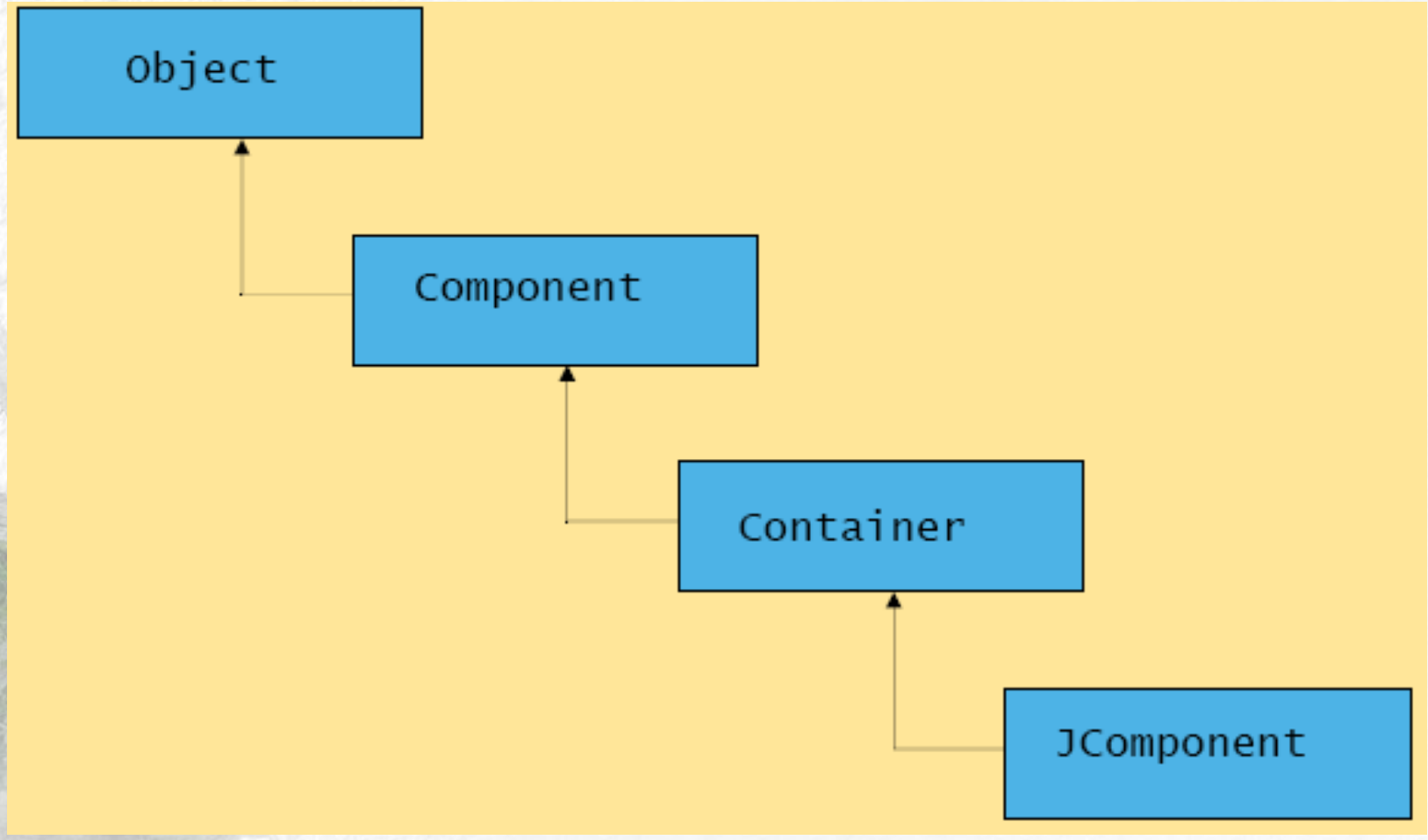
- Gói javax.swing.*
- Các thành phần bắt nguồn từ AWT (gói java.awt.*)
- Chứa đựng cảm quan (look and feel)
  - Sự thể hiện và cách người sử dụng tương tác với chương trình
- Những thành phần nhẹ (lightweight)
  - Được viết hoàn toàn bằng Java

Các thành phần GUI Swing

- Các thành phần
  - Chứa phương thức paint() để vẽ thành phần trên màn hình
- Các bộ chứa
  - Tập hợp các thành phần liên quan
  - Chứa phương thức add() để thêm các thành phần
- Lớp JComponent
  - Cảm quan khả kiến (Pluggable)
  - Phím tắt (tính dễ nhớ)
  - Khả năng xử lý sự kiện chung

Các thành phần GUI Swing

- Các siêu lớp của nhiều thành phần Swing



Các thành phần GUI cơ bản

- JLabel: Hiển thị văn bản hay những biểu tượng.
- JTextField: Trường nhập dữ liệu từ bàn phím, cũng có thể hiển thị thông tin.
- JButton: Nút nhấn dùng kích hoạt một sự kiện khi nhấp chuột.
- JCheckBox: Hộp kiểm tra cho phép được lựa chọn hay không được lựa chọn.
- …

Các thành phần GUI cơ bản

- JComboBox: Hộp danh mục thả xuống từ đó người sử dụng có thể chọn một bởi việc kích một mục trong danh sách hoặc nhập nội dung vào trong hộp.
- JList: Hộp danh sách từ đó người sử dụng có thể chọn bởi việc nhấp vào một mục trong danh sách. Có thể chọn nhiều mục.
- JPanel: Một Container trong đó những thành phần có thể được đặt và cách trình bày.

JLabel

- Cung cấp văn bản trên GUI
- Được định nghĩa với lớp JLabel
- Có thể trình bày :
  - Dòng văn bản chỉ đọc
  - Hình ảnh
  - Văn bản và hình ảnh

JLabel – Ví dụ

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class LabelTest extends JFrame {
   private JLabel label1, label2, label3;
   public LabelTest(){
      super( "Testing JLabel" );
      // get content pane and set its layout
      Container container = getContentPane();
      container.setLayout( new FlowLayout() );
      // JLabel constructor with a string argument
      label1 = new JLabel( "Label with text" );
      label1.setToolTipText( "This is label1" );
      container.add( label1 );
```

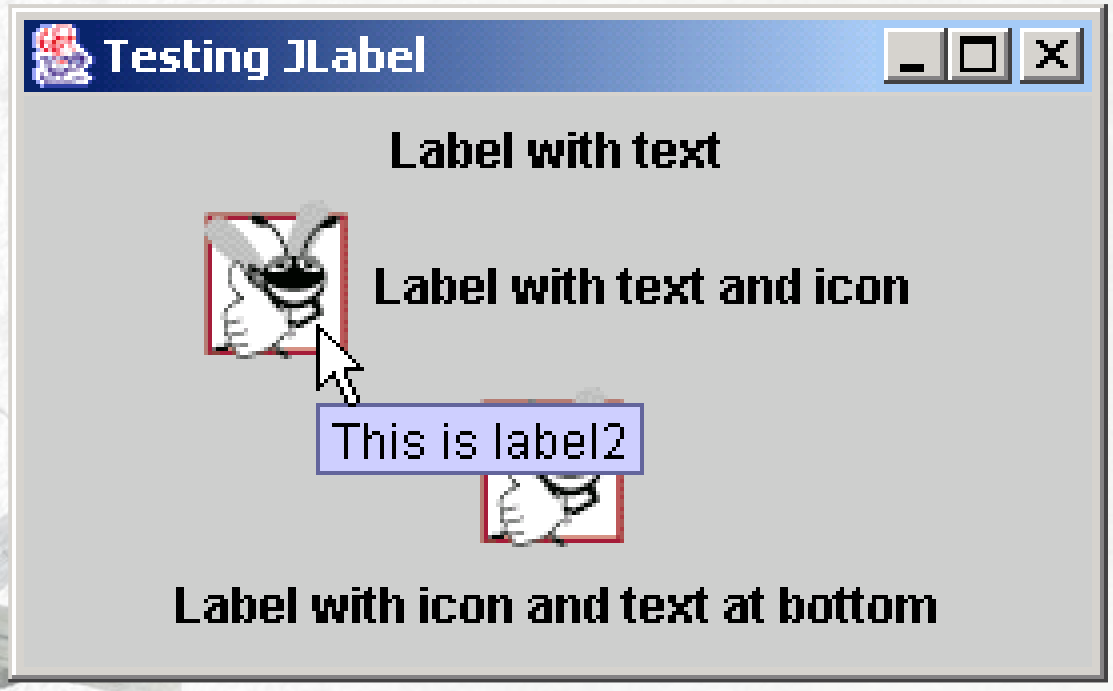
Khai báo JLabel

Tạo JLabel

JLabel – Ví dụ

```
    // JLabel constructor with string, Icon and alignment argum
    Icon bug = new ImageIcon( "bug1.gif" );
    label2 = new JLabel( "Label with text and icon", bug,
                         SwingConstants.LEFT );
    label2.setToolTipText( "This is label2" );
    container.add( label2 );
    // JLabel constructor no arguments
    label3 = new JLabel();
    label3.setText( "Label with icon and text at bottom" );
    label3.setIcon( bug );
    label3.setHorizontalTextPosition( SwingConstants.CENTER );
    label3.setVerticalTextPosition( SwingConstants.BOTTOM );
    label3.setToolTipText( "This is label3" );
    container.add( label3 );
}
```

JLabel – Ví dụ


Testing JLabel
Label with text
Label with text and icon
This is label2
Label with icon and text at bottom

Các JTextField

- JTextField
  - Hộp văn bản trong đó người sử dụng có thể nhập dữ liệu từ bàn phím
- JPasswordField
  - Mở rộng JTextField
  - Che giấu các ký tự mà người sử dụng nhập vào

Ví dụ sử dụng JTextField

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TextFieldTest extends JFrame implements ActionListener{
   private JTextField textField1, textField2, textField3;
   private JPasswordField passwordField;
   // set up GUI
   public TextFieldTest(){
      super( "Testing JTextField and JPasswordField" );
      Container container = getContentPane();
      container.setLayout( new FlowLayout() );
      // construct textfield with default sizing
      textField1 = new JTextField(10);
      container.add(textField1 );
      // construct textfield with default text
      textField2 = new JTextField( "Enter text here" );
      container.add( textField2 );
      ...
   }
   // process textfield events
   public void actionPerformed(ActionEvent event ){
      ...
   }
}
```

Ví dụ sử dụng JTextField

```
// construct textfield with default text,
// 20 visible elements and no event handler
textField3 = new JTextField( "Uneditable text field", 20 );
textField3.setEditable( false );
container.add( textField3 );

//construct passwordfield with default text
passwordField = new JPasswordField( "Hidden text" );
container.add( passwordField );
// register event handlers
textField1.addActionListener(this);
textField2.addActionListener(this);
textField3.addActionListener(this);
passwordField.addActionListener(this);
```

Ví dụ sử dụng JTextField

```
// process textfield events
public void actionPerformed(ActionEvent event ){
    String string = "";
    // user pressed Enter in JTextField textField1
    if ( event.getSource() == textField1 )
        string = "textField1: " + event.getActionCommand();
    // user pressed Enter in JTextField textField2
    else if ( event.getSource() == textField2 )
        string = "textField2: " + event.getActionCommand();
    // user pressed Enter in JTextField textField3
    else if ( event.getSource() == textField3 )
        string = "textField3: " + event.getActionCommand();
    // user pressed Enter in JTextField passwordField
    else if ( event.getSource() == passwordField ) {
        string = "passwordField: " +
                 new String( passwordField.getPassword() );
    }
    JOptionPane.showMessageDialog( null, string );
}
```

Ví dụ - kết quả


Testing JTextField and JPasswordField
Enter text here
Uneditable text field
****


Testing JTextField and JPasswordField
hello
Enter text here
Uneditable text field
****


Message
textField1: hello
OK


Testing JTextField and JPasswordField
hello
Enter text here
Uneditable text field
****


Message
passwordField: Hidden text
OK

JTextArea

- Vùng văn bản cho phép thao tác soạn thảo nhiều dòng văn bản.
- Thừa kế JTextComponent

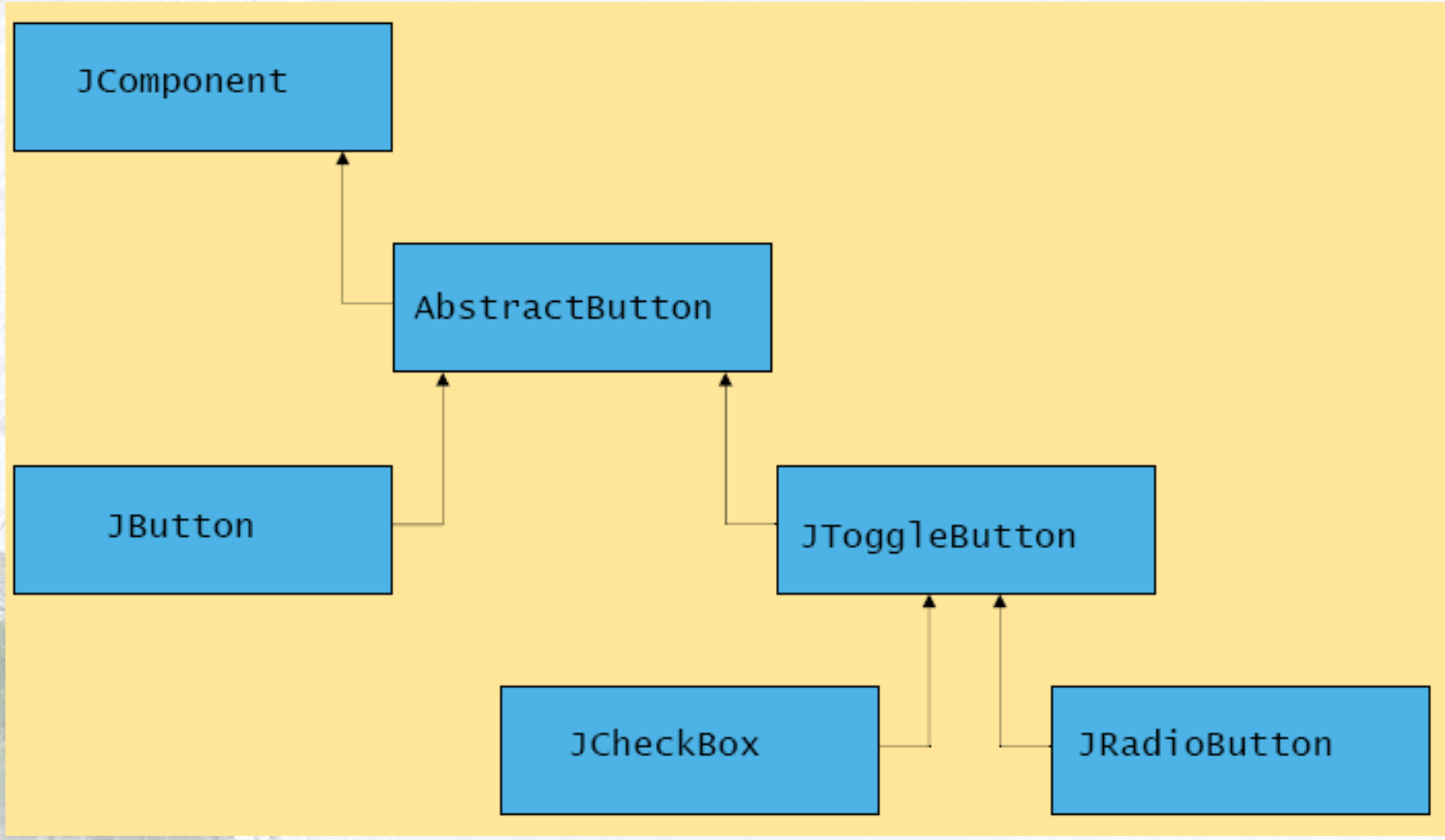
Ví dụ sử dụng JTextArea

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TextAreaDemo extends JFrame {
   private JTextArea textArea1, textArea2;
   // set up GUI
   public TextAreaDemo() {
      super( "TextArea Demo" );
      Box box = Box.createHorizontalBox();
      String string = "This is a demo string to\n" +
         "illustrate copying text\nfrom one textarea to \n" +
         "another textarea using an\nexternal event\n";
      // set up textArea1
      textArea1 = new JTextArea( string, 10, 15 );
      box.add( new JScrollPane( textArea1 ) );
      // add box to content pane
      Container container = getContentPane();
      container.add( box );
   } // end constructor TextAreaDemo
} // end class TextAreaDemo
```

JButton

- Nút nhấn - thành phần người sử dụng nhấp để kích hoạt một hành động cụ thể.
- Một vài kiểu khác nhau
  - Command Button
  - Check Box
  - Radio Button
  - …
- Các lớp dẫn xuất javax.swing.AbstractButton
  - Command Button được tạo với lớp JButton
- Sinh ra một ActionEvent khi người sử dụng nhấn trên nút.

Cây thừa kế các JButton



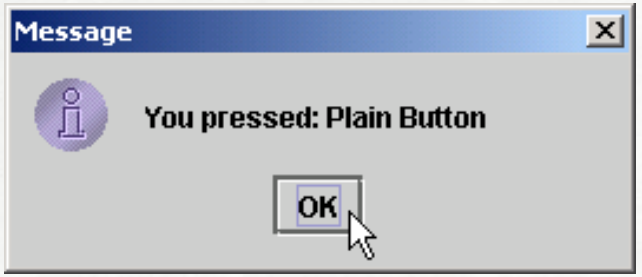
Ví dụ sử dụng JButton

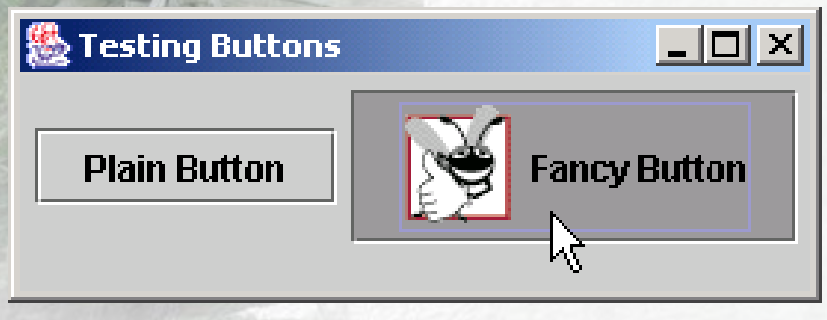
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ButtonTest extends JFrame {
   private JButton plainButton, fancyButton;
   // set up GUI
   public ButtonTest() {
      super( "Testing Buttons" );
      // get content pane and set its layout
      Container container = getContentPane();
      container.setLayout( new FlowLayout() );
      // create buttons
      plainButton = new JButton( "Plain Button" );
      container.add( plainButton );
      Icon bug1 = new ImageIcon( "bug1.gif" );
      Icon bug2 = new ImageIcon( "bug2.gif" );
      fancyButton = new JButton( "Fancy Button", bug1 );
      fancyButton.setRolloverIcon( bug2 );
      container.add( fancyButton );
```

Ví dụ sử dụng JButton

```
        // create an instance of inner class ButtonHandler
        // to use for button event handling
        ButtonHandler handler = new ButtonHandler();
        fancyButton.addActionListener( handler );
        plainButton.addActionListener( handler );
    }
    // inner class for button event handling
    private class ButtonHandler implements ActionListener {
        // handle button event
        public void actionPerformed( ActionEvent event ) {
            JOptionPane.showMessageDialog( ButtonTest.this,
                "You pressed: " + event.getActionCommand() );
        }
    } // end private inner class ButtonHandler
} // End ButtonTest class
```

Ví dụ JButton - kết quả





JCheckBox và JRadioButton

- Các nút trạng thái
  - Các giá trị On/Off hoặc true/false
- Java cung cấp 3 kiểu:
  - JToggleButton
  - JCheckBox
  - JRadioButton

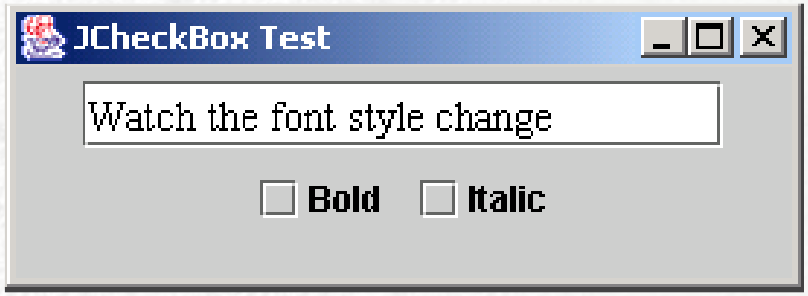
Ví dụ sử dụng JCheckBox

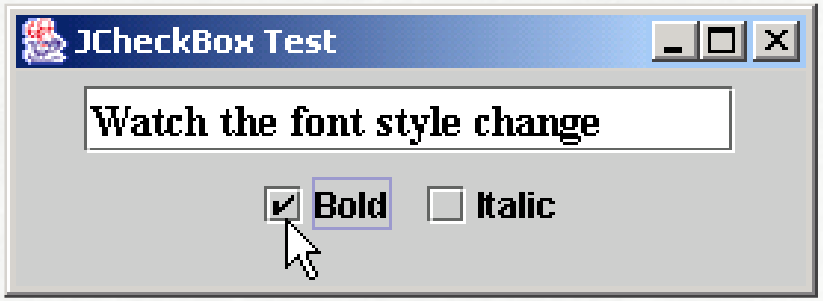
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//
public class CheckBoxTest extends JFrame implements ItemListener {
   private JTextField field;
   private JCheckBox bold, italic;
   // set up GUI
   public CheckBoxTest() {
      super( "JCheckBox Test" );
      // get content pane and set its layout
      Container container = getContentPane();
      container.setLayout( new FlowLayout() );
      // set up JTextField and set its font
      field = new JTextField( "Watch the font style change", 20 );
      field.setFont( new Font( "Serif", Font.PLAIN, 14 ) );
      container.add( field );
      // create checkbox objects
      bold = new JCheckBox( "Bold" );
      container.add( bold );
      italic = new JCheckBox( "Italic" );
      container.add( italic );
```

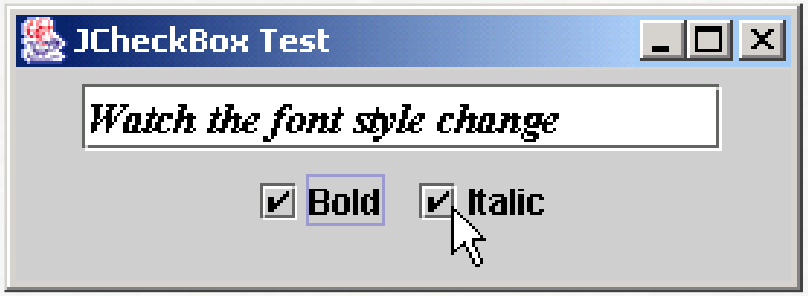
Ví dụ sử dụng JCheckBox

```
        // register listeners for JCheckBoxes
        bold.addItemListener(this);
        italic.addItemListener(this);

        // respond to checkbox events
        public void itemStateChanged( ItemEvent event ) {
            int valBold = Font.PLAIN;
            int valItalic = Font.PLAIN;
            // process bold checkbox events
            if ( event.getSource() == bold )
               valBold = bold.isSelected() ? Font.BOLD : Font.PLAIN;
            // process italic checkbox events
            if ( event.getSource() == italic )
               valItalic = italic.isSelected() ? Font.ITALIC : Font.PLAIN;
            // set text field font
            field.setFont( new Font( "Serif", valBold + valItalic, 14 ) );
        }
    }
}
```

Ví dụ JCheckBox - kết quả







JComboBox

- Hộp danh sách chứa các mục từ đó người sử dụng có thể lựa chọn một mục khi nhấp vào nó.
- Còn gọi là hộ danh sách thả xuống

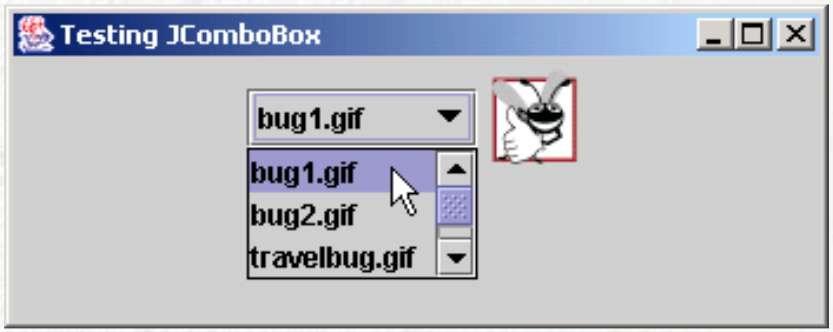
Ví dụ sử dụng JComboBox

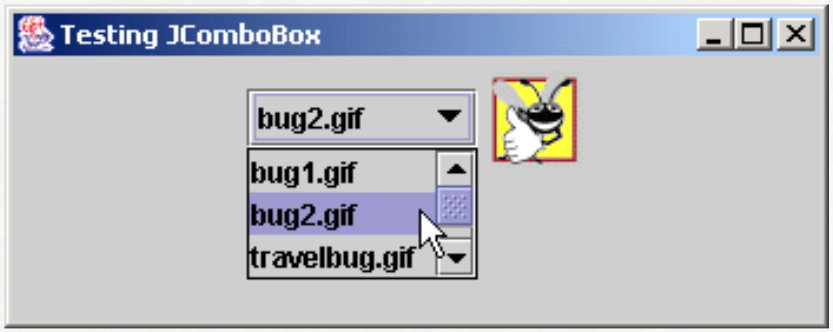
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class ComboBoxTest extends JFrame {
   private JComboBox imagesComboBox;
   private JLabel label;
   private String names[]={"bug1.gif", "bug2.gif", "bug3.gif", "bug4.gif"};
   private Icon icons[] = { new ImageIcon( names[0] ),
      new ImageIcon( names[1]), new ImageIcon( names[2] ),
      new ImageIcon( names[3]) };
   // set up GUI
   public ComboBoxTest() {
      super( "Testing JComboBox" );
      // get content pane and set its layout
      Container container = getContentPane();
      container.setLayout( new FlowLayout());
      // set up JLabel to display ImageIcons
     label = new JLabel( icons[0]);
     container.add( label );
```

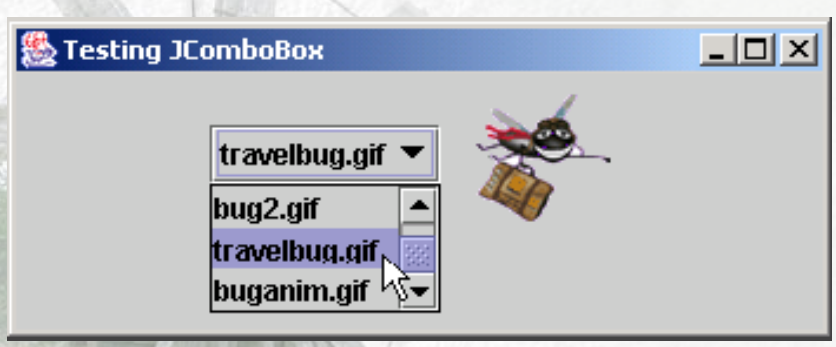
Ví dụ sử dụng JComboBox

```
      // set up JComboBox and register its event handler
      imagesComboBox = new JComboBox(names);
      imagesComboBox.setMaximumRowCount(3);

      imagesComboBox.addItemListener(
         new ItemListener() { // anonymous inner class
            // handle JComboBox event
            public void itemStateChanged( ItemEvent event ) {
               // determine whether check box selected
               if ( event.getStateChange() == ItemEvent.SELECTED )
                  label.setIcon( icons[imagesComboBox.getSelectedIndex()]);
            }
        } // end anonymous inner class
      ); // end call to addItemListener

      container.add( imagesComboBox );
   }
} //// End ComboBoxTest class
```

Ví dụ JComboBox – Kết quả







JList

- Danh sách các mục
- Người sử dụng có thể chọn một hoặc nhiều mục
- Single-selection vs. multiple-selection

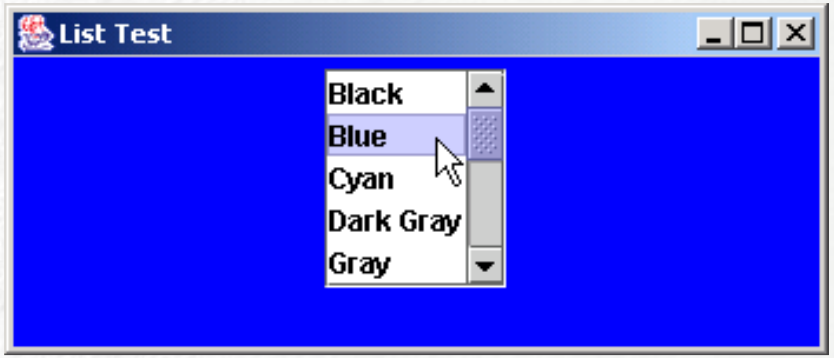
Ví dụ sử dụng JList

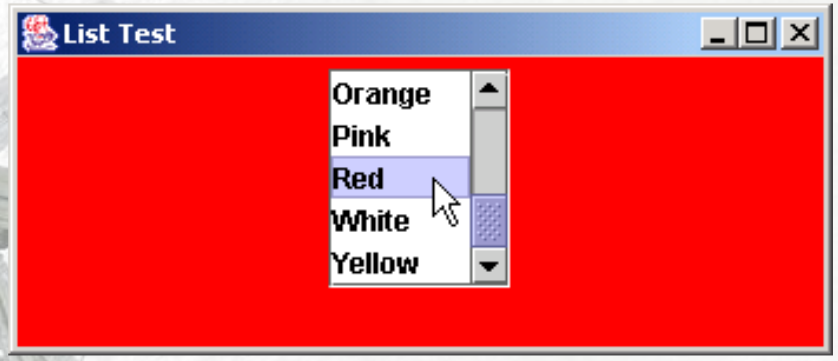
```
// Selecting colors from a JList.
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;

public class ListTest extends JFrame {
   private JList colorList;
   private final String colorNames[] = { "Red", "Blue", "Green"};
   private final Color colors[] = { Color.RED, Color.BLUE, Color.GREEN};
   // set up GUI
  public ListTest() {
     super( "List Test" );
     // get content pane and set its layout
     Container container = getContentPane();
     container.setLayout( new FlowLayout());
     // create a list with items in colorNames array
     colorList = new JList( colorNames );
     colorList.setVisibleRowCount( 5 );
```

Ví dụ sử dụng JList

```
        // do not allow multiple selections
        colorList.setSelectionMode(
            ListSelectionModel.SINGLE_SELECTION );
        // add a JScrollPane containing JList to content pane
        container.add( new JScrollPane(colorList));
        colorList.addListSelectionListener(
           new ListSelectionListener() { // anonymous inner class
              // handle list selection events
              public void valueChanged( ListSelectionEvent event ) {
                 container.setBackground(
                    colors[colorList.getSelectedIndex()]);
              }
           } // end anonymous inner class
        ); // end call to addListSelectionListener
     }
} //// End ListTest class
```

Ví dụ Jlist – Kết quả





Xử lý sự kiện chuột

- Các giao tiếp lắng nghe cho các sự kiện chuột
  - MouseListener
  - MouseMotionListener
- Lắng nghe cho đối tượng sự kiện MouseEvent.

Các phương thức của MouseListener

- public void mousePressed(MouseEvent event): Được gọi khi một nút chuột được nhấn trên một thành phần.
- public void mouseClicked(MouseEvent event): Được gọi khi một nút chuột được nhấn và thả ra trên một thành phần.
- public void mouseReleased(MouseEvent event): Được gọi khi một nút chuột được thả ra sau khi được nhấn. Trước sự kiện này luôn luôn là một sự kiện mousePressed.
- public void mouseEntered(MouseEvent event): Được gọi khi con trỏ chuột vào những ranh giới của một thành phần.
- public void mouseExited(MouseEvent event): Được gọi khi con trỏ chuột rời ranh giới của một thành phần

MouseMotionListener

- public void mouseDragged(MouseEvent event)
  - Được gọi khi nút chuột được nhấn và di chuyển.
  - Trước sự kiện này luôn luôn là gọi tới sự kiện mousePressed.
  - Tất cả các sự kiện kéo đều được gửi tới thành phần mà trên đó sự kéo bắt đầu.
- public void mouseMoved(MouseEvent event):
  - Được gọi khi con chuột được di chuyển trên một thành phần.
  - Tất cả các sự kiện chuyển động đều được gửi tới thành phần mà vị trí con chuột hiện thời ở đó.

Ví dụ sử dụng sự kiện chuột

```
// Demonstrating mouse events.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MouseTracker extends JFrame
                implements MouseListener, MouseMotionListener {
   private JLabel statusBar;
   // set up GUI and register mouse event handlers
   public MouseTracker() {
      super( "Demonstrating Mouse Events" );
      statusBar = new JLabel();
      getContentPane().add( statusBar, BorderLayout.SOUTH );
      addMouseListener( this ); // listens for own mouse and
      addMouseMotionListener( this ); // mouse-motion events
      //
      setSize( 275, 100 );
      setVisible( true );
   }
```

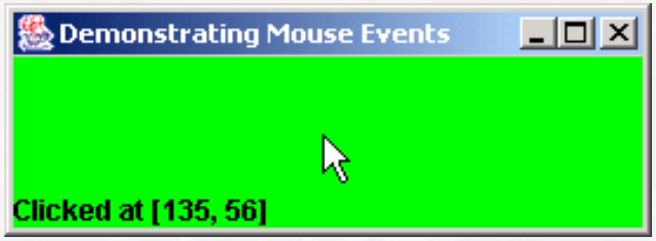
Ví dụ sử dụng sự kiện chuột

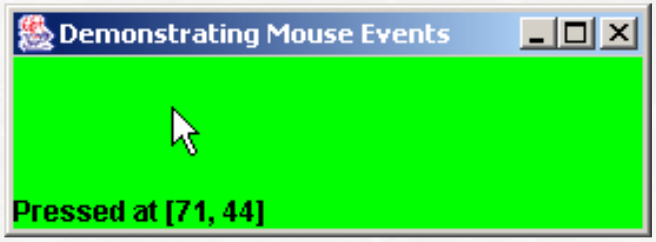
```
// handle event when mouse released immediately after press
public void mouseClicked( MouseEvent event ) {
   statusBar.setText( "Clicked at [" + event.getX() +
      ", " + event.getY() + "]" );
}
// handle event when mouse pressed
public void mousePressed( MouseEvent event ) {
   statusBar.setText( "Pressed at [" + event.getX() +
      ", " + event.getY() + "]" );
}
// handle event when mouse released after dragging
public void mouseReleased( MouseEvent event ) {
   statusBar.setText( "Released at [" + event.getX() +
      ", " + event.getY() + "]" );
}
// handle event when mouse enters area
public void mouseEntered( MouseEvent event ) {
   statusBar.setText( "Mouse entered at [" + event.getX() +
      ", " + event.getY() + "]" );
   getContentPane().setBackground( Color.GREEN );
}
```

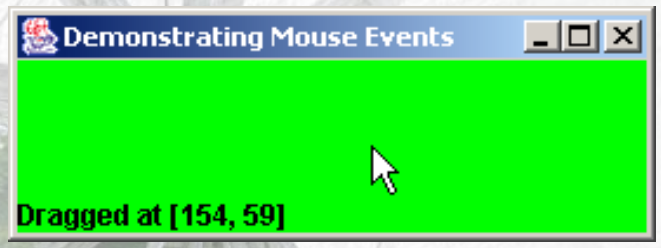
Ví dụ sử dụng sự kiện chuột

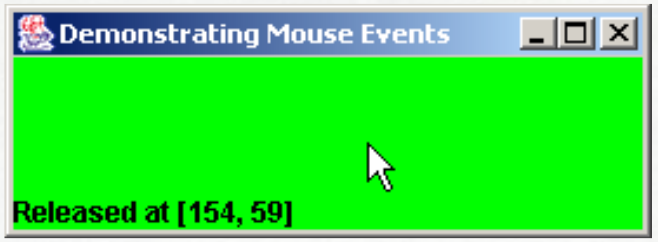
```
   // handle event when mouse exits area
   public void mouseExited( MouseEvent event ) {
      statusBar.setText( "Mouse outside window" );
      getContentPane().setBackground( Color.WHITE );
   }
   // MouseMotionListener event handlers
   // handle event when user drags mouse with button pressed
   public void mouseDragged( MouseEvent event ) {
      statusBar.setText( "Dragged at [" + event.getX() +
         ", " + event.getY() + "]" );
   }
   // handle event when user moves mouse
   public void mouseMoved( MouseEvent event ) {
      statusBar.setText( "Moved at [" + event.getX() +
         ", " + event.getY() + "]" );
   }
} // end class MouseTracker
```

Ví dụ sự kiện chuột - Kết quả



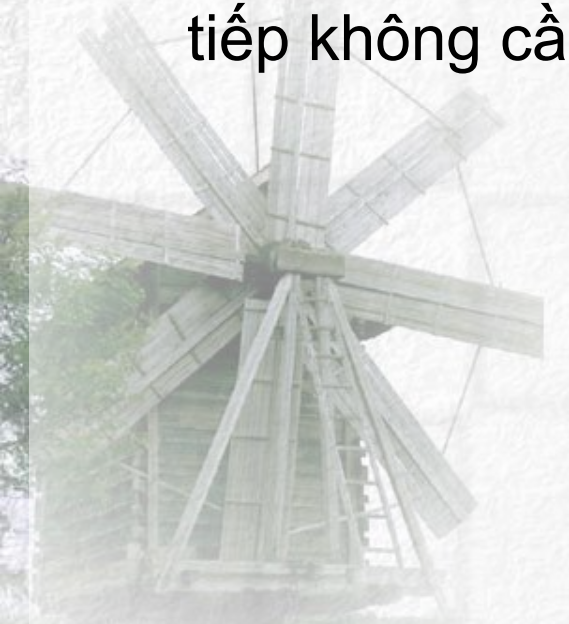






Các lớp Adapter

- Hiện thực giao tiếp
- Cung cấp sự cài đặt mặc định của mỗi phương thức giao tiếp
- Được sử dụng khi tất cả các phương thức trong giao tiếp không cần thiết



Các lớp Adapter

- Các lớp Event-Adapter và giao tiếp chúng hiện thực

Event-adapter class	Implements interface
ComponentAdapter	ComponentListener
ContainerAdapter	ContainerListener
FocusAdapter	FocusListener
KeyAdapter	KeyListener
MouseAdapter	MouseListener
MouseMotionAdapter	MouseMotionListener
WindowAdapter	WindowListener

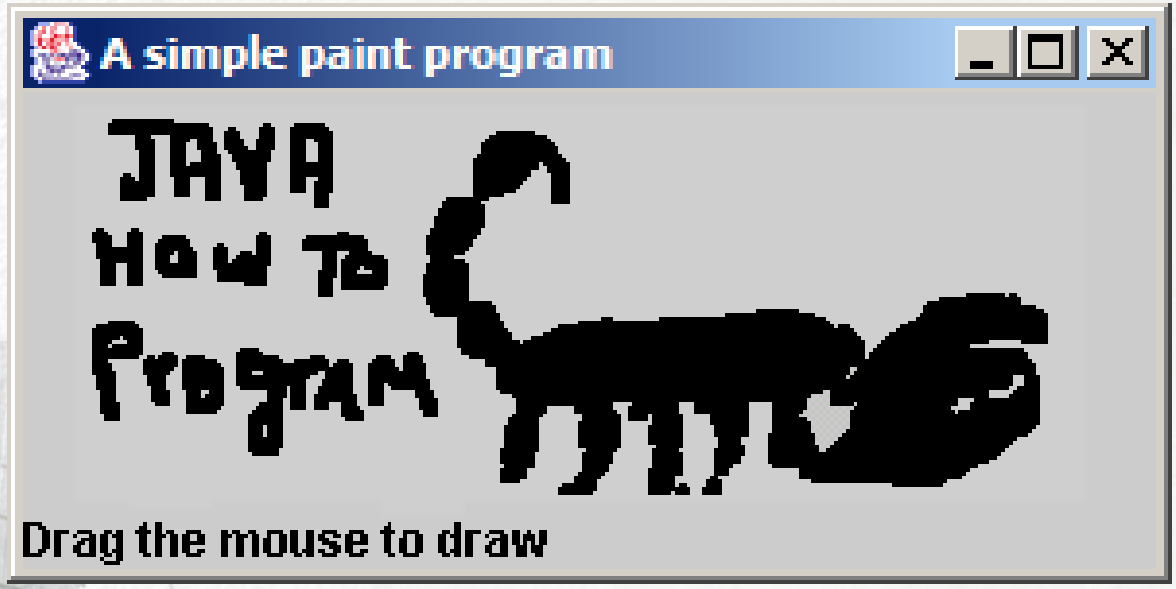
Ví dụ sử dụng lớp Adapter

```
// Using class MouseMotionAdapter.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Painter extends JFrame {
   //
   private Point point = new Point();
   // set up GUI and register mouse event handler
   public Painter() {
      super( "A simple paint program" );
      // create a label and place it in SOUTH of BorderLayout
      getContentPane().add( new JLabel( "Drag the mouse to draw" ),
         BorderLayout.SOUTH );
```

Ví dụ sử dụng lớp Adapter

```
      // register listener  - by using anonymous inner class
      addMouseMotionListener( new MouseMotionAdapter() {
         // store drag coordinates and repaint
         public void mouseDragged( MouseEvent event ) {
            point = event.getPoint();
            repaint();
         }
      }); // end call to addMouseMotionListener
   } // end Painter constructor
   // draw oval in a 4-by-4 bounding box at specified location
   public void paint( Graphics g ) {
      //super.paint( g ); // clears drawing area
     g.fillOval(point.x, point.y, 4, 4);
   }
   public static void main( String args[] )
   {
      Painter painter = new Painter();
     painter.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
     painter.show();
   }
} // end class Painter
```

Ví dụ sử dụng lớp Adapter



Xử lý sự kiện phím

- Giao tiếp KeyListener
- Xử lý những sự kiện phím
  - Sinh ra khi những phím trên bàn phím được nhấn và thả
- Lớp KeyEvent chứa mã phím ảo mà đại diện cho phím

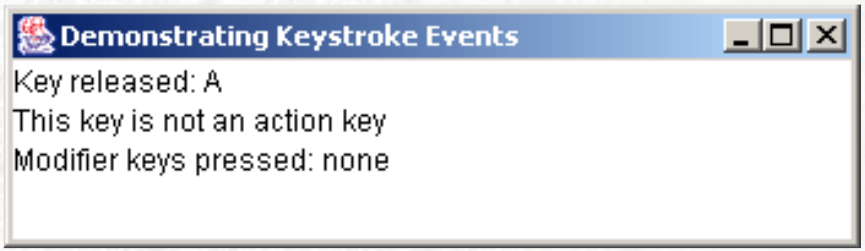
Ví dụ sử dụng sự kiện bàn phím

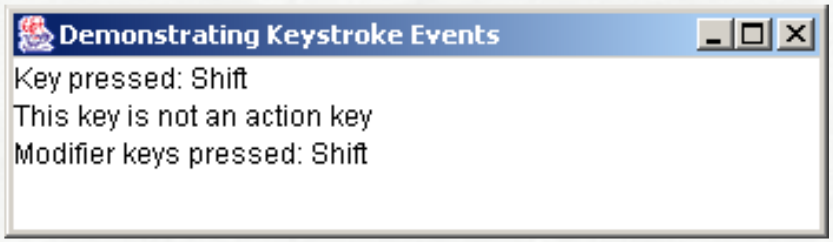
```
// Demonstrating keystroke events.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class KeyDemo extends JFrame implements KeyListener {
   private String line1 = "", line2 = "", line3 = "";
   private JTextArea textArea;
   // set up GUI
   public KeyDemo() {
      super( "Demonstrating Keystroke Events" );
      // set up JTextArea
      textArea = new JTextArea( 10, 15 );
      textArea.setText( "Press any key on the keyboard..." );
      textArea.setEnabled( false );
      textArea.setDisabledTextColor( Color.BLACK );
      getContentPane().add( textArea );
      addKeyListener( this ); // allow frame to process Key events
      setSize( 350, 100 );
      setVisible( true );
   }
```

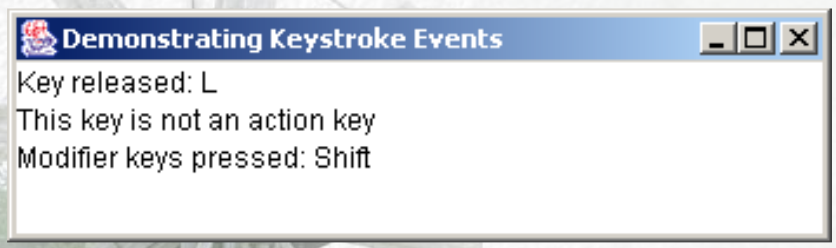
Ví dụ sử dụng sự kiện bàn phím

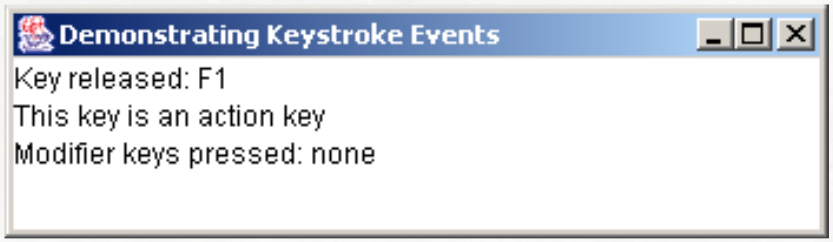
```
   // handle press of any key
   public void keyPressed( KeyEvent event ) {
      line1 = "Key pressed: " + event.getKeyText( event.getKeyCode() );
      setLines2and3( event );
   }
   // handle release of any key
   public void keyReleased( KeyEvent event ) {
      line1 = "Key released: " + event.getKeyText( event.getKeyCode() );
      setLines2and3( event );
   }
   // handle press of an action key
   public void keyTyped( KeyEvent event ) {
      line1 = "Key typed: " + event.getKeyChar();
      setLines2and3( event );
   }
   // set second and third lines of output
   private void setLines2and3( KeyEvent event ) {
      line2 = "This key is " +(event.isActionKey()? "" : "not ") + "an action key";
      String temp = event.getKeyModifiersText( event.getModifiers() );
      line3 = "Modifier keys pressed: " + (temp.equals("") ? "none" : temp );
      textArea.setText( line1 + "\n" + line2 + "\n" + line3 + "\n" );
   }
} // end class KeyDemo
```

Ví dụ sử dụng sự kiện bàn phím









Quản lý bố cục

- Cung cấp để sắp xếp các thành phần GUI
- Cung cấp những khả năng cách trình bày cơ bản
- Xử lý các chi tiết bố cục
- Lập trình viên có thể tập trung vào “vẻ ngoài” cơ bản
- Giao tiếp LayoutManager

Quản lý bố cục

- FlowLayout: Mặc định cho Java.awt.Applet, Java.awt.Panel và javax.swing.JPanel. Đặt các thành phần theo tuần tự (trái qua phải) theo thứ tự khi chúng được thêm. Cũng có thể chỉ rõ thứ tự của các thành phần bởi việc sử dụng phương thức add() Container, với các đối số là một thành phần và một số nguyên chỉ số.
- BorderLayout: Mặc định cho khung nội dung của JFrames (và các Window khác) và JApplets. Sắp xếp các thành phần vào trong 5 vùng: Bắc (NORTH), Nam (SOUTH), Đông (EAST), Tây (WEST) và Trung tâm (CENTER)
- GridLayout: Sắp xếp các thành phần vào trong các hàng và các cột.
- …

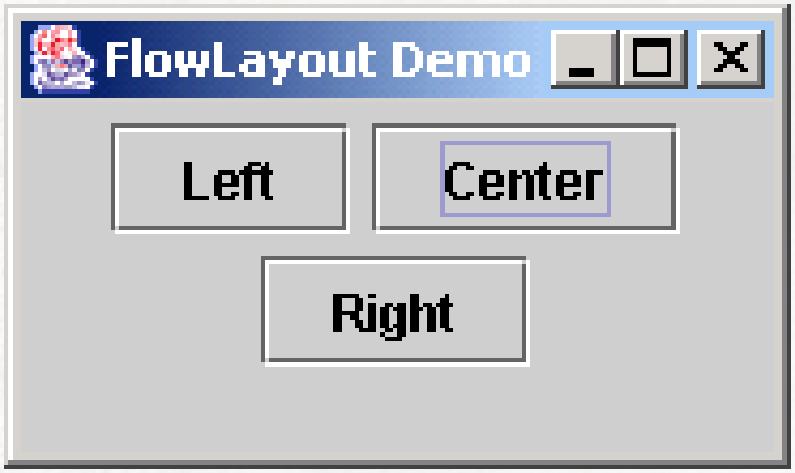
FlowLayout

- Bộ quản lý bố cục cơ bản nhất
- Các thành phần GUI được bố trí trong bộ chứa từ trái qua phải.

Ví dụ sử dụng FlowLayout

```
// Demonstrating FlowLayout alignments.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class FlowLayoutDemo extends JFrame {
   private JButton leftButton, centerButton, rightButton;
   private Container container;
   private FlowLayout layout;
   // set up GUI and register button listeners
   public FlowLayoutDemo() {
      super( "FlowLayout Demo" );
      layout = new FlowLayout();
      // get content pane and set its layout
      container = getContentPane();
      container.setLayout( layout );
      // set up buttons
      ...
   }
}
```

Ví dụ FlowLayout – Kết quả



BorderLayout

- Sắp xếp các thành phần vào năm vùng
  - NORTH (đỉnh container)
  - SOUTH (đáy container)
  - EAST (bên trái container)
  - WEST (bên phải container)
  - CENTER (ở giữa container)

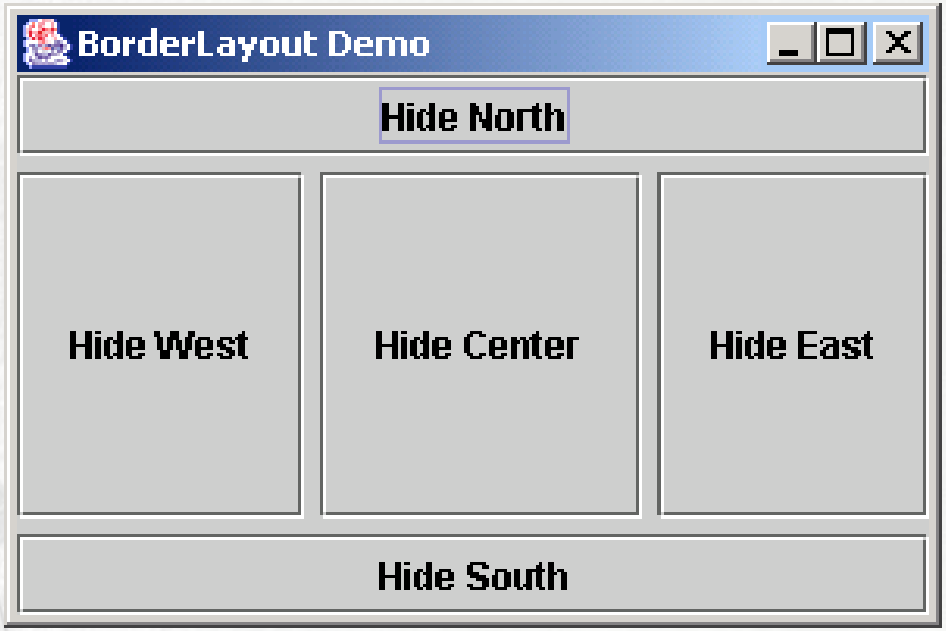
Ví dụ sử dụng BorderLayout

```
// Demonstrating BorderLayout.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class BorderLayoutDemo extends JFrame {
   private JButton buttons[];
   private final String names[] = { "Hide North", "Hide South",
      "Hide East", "Hide West", "Hide Center" };
   private BorderLayout layout;
   // set up GUI
   public BorderLayoutDemo() {
      super( "BorderLayout Demo" );
      layout = new BorderLayout( 5, 5 ); // 5 pixel gaps
      // get content pane and set its layout
      Container container = getContentPane();
      container.setLayout( layout );
      // instantiate button objects
      buttons = new JButton[ names.length ];
      for ( int count = 0; count < names.length; count++ ) {
         buttons[ count ] = new JButton( names[ count ] );
         buttons[ count ].addActionListener( this );
      }
```

Ví dụ sử dụng BorderLayout

```
        // place buttons in BorderLayout; order not important
        container.add( buttons[ 0 ], BorderLayout.NORTH );
        container.add( buttons[ 1 ], BorderLayout.SOUTH );
        container.add( buttons[ 2 ], BorderLayout.EAST );
        container.add( buttons[ 3 ], BorderLayout.WEST );
        container.add( buttons[ 4 ], BorderLayout.CENTER );
        //
        setSize( 300, 200 );
        setVisible( true );
    } // end constructor BorderLayoutDemo

    public static void main( String args[] ) {
        BorderLayoutDemo application = new BorderLayoutDemo();
        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    }

} // end class BorderLayoutDemo
```

Ví dụ sử dụng BorderLayout



GridLayout

- Chia Container thành một lưới gồm các hàng và các cột xác định
- Các thành phần được bổ sung bắt đầu tại ô trên-trái
  - Tiến hành trái-qua-phải cho đến khi hàng đầy

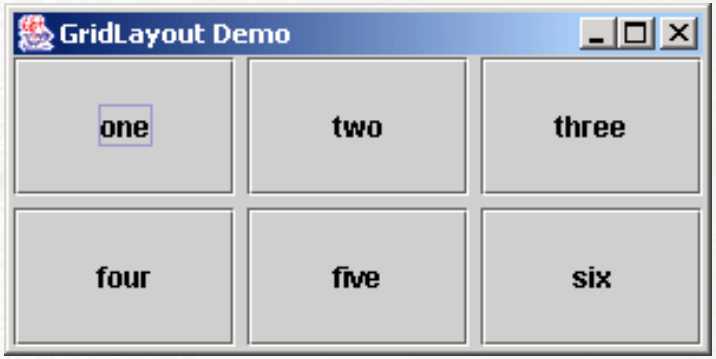
Ví dụ sử dụng GridLayout

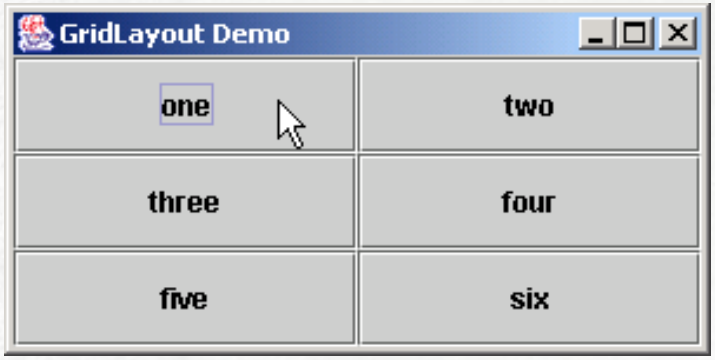
```
// Demonstrating GridLayout.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class GridLayoutDemo extends JFrame implements ActionListener {
   private JButton buttons[];
   private final String names[] =
      { "one", "two", "three", "four", "five", "six" };
   private boolean toggle = true;
   private Container container;
   private GridLayout grid1, grid2;
   // set up GUI
   public GridLayoutDemo() {
      super( "GridLayout Demo" );
      // set up layouts
      grid1 = new GridLayout( 2, 3, 5, 5 );
      grid2 = new GridLayout( 3, 2 );
      // get content pane and set its layout
      container = getContentPane();
      container.setLayout( grid1 );
```

Ví dụ sử dụng GridLayout

```
      // create and add buttons
      buttons = new JButton[ names.length ];
      for ( int count = 0; count < names.length; count++ ) {
         buttons[ count ] = new JButton( names[ count ] );
         buttons[ count ].addActionListener( this );
         container.add( buttons[ count ] );
      }
      setSize( 300, 150 );  setVisible( true );
   } // end constructor GridLayoutDemo

   // handle button events by toggling between layouts
   public void actionPerformed( ActionEvent event ) {
      if ( toggle )
         container.setLayout( grid2 );
      else
         container.setLayout( grid1 );
      toggle = !toggle; // set toggle to opposite value
      container.validate();
   }
   public static void main( String args[] ) {
      GridLayoutDemo application = new GridLayoutDemo();
      application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }
} // end class GridLayoutDemo
```

Ví dụ sử dụng GridLayout





Panel – Khung chứa

- Giúp tổ chức các thành phần
- Lớp JPanel là lớp xuất của JComponent
- Có thể có nhiều thành phần (và các khung chứa panel khác) được thêm vào chúng.

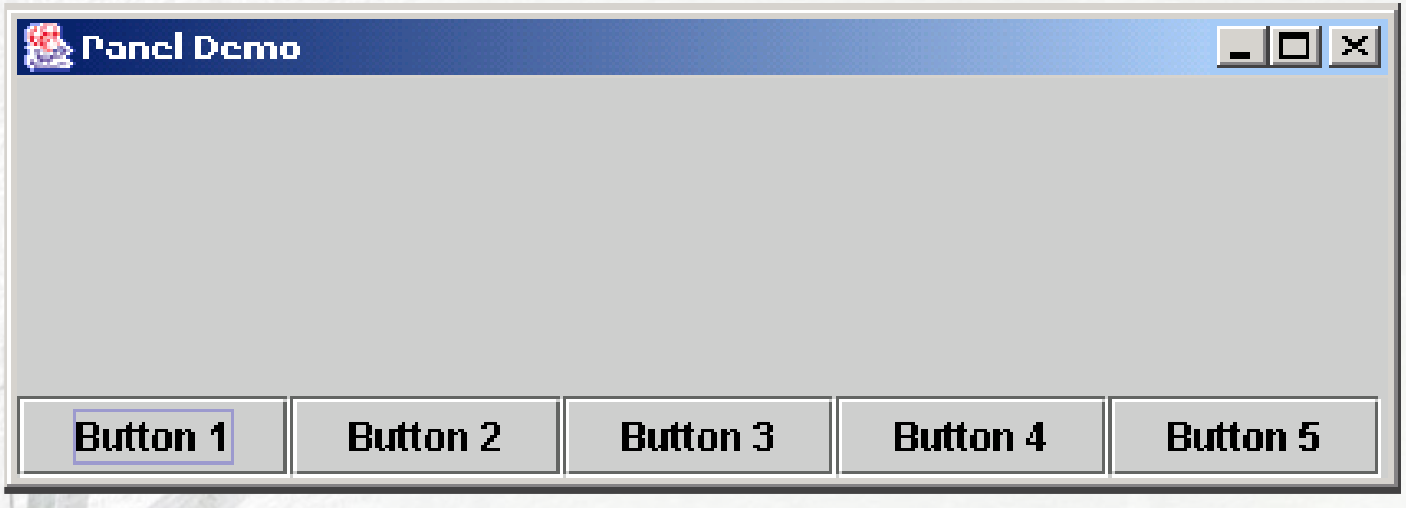
Ví dụ sử dụng JPanel

```
// Using a JPanel to help lay out components.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PanelDemo extends JFrame {
   private JPanel buttonPanel;
   private JButton buttons[];
   // set up GUI
   public PanelDemo() {
      super( "Panel Demo" );
      // get content pane
      Container container = getContentPane();
      // create buttons array
      buttons = new JButton[ 5 ];
      // set up panel and set its layout
      buttonPanel = new JPanel();
      buttonPanel.setLayout( new GridLayout( 1, buttons.length ) );
```

Ví dụ sử dụng JPanel

```
      // create and add buttons
      for ( int count = 0; count < buttons.length; count++ ) {
         buttons[ count ] = new JButton( "Button " + ( count + 1 ) );
         buttonPanel.add( buttons[ count ] );
      }
      container.add( buttonPanel, BorderLayout.SOUTH );
      setSize( 425, 150 );
      setVisible( true );
   } // end constructor PanelDemo

   public static void main( String args[] ) {
      PanelDemo application = new PanelDemo();
      application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }
} // end class PanelDemo
```

Ví dụ sử dụng JPanel



JTabbedPane

- Sắp xếp các thành phần trong các lớp
  - Một lớp xuất hiện tại một thời điểm
  - Truy cập mỗi lớp thông qua Tab
- Lớp JTabbedPane

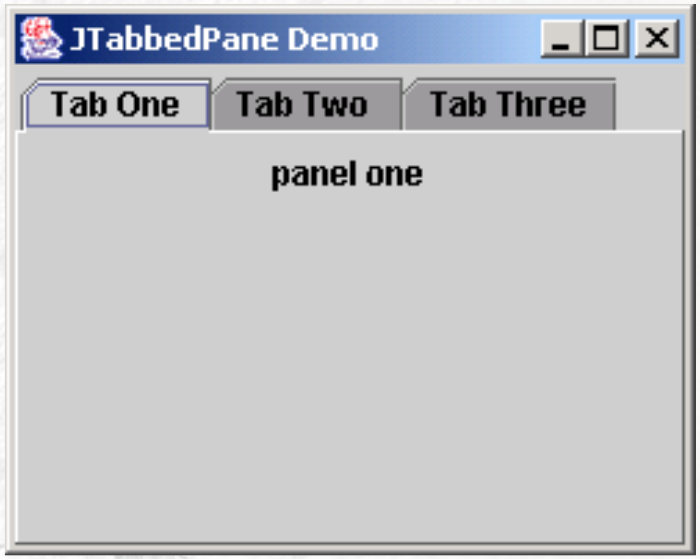
Ví dụ sử dụng JTabbedPane

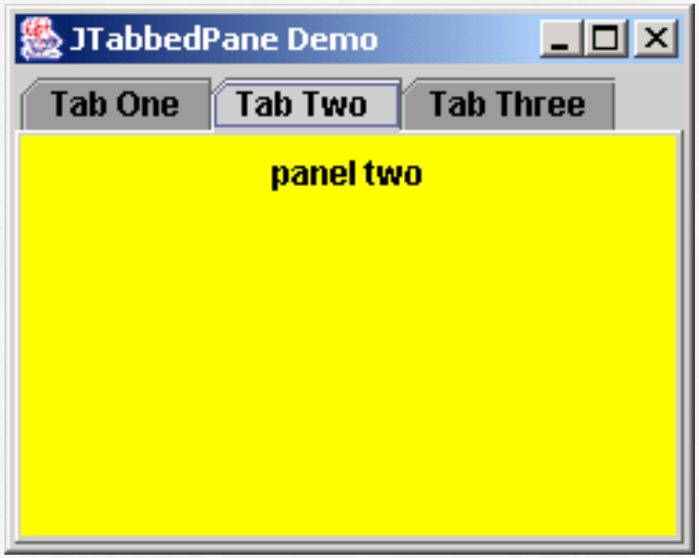
```
// Demonstrating JTabbedPane.
import java.awt.*;
import javax.swing.*;

public class JTabbedPaneDemo extends JFrame {
   // set up GUI
   public JTabbedPaneDemo() {
      super( "JTabbedPane Demo " );
      // create JTabbedPane
      JTabbedPane tabbedPane = new JTabbedPane();
      // set up pane11 and add it to JTabbedPane
      JLabel label1 = new JLabel( "panel one", SwingConstants.CENTER );
      JPanel panel1 = new JPanel();
      panel1.add( label1 );
      tabbedPane.addTab( "Tab One", null, panel1, "First Panel" );
      // set up panel2 and add it to JTabbedPane
      JLabel label2 = new JLabel( "panel two", SwingConstants.CENTER );
      JPanel panel2 = new JPanel();
      panel2.setBackground( Color.YELLOW );
      panel2.add( label2 );
      tabbedPane.addTab( "Tab Two", null, panel2, "Second Panel" );
```

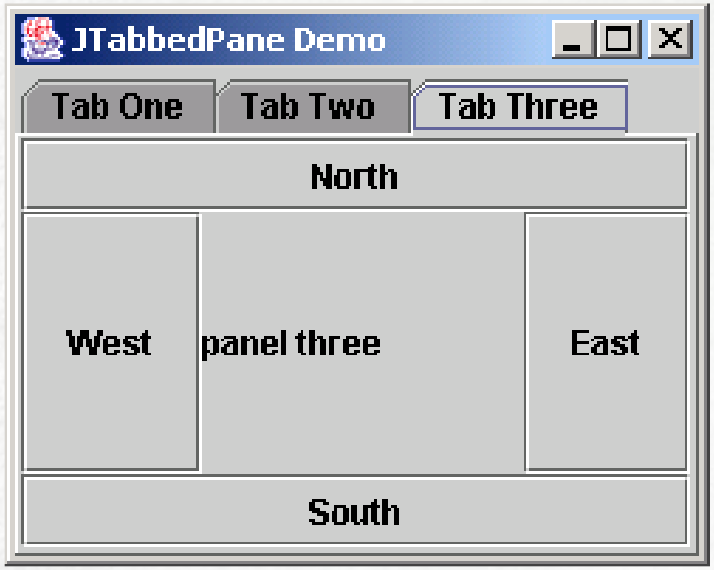
Ví dụ sử dụng JTabbedPane

```
        // set up panel3 and add it to JTabbedPane
        JLabel label3 = new JLabel( "panel three" );
        JPanel panel3 = new JPanel();
        panel3.setLayout( new BorderLayout() );
        panel3.add( new JButton( "North" ), BorderLayout.NORTH );
        panel3.add( new JButton( "West" ), BorderLayout.WEST );
        panel3.add( new JButton( "East" ), BorderLayout.EAST );
        panel3.add( new JButton( "South" ), BorderLayout.SOUTH );
        panel3.add( label3, BorderLayout.CENTER );
        tabbedPane.addTab( "Tab Three", null, panel3, "Third Panel" );
        // add JTabbedPane to container
        getContentPane().add( tabbedPane );
    } // end constructor

} // end JTabbedPaneDemo class
```

Ví dụ sử dụng JTabbedPane

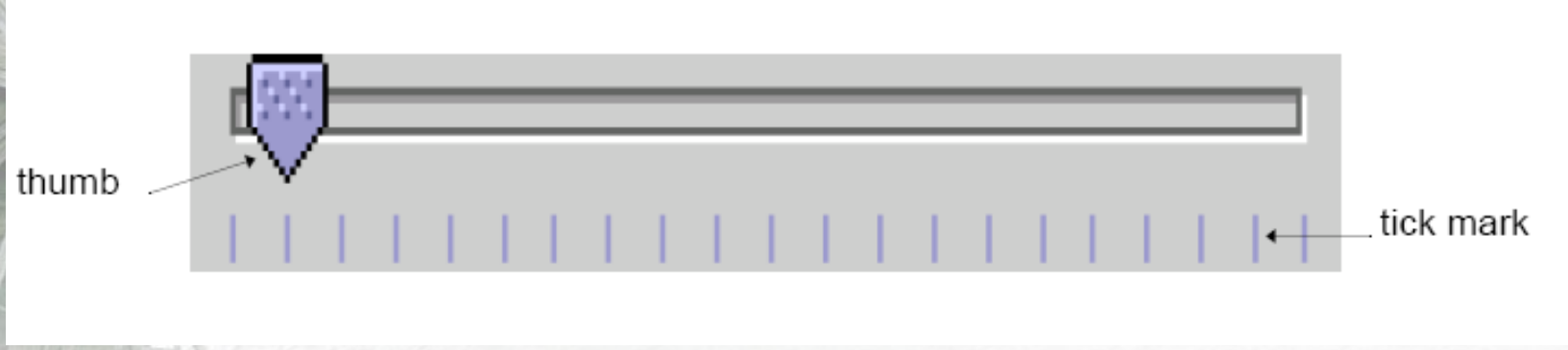






Con chạy - JSlider

- Cho phép người sử dụng chọn giá trị nguyên trong một vùng giá trị xác định.
- Một số đặc tính:
  - Tick marks (major and minor)
  - Snap-to ticks
  - Hướng (ngang hoặc đứng)



Ví dụ: OvalPanel.java

```
// A customized JPanel class.
import java.awt.*;
import javax.swing.*;

public class OvalPanel extends JPanel {
   private int diameter = 10;

   // draw an oval of the specified diameter
   public void paintComponent( Graphics g ) {
      super.paintComponent( g );
      g.fillOval( 10, 10, diameter, diameter );
   }
```

Ví dụ: OvalPanel.java

```
   // validate and set diameter, then repaint
   public void setDiameter( int newDiameter ) {
      // if diameter invalid, default to 10
      diameter = ( newDiameter >= 0 ? newDiameter : 10 );
      repaint();
   }
   // used by layout manager to determine preferred size
   public Dimension getPreferredSize() {
      return new Dimension( 200, 200 );
   }

   // used by layout manager to determine minimum size
   public Dimension getMinimumSize() {
      return getPreferredSize();
   }
} // end class OvalPanel
```

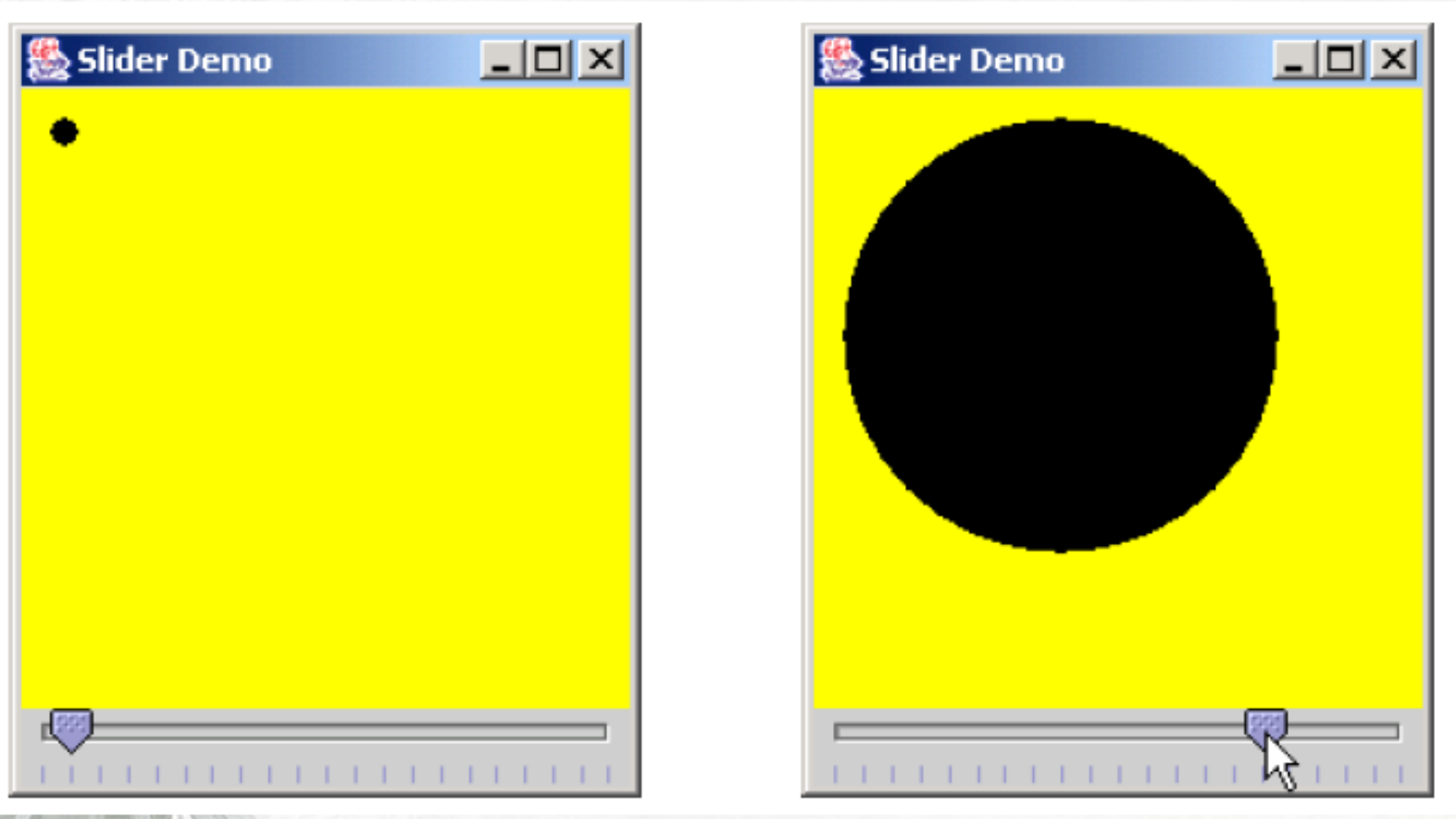
Ví dụ: SliderDemo.java

```
// Using JSliders to size an oval.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class SliderDemo extends JFrame {
   private JSlider diameterSlider;
   private OvalPanel myPanel;
   // set up GUI
   public SliderDemo()  {
      super( "Slider Demo" );
      // set up OvalPanel
      myPanel = new OvalPanel();
      myPanel.setBackground( Color.YELLOW );
      // set up JSlider to control diameter value
      diameterSlider =
         new JSlider( SwingConstants.HORIZONTAL, 0, 200, 10 );
      diameterSlider.setMajorTickSpacing( 10 );
      diameterSlider.setPaintTicks( true );
```

Ví dụ: SliderDemo.java

```
        // register JSlider event listener
        diameterSlider.addChangeListener(
            new ChangeListener() { // anonymous inner class
            // handle change in slider value
                public void stateChanged( ChangeEvent e ) {
                    myPanel.setDiameter(diameterSlider.getValue());
                }
            } // end anonymous inner class
        ); // end call to addChangeListener
        // attach components to content pane
        Container container = getContentPane();
        container.add( diameterSlider, BorderLayout.SOUTH );
        container.add( myPanel, BorderLayout.CENTER );

        setSize( 220, 270 );
        setVisible( true );
    } // end constructor SliderDemo
```

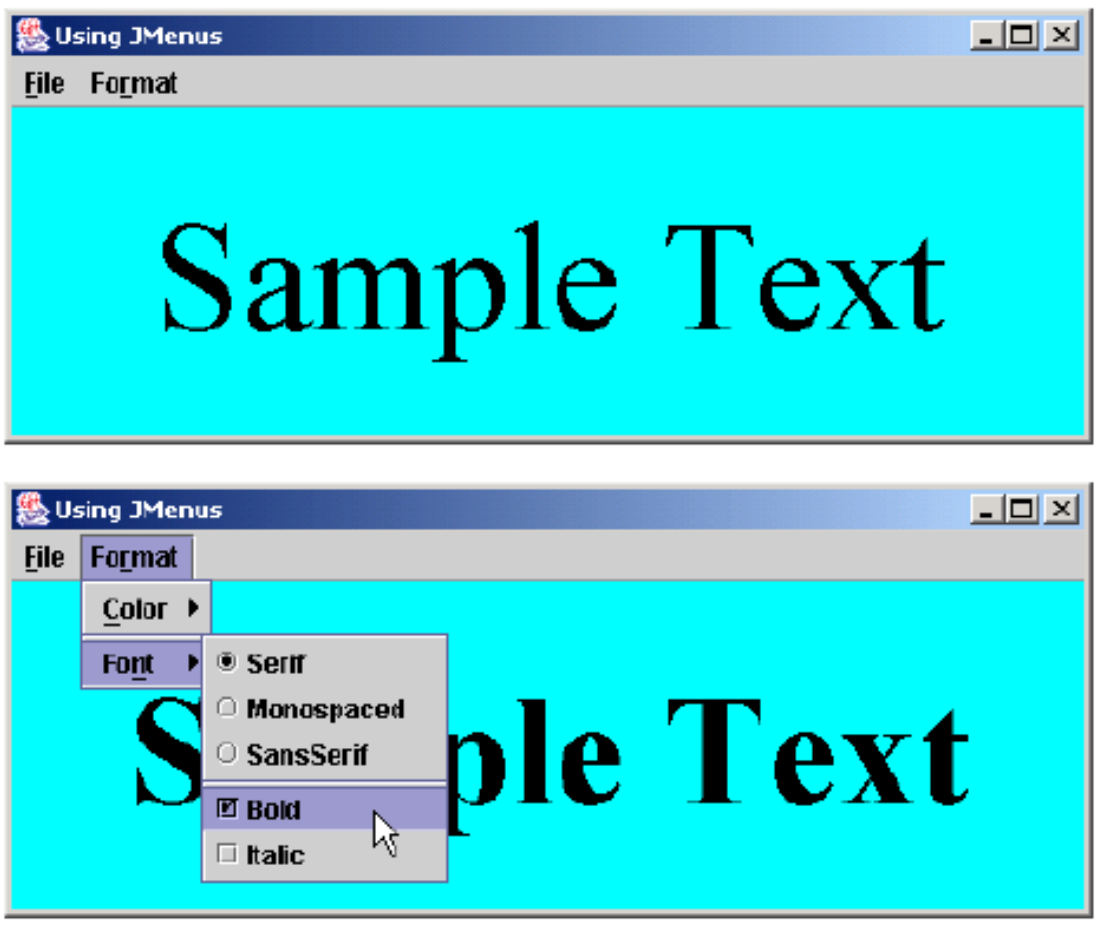
Ví dụ: SliderDemo



Sử dụng Menu với JFrame

- Cho phép thực hiện các hành động với GUI
- Chứa bởi thanh menu (menu bar)
  - JMenuBar
- Bao gồm các mục menu (menu items)
  - JMenuItem

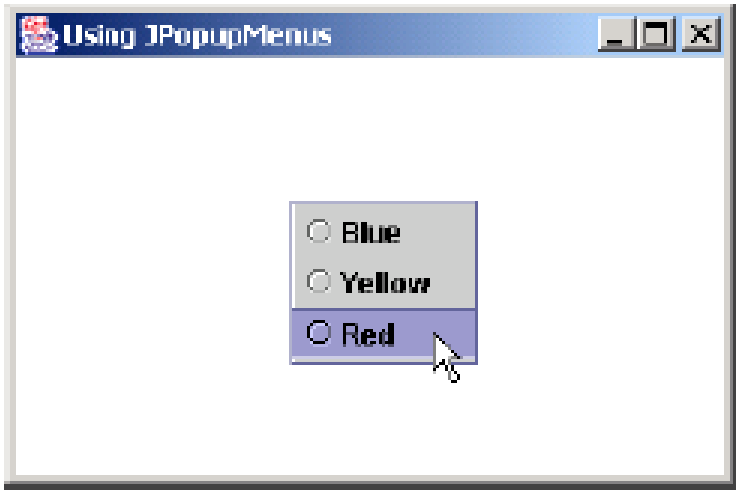
Ví dụ: MenuDemo.java

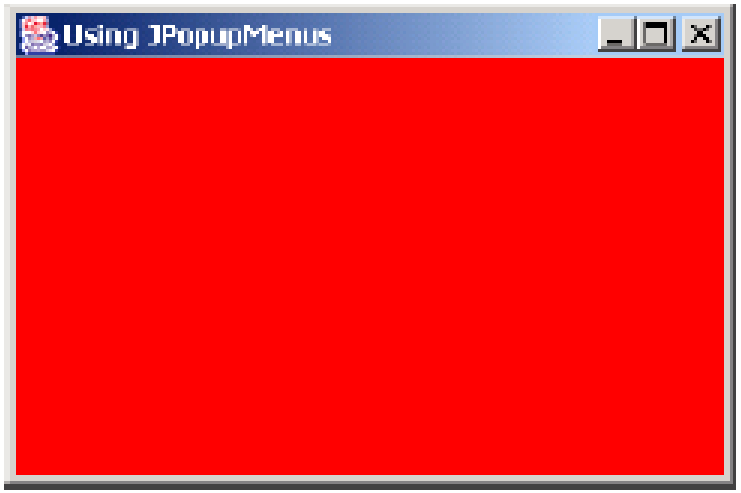


JPopupMenu

- Context-sensitive popup menus
  - JPopupMenu
  - Menu được phát sinh phụ thuộc vào thành phần đang truy cập.
- Cách tạo:
  - JPopupMenu popup = new JPopupMenu();
  - JMenuItem items = new JMenuItem(“Red”);
  - popup.add(items);
  - …
  - // handling event – mousePressed
  - popup.show(ev.getComponent(), ev.getX(), ev.getY());

Ví dụ: PopupDemo.java





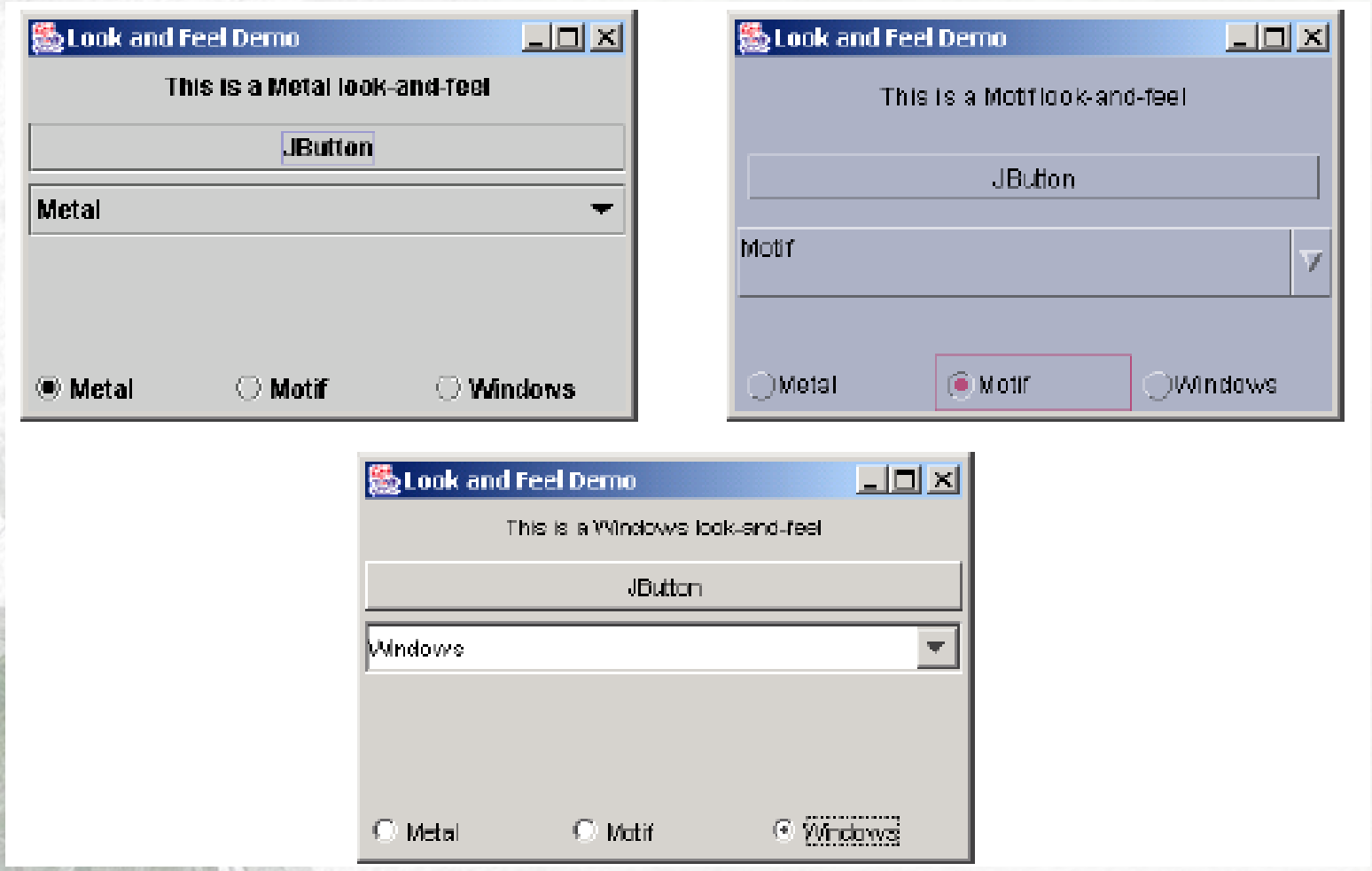
Look-and-Feel (cảm quan)

- Thay đổi cảm quan (dáng vẻ của giao diện)
  - Ví dụ, Microsoft Windows look-and-feel đến Motif look-and-feel.
- Linh động

Cách sử dụng

- private UIManager.LookAndFeelInfo looks[];
- …
- // get installed look-and-feel information
- looks = UIManager.getInstalledLookAndFeels();
- …
- // change look and feel
- UIManager.setLookAndFeel(looks[index].getClassName());
- SwingUtilities.updateComponentTreeUI(this);

Ví dụ: LookFeelDemo.java



HẾT

BÀI 11