



PHẦN 2: JAVA NÂNG CAO

Nội dung:

Chương 5: Lập trình giao diện với Swing

- 5.1 Giới thiệu về Swing
- 5.2 Các control giao diện cơ bản
- 5.3 Nhóm lớp chứa (Container)
- 5.4 Mô hình quản lý cách trình bày - Layout manager
- 5.5 Xử lý sự kiện

Chương 6: Lập trình CSDL trong Java

- 6.1 Giới thiệu về JDBC (Java Database Connectivity)
- 6.2 Kết nối dữ liệu với Oracle, MySQL, SQL Server...
- 6.3 Các thao tác dữ liệu căn bản

Chương 7: Project quản lý sinh viên

- 7.1 Tạo project qlsv-swing
- 7.2. Tạo chức năng login
 - 1. Tạo lớp User.java
 - 2. Tạo lớp UserDao.java
 - 3. Tạo lớp LoginView.java
 - 4. Tạo lớp LoginController.java
- 7.3. Tạo chức năng quản lý sinh viên
 - 1. Tạo lớp Student.java
 - 2. Tạo lớp StudentXML.java
 - 3. Tạo lớp StudentDao.java
 - 4. Tạo lớp StudentView.java
 - 5. Tạo lớp StudentController.java
 - 6. Tạo lớp FileUtils.java
- 7.4. Tạo lớp App.java
 - 1. Run bài tập quản lý sinh viên trong java swing
 - 2. Login
 - 3. Thêm sinh viên

Chương 5: Lập trình giao diện với Swing

5.1 Giới thiệu Java Swing

Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng window-based. Nó được xây dựng trên API AWT (Abstract Windowing Toolkit) và được viết hoàn toàn bằng Java.

Không giống như AWT, Java Swing cung cấp các thành phần không phụ thuộc vào nền tảng và nhẹ hơn.

Gói javax.swing cung cấp các lớp cho java swing API như JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser, v.v.

Nội dung chính:

- Sự khác nhau giữa AWT và Swing
- JFC là gì?
- Phân cấp các lớp Java Swing
- Các phương thức thường dùng của lớp Component
- Ví dụ về Java Swing
 - Ví dụ Swing Java đơn giản
 - Ví dụ Java Swing - tạo đối tượng của lớp JFrame
 - Ví dụ Java Swing - kế thừa lớp JFrame

5.1.1 Sự khác nhau giữa AWT và Swing

Có rất nhiều sự khác biệt giữa java awt và swing được đưa ra dưới đây.

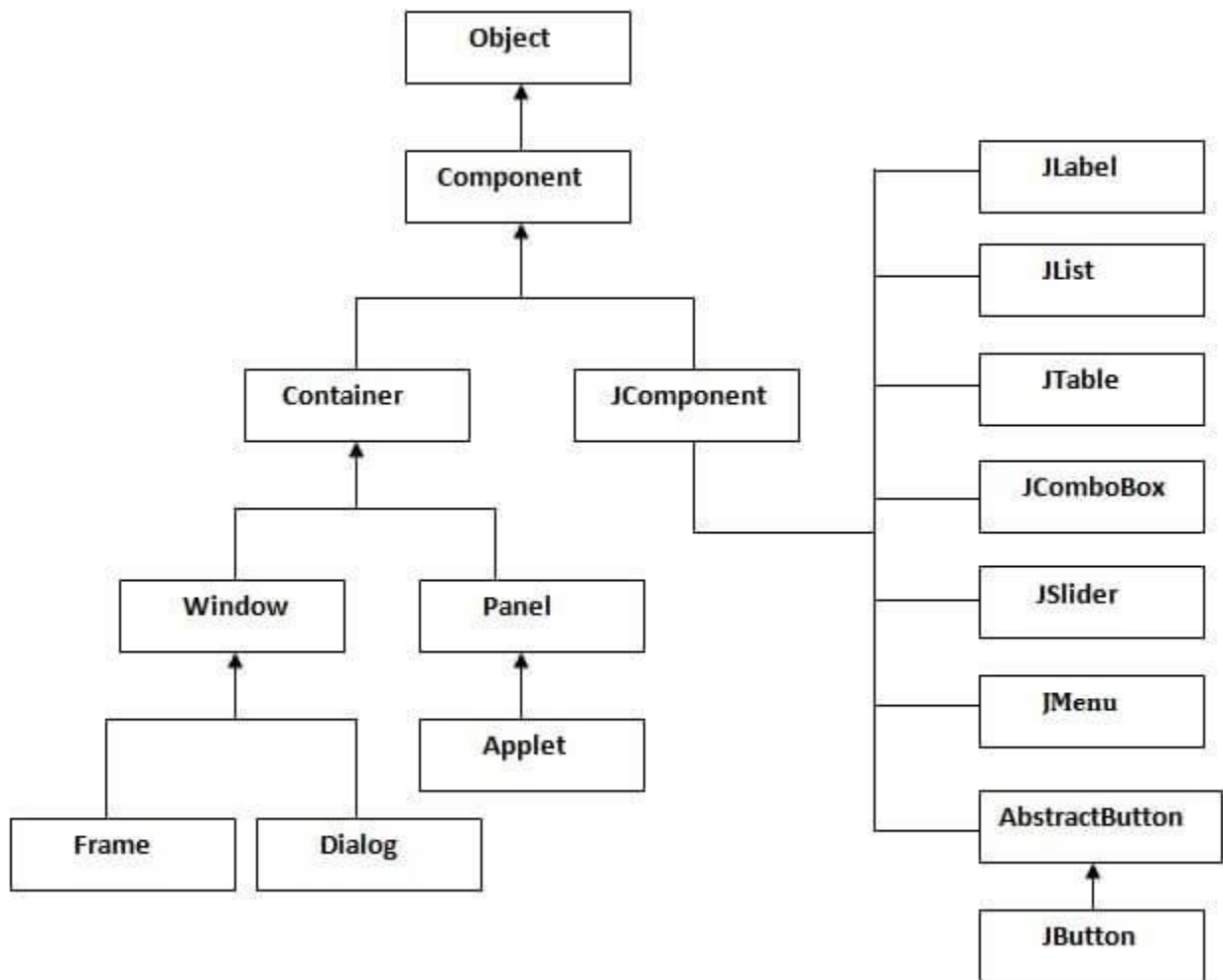
STT	Java AWT	Java Swing
1)	Các thành phần AWT là phụ thuộc nền tảng .	Các thành phần Java swing là độc lập nền tảng .
2)	Các thành phần AWT là nặng .	Các thành phần Swing là nhẹ .
3)	AWT không hỗ trợ plugin .	Swing có hỗ trợ plugin .
4)	AWT cung cấp ít thành phần hơn Swing.	Swing cung cấp nhiều thành phần mạnh mẽ hơn AWT như tables, lists, scrollpanes, colorchooser, tabbedpane, vv.
5)	AWT không tuân theo MVC (Model View Controller) trong đó model biểu diễn data, view biểu diễn hiển thị và controller biểu diễn các action để kết nối model với view.	Swing tuân theo mô hình MVC .

5.1.2 JFC là gì?

Java Foundation Class (JFC) là một bộ các thành phần GUI đơn giản hóa sự phát triển của các ứng dụng desktop.

5.1.3 Phân cấp các lớp Java Swing

Hệ thống phân cấp của API java swing được đưa ra dưới đây.



5.1.4 Các phương thức thường dùng của lớp Component

Các phương thức của lớp Component được sử dụng rộng rãi trong java swing được đưa ra dưới đây.

Phương thức	Mô tả
<code>public void add(Component c)</code>	thêm một thành phần vào thành phần khác.
<code>public void setSize(int width, int height)</code>	thiết lập kích thước của thành phần.
<code>public void setLayout(LayoutManager m)</code>	thiết lập trình quản lý bố cục (layout) cho

	thành phần.
<code>public void setVisible(boolean b)</code>	thiết lập khả năng hiển thị của thành phần. Nó theo mặc định là false (ẩn)

5.1.5 Ví dụ về Java Swing

Có hai cách để tạo khung (Frame):

- Bằng cách tạo đối tượng của lớp JFrame.
- Bằng cách kế thừa lớp JFrame.

Chúng ta có thể viết code của Swing bên trong hàm `main()`, constructor hoặc bất kỳ phương thức nào khác.

5.1.5.1 Ví dụ Swing Java đơn giản

Chúng ta hãy xem một ví dụ swing đơn giản, nơi chúng ta đang tạo một button và thêm nó vào đối tượng JFrame bên trong phương thức `main()`.

File: FirstSwingExample.java

```
package vn.plpsoft.swing;

import javax.swing.JButton;
import javax.swing.JFrame;

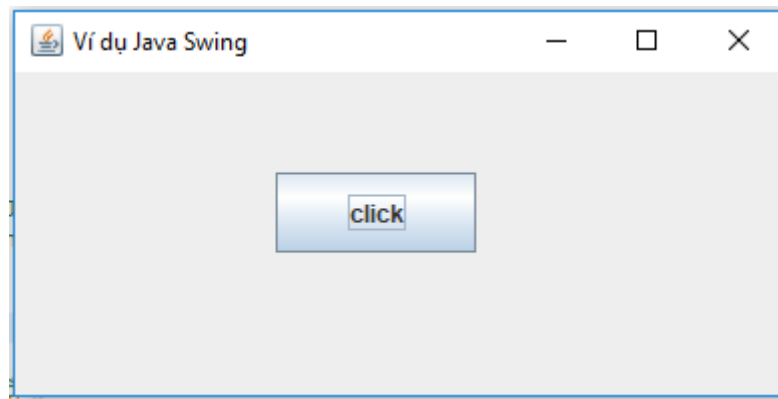
public class FirstSwingExample {
    public static void main(String[] args) {
        JFrame f = new JFrame();// tạo thể hiện của JFrame

        JButton b = new JButton("click");// tạo thể hiện của JButton
        b.setBounds(130, 50, 100, 40);// trục x , y , width, height

        f.setTitle("Ví dụ Java Swing");
        f.add(b);// thêm button vào JFrame

        f.setSize(400, 200);// thiết lập kích thước cho cửa sổ
        f.setLayout(null);// không sử dụng trình quản lý bố cục
        f.setVisible(true);// hiển thị cửa sổ
    }
}
```

Kết quả:



5.1.5.2 Ví dụ Java Swing - tạo đối tượng của lớp JFrame

Chúng ta cũng có thể viết tất cả các mã tạo JFrame, JButton bên trong constructor.

```
package vn.plpsoft.swing;

import javax.swing.JButton;
import javax.swing.JFrame;

public class JavaSwingExample2 {
    JFrame f;

    public JavaSwingExample2() {
        f = new JFrame();// tạo thể hiện của JFrame

        JButton b = new JButton("click");// tạo thể hiện của JButton
        b.setBounds(130, 50, 100, 40);

        f.add(b);// thêm button vào JFrame

        f.setSize(400, 200);// thiết lập kích thước cho cửa sổ
        f.setLayout(null);// không sử dụng trình quản lý bố cục
        f.setVisible(true);// hiển thị cửa sổ
    }
}
```

5.1.5.3 Ví dụ Java Swing - kế thừa lớp JFrame

Chúng ta cũng có thể kế thừa lớp JFrame, vì vậy không cần phải tạo thể hiện của lớp JFrame.

```
package vn.plpsoft.swing;

import javax.swing.JButton;
import javax.swing.JFrame;

public class JavaSwingExample3 extends JFrame { // kế thừa lớp JFrame

    public JavaSwingExample3() {
        JButton b = new JButton("click");// tạo button
        b.setBounds(130, 50, 100, 40);

        add(b); // thêm button vào JFrame
        setSize(400, 200);
        setLayout(null);
        setVisible(true);
    }

    public static void main(String[] args) {
        new JavaSwingExample3();
    }
}
```

5.2 Các control giao diện cơ bản

5.2.1 Lớp JLabel trong Java Swing

Lớp **JLabel** trong Java Swing có thể hiển thị hoặc text, hoặc hình ảnh hoặc cả hai. Các nội dung của Label được gán bởi thiết lập căn chỉnh ngang và dọc trong khu vực hiển thị của nó. Theo mặc định, các label được căn chỉnh theo chiều dọc trong khu vực hiển thị. Theo mặc định, text-only label là căn chỉnh theo cạnh, image-only label là căn chỉnh theo chiều ngang.

5.2.1.1 Cú pháp của lớp javax.swing.JLabel

Cú pháp khai báo cho lớp **javax.swing.JLabel** là:

```
public class JLabel extends JComponent  
implements SwingConstants, Accessible
```

Lớp javax.swing.JLabel có một trường là **protected Component labelFor**.

5.2.1.2 Constructor của lớp JLabel trong Java Swing

- 1. JLabel():** Tạo một instance của JLabel, không có hình ảnh, và với một chuỗi trống cho title.
- 2. JLabel(Icon image):** Tạo một instance của JLabel với hình ảnh đã cho.
- 3. JLabel(Icon image, int horizontalAlignment):** Tạo một instance của JLabel với hình ảnh và căn chỉnh ngang đã cho.
- 4. JLabel(String text):** Tạo một instance của JLabel với text đã cho.
- 5. JLabel(String text, Icon icon, int horizontalAlignment):** Tạo một instance của JLabel với text, hình ảnh, và căn chỉnh ngang đã cho.
- 7. JLabel(String text, int horizontalAlignment):** Tạo một instance của JLabel với text và căn chỉnh ngang đã cho.

5.2.1.3 Một số phương thức thường được sử dụng của lớp JLabel:

STT	Phương thức & Mô tả
1	void setDisabledIcon(Icon disabledIcon) Thiết lập icon để được hiển thị nếu JLabel này là "disabled" (JLabel.setEnabled(false))
2	void setDisplayedMnemonic(char aChar) Xác định displayedMnemonic như là một giá trị char
3	void setDisplayedMnemonic(int key) Xác định keycode mà chỉ dẫn một mnemonic key
4	void setDisplayedMnemonicIndex(int index) Cung cấp một hint cho L&F, từ đó ký tự trong text nên được trang trí để biểu diễn mnemonic
5	void setHorizontalAlignment(int alignment) Thiết lập sự căn chỉnh nội dung của label theo trục X
6	void setHorizontalTextPosition(int textPosition) Thiết lập vị trí theo chiều ngang của phần text của label, cân xứng với hình ảnh của nó
7	void setIcon(Icon icon) Định nghĩa icon mà thành phần này sẽ hiển thị

8	void setIconTextGap(int iconTextGap) Nếu cả hai thuộc tính icon và text được thiết lập, thuộc tính này xác định khoảng trống giữa chúng
9	void setLabelFor(Component c) Thiết lập thành phần đang gán nhãn cho
10	void setText(String text) Định nghĩa trường text dòng đơn mà thành phần này sẽ hiển thị
11	void setUI(LabelUI ui) Thiết lập đối tượng L&F mà biểu diễn thành phần này
12	void setVerticalAlignment(int alignment) Thiết lập sự căn chỉnh nội dung của label theo trục Y
13	void setVerticalTextPosition(int textPosition) Thiết lập vị trí theo chiều dọc của phần text của label, cân xứng với hình ảnh của nó
14	void updateUI() Phục hồi thuộc tính UI về một giá trị từ L&F hiện tại

5.2.1.4 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.2.1.5 Chương trình ví dụ lớp JLabel

SwingJLabelExam1.java

```
package vn.plpsoft.swing;  
  
import java.awt.Color;  
import java.awt.FlowLayout;  
import java.awt.GridLayout;
```



```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class SwingJlabelExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

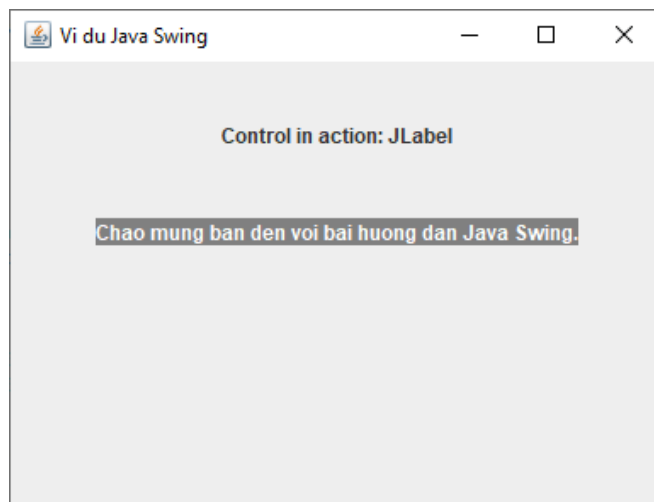
    public SwingJlabelExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        SwingJlabelExam1 swingJlabelExam1 = new SwingJlabelExam1();
        swingJlabelExam1.showLabelDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
    }
}
```

```
        mainFrame.setVisible(true);  
    }  
  
    private void showLabelDemo() {  
        headerLabel.setText("Control in action: JLabel");  
        JLabel label = new JLabel("", JLabel.CENTER);  
        label.setText("Chao mung ban den voi bai huong dan Java Swing.");  
        label.setOpaque(true);  
        label.setBackground(Color.GRAY);  
        label.setForeground(Color.WHITE);  
        controlPanel.add(label);  
        mainFrame.setVisible(true);  
    }  
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.2 Lớp JButton trong Java Swing

Lớp **JButton** trong Java Swing được sử dụng để tạo một nút có thể click. Thành phần này có một label và tạo một sự kiện (event) khi được nhấn. Bạn cũng có thể chèn ảnh vào button.

5.2.2.1 Cú pháp của lớp `javax.swing.JButton`

Cú pháp khai báo cho lớp `javax.swing.JButton` là:

```
public class JButton extends AbstractButton  
    implements Accessible
```

5.2.2.2 Lớp này kế thừa các phương thức từ các lớp sau:

- `javax.swing.AbstractButton`
- `javax.swing.JComponent`

- java.awt.Container
- java.awt.Component
- java.lang.Object

5.2.2.3 Lớp JButton có các constructor sau:

1. **JButton()**: Tạo một button mà không thiết lập text hoặc icon.
2. **JButton(Action a)**: Tạo một button tại đây các thuộc tính được nhận từ Action đã cung cấp.
3. **JButton(Icon icon)**: Tạo một button với một icon.
4. **JButton(String text)**: Tạo một button với text.
5. **JButton(String text, Icon icon)**: Tạo một button với text ban đầu và một icon.

5.2.2.4 Các phương thức được sử dụng phổ biến của lớp JButton

1. **void removeNotify()**: Ghi đè JComponent.removeNotify để kiểm tra xem button này hiện tại được thiết lập như là button mặc định trên RootPane hay không, và nếu có, thiết lập button mặc định của RootPane về null để bảo đảm rằng Rootpane không giữ một tham chiếu button không hợp lệ.
2. **void setDefaultCapable(boolean defaultCapable)**: Thiết lập thuộc tính defaultCapable, mà quyết định xem có hay không button này có thể được tạo là button mặc định cho Root Pane của nó
3. **void updateUI()**: Phục hồi thuộc tính UI về một giá trị từ L&F hiện tại

5.2.2.5 Một số phương thức được sử dụng phổ biến của của lớp AbstractButton

1. **public void setText(String s)**: được sử dụng để thiết lập text đã cho trên button.
 2. **public String getText()**: được sử dụng để trả về text của button.
 3. **public void setEnabled(boolean b)**: được sử dụng để kích hoạt hoặc vô hiệu hóa button.
 4. **public void setIcon(Icon b)**: được sử dụng để thiết lập Icon đã cho trên button.
 5. **public Icon getIcon()**: được sử dụng để lấy Icon của button.
 6. **public void setMnemonic(int a)**: được sử dụng để thiết lập thuộc tính mnemonic trên button.
 7. **public void addActionListener(ActionListener a)**: được sử dụng để thêm action listener tới đối tượng này.
-

5.2.2.6 Chương trình đơn giản để hiển thị hình ảnh trên button

SwingJButtonExam1.java

```
package vn.plpsoft.swing;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;

public class SwingJButtonExam1 {
    SwingJButtonExam1() {
        JFrame f = new JFrame();

        JButton b = new JButton(new ImageIcon("b.jpg"));
        b.setBounds(130, 100, 100, 40);

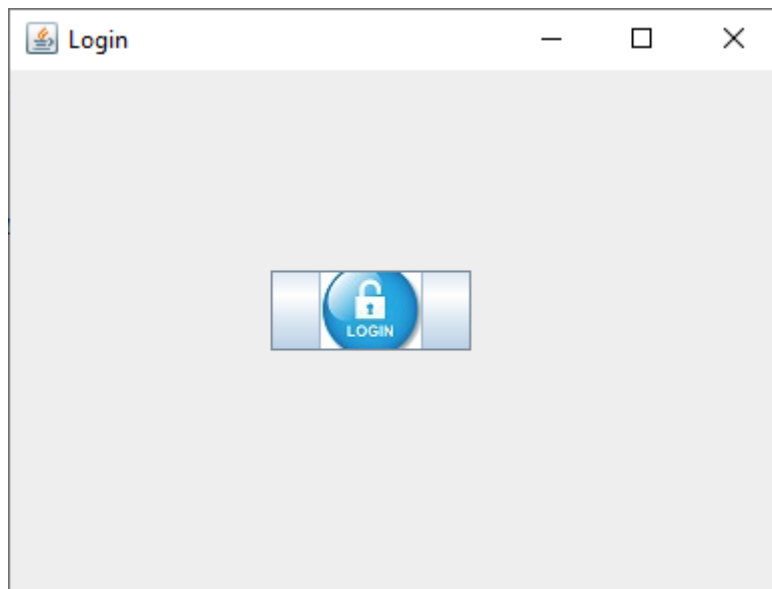
        f.add(b);

        f.setSize(400, 300);
        f.setLayout(null);
        f.setVisible(true);

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        new SwingJButtonExam1();
    }
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.2.7 Chương trình ví dụ khác về lớp JButton

SwingJButtonExam2.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.SwingConstants;

public class SwingJButtonExam2 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public SwingJButtonExam2 () {
```

```
        prepareGUI();
    }

    public static void main(String[] args) {
        SwingJButtonExam2 swingJButtonExam2 = new
SwingJButtonExam2();
        swingJButtonExam2.showButtonDemo();
    }

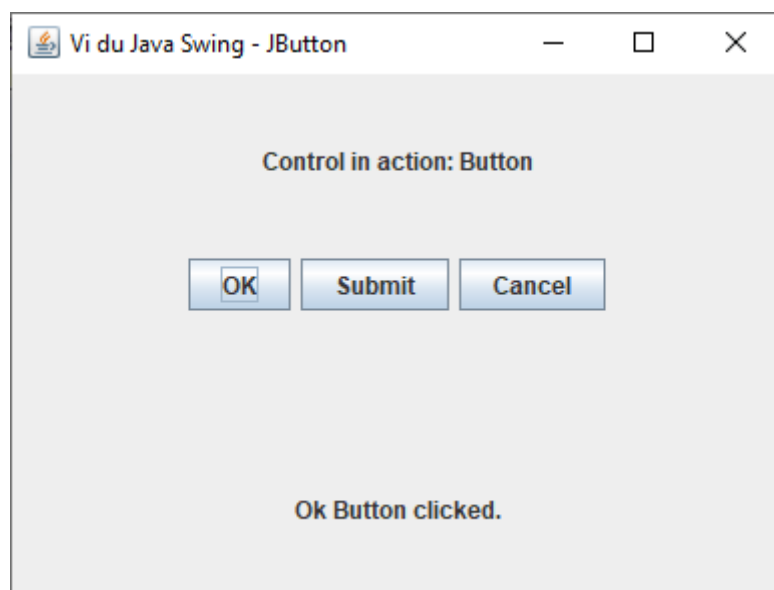
    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing - JButton");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showButtonDemo() {
        headerLabel.setText("Control in action: Button");
        JButton okButton = new JButton("OK");
        JButton javaButton = new JButton("Submit");
        JButton cancelButton = new JButton("Cancel");
        cancelButton.setHorizontalTextPosition(SwingConstants.LEFT);

        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                statusLabel.setText("Ok Button clicked.");
            }
        });
    }
}
```

```
        }  
    });  
    javaButton.addActionListener(new ActionListener() {  
        public void actionPerformed(ActionEvent e) {  
            statusLabel.setText("Submit Button clicked.");  
        }  
    });  
    cancelButton.addActionListener(new ActionListener() {  
        public void actionPerformed(ActionEvent e) {  
            statusLabel.setText("Cancel Button clicked.");  
        }  
    });  
    controlPanel.add(okButton);  
    controlPanel.add(javaButton);  
    controlPanel.add(cancelButton);  
    mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.3 Lớp JTable trong Java Swing

Lớp JTable trong Java Swing được sử dụng để hiển thị dữ liệu trên các ô của bảng hai chiều. Các constructor được sử dụng phổ biến của lớp JTable là:

1. JTable(): Tạo một bảng với các ô trống.

2. JTable(Object[][] rows, Object[] columns): Tạo một bảng với dữ liệu đã cho.

Ví dụ về lớp JTable trong Java Swing

```
package vn.plpsoft.swing;

import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class SwingJTableExam1 {
    JFrame f;

    SwingJTableExam1() {
        f = new JFrame();

        String data[][] = { { "101", "Tran Van Minh", "6000" },
                             { "102", "Phan Van Tai", "8000" },
                             { "101", "Do Cao Hoc", "7000" } };

        String column[] = { "ID", "NAME", "SALARY" };

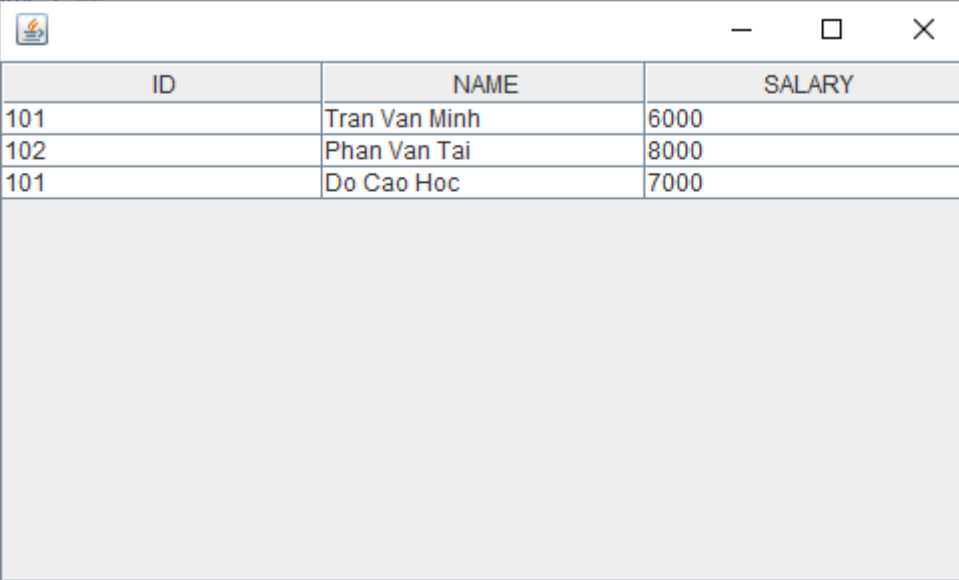
        JTable jt = new JTable(data, column);
        jt.setBounds(30, 40, 200, 300);

        JScrollPane sp = new JScrollPane(jt);
        f.add(sp);

        f.setSize(500, 300);
        f.setVisible(true);
    }

    public static void main(String[] args) {
        new SwingJTableExam1();
    }
}
```

Chạy chương trình Java trên cho kết quả như sau:



The screenshot shows a standard Java Swing window with a title bar containing a small icon, a minus button, a maximize button, and a close button. The window contains a table with three columns: ID, NAME, and SALARY. The table has three data rows. Below the table is a large, empty rectangular area, likely intended for additional graphical output or text.

ID	NAME	SALARY
101	Tran Van Minh	6000
102	Phan Van Tai	8000
101	Do Cao Hoc	7000

5.2.4. Lớp Graphics trong Java Swing

Lớp **Graphics** trong Java cung cấp nhiều phương thức để lập trình đồ họa.

5.2.4.1 Các phương thức của lớp Graphics

Dưới đây là một số phương thức được sử dụng phổ biến của lớp Graphics:

public abstract void drawString(String str, int x, int y): được sử dụng để vẽ chuỗi đã cho.

public void drawRect(int x, int y, int width, int height): vẽ một hình chữ nhật với độ rộng width và chiều cao height đã cho.

public abstract void fillRect(int x, int y, int width, int height): được sử dụng để điền màu mặc định và độ rộng và chiều cao đã cho vào hình chữ nhật.

public abstract void drawOval(int x, int y, int width, int height): được sử dụng để vẽ hình Oval với độ rộng và chiều cao đã cho.

public abstract void fillOval(int x, int y, int width, int height): được sử dụng để điền màu mặc định và độ rộng và chiều cao đã cho vào hình Oval.

public abstract void drawLine(int x1, int y1, int x2, int y2): được sử dụng để vẽ line giữa hai điểm có tọa độ lần lượt là (x1, y1) và (x2, y2).

public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer): được sử dụng để vẽ hình ảnh đã cho.

public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle): được sử dụng để vẽ đường tròn circular hoặc elip.

public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle): được sử dụng để điền một hình tròn hoặc elip.

public abstract void setColor(Color c): được sử dụng để thiết lập màu hiện tại của đồ họa thành màu color đã cho.

public abstract void setFont(Font font): được sử dụng để thiết lập font hiện tại của đồ họa thành font đã cho.

5.2.4.2 Chương trình ví dụ về lớp Graphics trong Java Swing

```
package vn.plpsoft.swing;

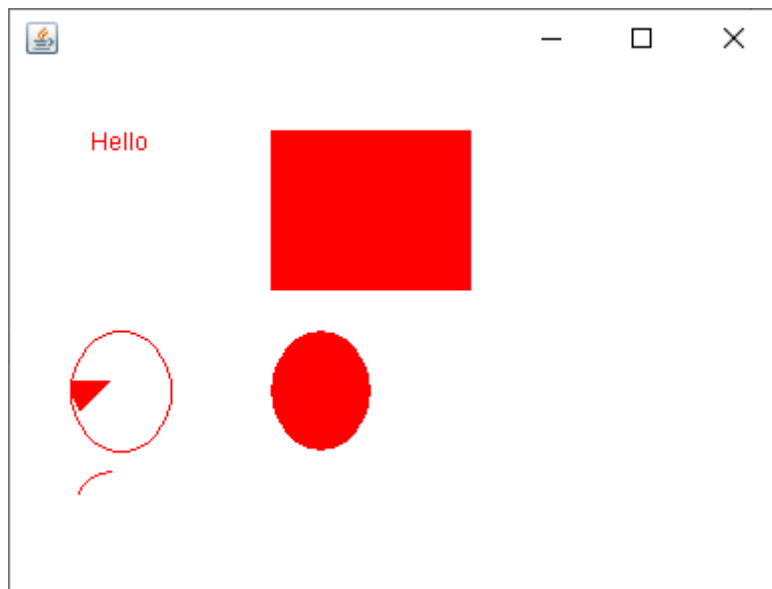
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Graphics;

import javax.swing.JFrame;

public class DisplayGraphics extends Canvas {
    public void paint(Graphics g) {
        g.drawString("Hello", 40, 40);
        setBackground(Color.WHITE);
        g.fillRect(130, 30, 100, 80);
        g.drawOval(30, 130, 50, 60);
        setForeground(Color.RED);
        g.fillOval(130, 130, 50, 60);
        g.drawArc(30, 200, 40, 50, 90, 60);
        g.fillArc(30, 130, 40, 50, 180, 40);
    }

    public static void main(String[] args) {
        DisplayGraphics m = new DisplayGraphics();
        JFrame f = new JFrame();
        f.add(m);
        f.setSize(400, 300);
        f.setVisible(true);
    }
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.5 Lớp JColorChooser trong Java Swing

Lớp **JColorChooser** trong **Java Swing** cung cấp một pane cho các control được thiết kế để cho phép người dùng thao tác và lựa chọn một màu. Cú pháp khai báo cho lớp **javax.swing.JColorChooser** là:

```
public class JColorChooser extends JComponent
implements Accessible
```

5.2.5.1 Lớp này kế thừa các phương thức từ các lớp sau

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.2.5.2 Lớp JColorChooser này có các trường sau

- **protected AccessibleContext accessibleContext**
- **static String CHOOSER_PANELS_PROPERTY** – là tên thuộc tính của mảng chooserPanel.
- **static String PREVIEW_PANEL_PROPERTY** – là tên thuộc tính preview panel.
- **static String SELECTION_MODEL_PROPERTY** – là tên thuộc tính selection model.

5.2.5.3 Lớp JColorChooser có 3 constructor sau:

- 1. JColorChooser():** Tạo một bảng chọn màu với một màu ban đầu là màu trắng.
- 2. JColorChooser(Color initialColor):** Tạo một bảng chọn màu với màu khởi tạo đã cho.
- 3. JColorChooser(ColorSelectionModel model):** Tạo một bảng chọn màu với ColorSelectionModel đã cho.

5.2.5.4 Một số phương thức được sử dụng phổ biến của lớp JColorChooser

STT	Phương thức & Mô tả
1	void addChooserPanel(AbstractColorChooserPanel panel) Thêm một bảng chọn màu tới ColorChooser
2	static JDialog createDialog(Component c, String title, boolean modal, JColorChooser chooserPane, ActionListener okListener, ActionListener cancelListener) Tạo và trả về một hộp thoại dialog chứa bảng ColorChooser cùng với các nút "OK", "Cancel", và "Reset"
3	AbstractColorChooserPanel removeChooserPanel(AbstractColorChooserPanel panel) Xóa Color Panel đã cho
4	void setChooserPanels(AbstractColorChooserPanel[] panels) Xác định Color Panels được sử dụng để chọn một giá trị màu
5	void setColor(Color color) Thiết lập màu hiện tại của Color Chooser tới màu đã cho
6	void setColor(int c) Thiết lập màu hiện tại của Color Chooser tới màu đã cho
7	void setColor(int r, int g, int b) Thiết lập màu hiện tại của Color Chooser tới màu RGB đã cho
8	void setDragEnabled(boolean b) Thiết lập thuộc tính dragEnabled, mà phải là true để kích hoạt bộ xử lý hoạt động drag tự động (phần đầu tiên của hoạt động drag và drop) trên thành phần này
9	void setPreviewPanel(JComponent preview) Thiết lập Preview Panel hiện tại

10	void setSelectionModel(ColorSelectionModel newModel) Thiết lập Model chứa màu đã chọn
11	void setUI(ColorChooserUI ui) Thiết lập đối tượng L&F mà truyền đối tượng này
12	static Color showDialog(Component component, String title, Color initialColor) Hiển thị một hộp thoại dialog
13	void updateUI() Thông báo từ UIManager rằng L&F đã thay đổi

5.2.5.5 Chương trình ví dụ về lớp JColorChooser

JColorChooserExam1.java

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JFrame;

public class JColorChooserExam1 extends JFrame
    implements ActionListener {
    JButton b;
    Container c;

    JColorChooserExam1() {
        c = getContentPane();
        c.setLayout(new FlowLayout());

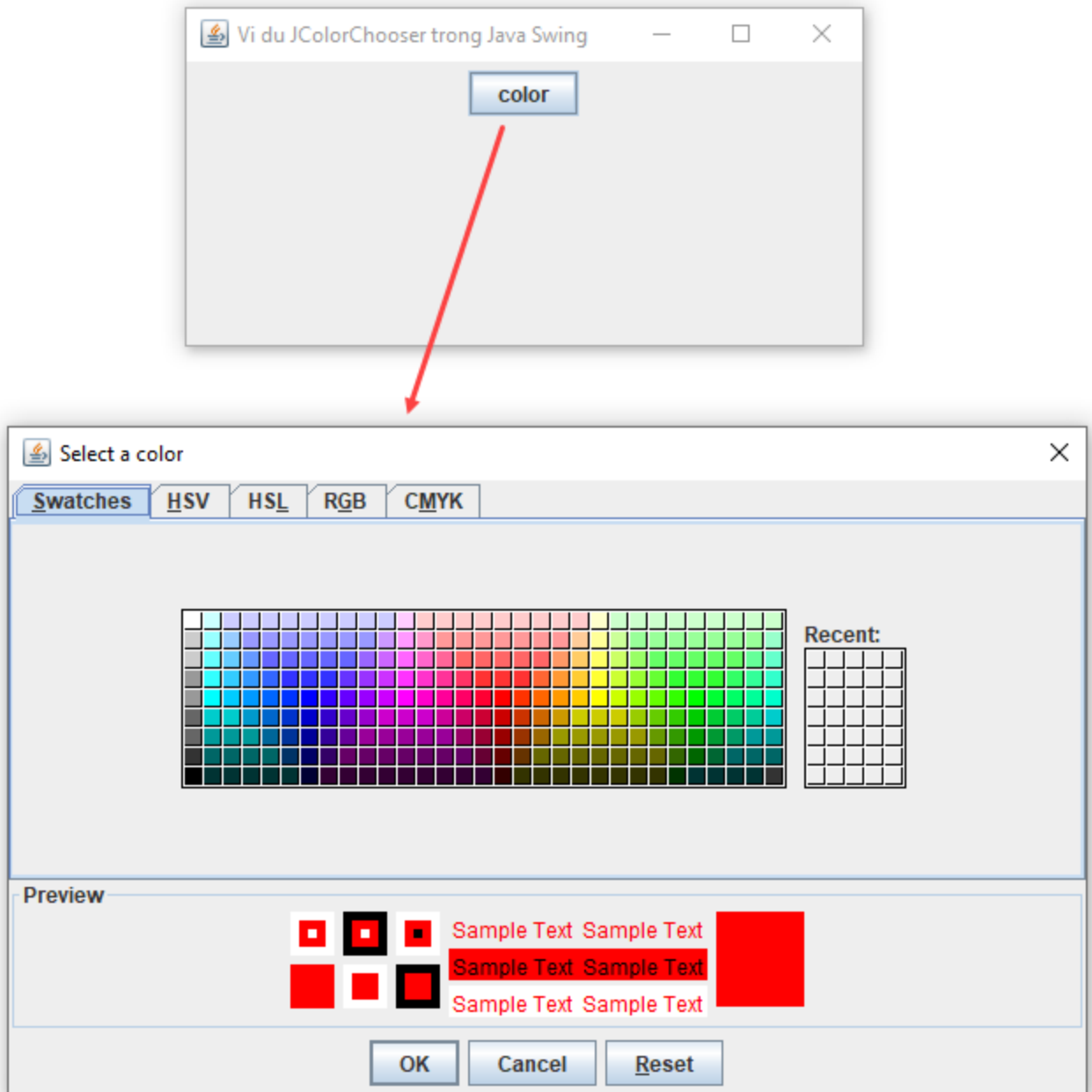
        b = new JButton("color");
        b.addActionListener(this);
    }
}
```

```
        c.add(b);
    }

    public void actionPerformed(ActionEvent e) {
        Color initialcolor = Color.RED;
        Color color = JColorChooser.showDialog(this,
            "Select a color", initialcolor);
        c.setBackground(color);
    }

    public static void main(String[] args) {
        JColorChooserExam1 ch = new JColorChooserExam1();
        ch.setSize(400, 200);
        ch.setVisible(true);
        ch.setTitle("Vi du JColorChooser trong Java Swing");
        ch.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.5.6 Chương trình ví dụ khác về lớp JColorChooser

JColorChooserExam2.java

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
```

```
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JColorChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class JColorChooserExam2 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JColorChooserExam2() {
        prepareGUI();
    }

    public static void main(String[] args) {
        JColorChooserExam2 swingControlDemo = new
JColorChooserExam2();
        swingControlDemo.showColorChooserDemo();
    }

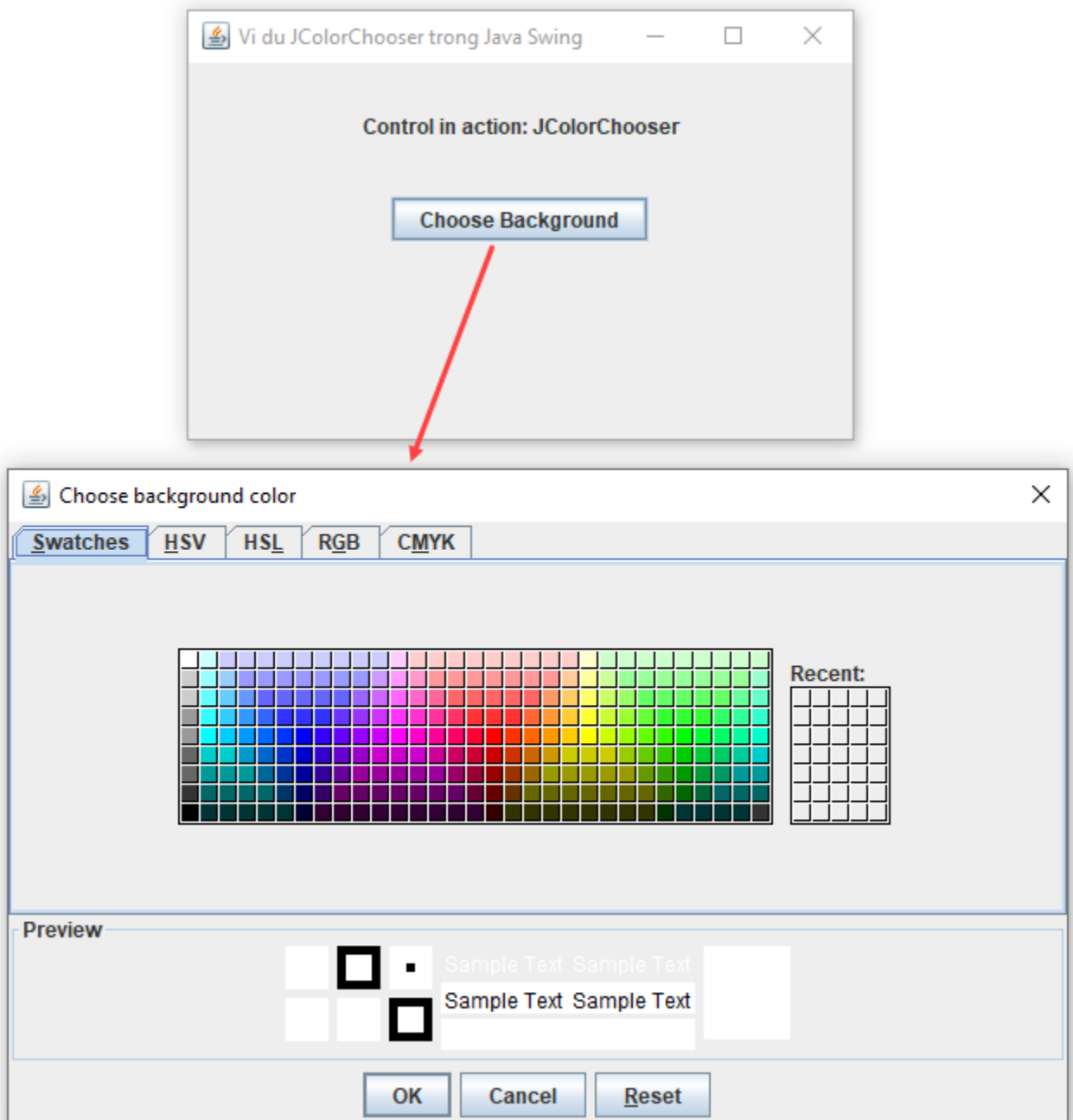
    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
```



```
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showColorChooserDemo() {
        headerLabel.setText("Control in action: JColorChooser");
        JButton chooseButton = new JButton("Choose Background");
        chooseButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Color backgroundColor =
JColorChooser.showDialog(mainFrame,
                    "Choose background color", Color.white);
                if (backgroundColor != null) {
                    controlPanel.setBackground(backgroundColor);
                    mainFrame.getContentPane().setBackground(backgro
undColor);
                }
            }
        });
        controlPanel.add(chooseButton);
        mainFrame.setTitle("Vi du JColorChooser trong Java Swing");
        mainFrame.setVisible(true);
    }
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.6 Lớp JCheckBox trong Java Swing

Lớp **JCheckBox** trong Java Swing là một trình triển khai của một checkbox, là một item mà có thể được lựa chọn (selected) hoặc không được lựa chọn (unselected), và hiển thị trạng thái của nó tới người dùng.

Sau đây là phần khai báo cho lớp **javax.swing.JCheckBox**:

```
public class JCheckBox extends JToggleButton
implements Accessible
```

5.2.6.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.AbstractButton
- javax.swing.JToggleButton
- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

Lớp javax.swing.JCheckBox có trường **static String BORDER_PAINTED_FLAT_CHANGED_PROPERTY**. Trường này nhận diện một thay đổi tới thuộc tính flat.

5.2.6.2 Các constructor của lớp JCheckBox

1. **JCheckBox()**: Tạo một unselected checkbox ban đầu không có text và icon.
2. **JCheckBox(Action a)**: Tạo một checkbox, với các thuộc tính được lấy từ Action đã cho.
3. **JCheckBox(Icon icon)**: Tạo một unselected checkbox với một icon.
4. **JCheckBox(Icon icon, boolean selected)**: Tạo một checkbox với một icon và xác định rằng ban đầu nó là selected hoặc không .
5. **JCheckBox(String text)**: Tạo một unselected checkbox ban đầu với text.
6. **JCheckBox(String text, boolean selected)**: Tạo một checkbox với text và xác định rằng ban đầu nó là selected hoặc không.
7. **JCheckBox(String text, Icon icon)**: Tạo một unselected checkbox ban đầu với text và icon đã cho.
8. **JCheckBox(String text, Icon icon, boolean selected)**: Tạo một checkbox với text và icon, và xác định rằng ban đầu nó là selected hoặc không.

5.2.6.3 Các phương thức của lớp JCheckBox trong Java Swing

AccessibleContext getAccessibleContext(): Lấy AccessibleContext được liên kết với JCheckBox này.

String getUIClassID(): Trả về một chuỗi xác định tên của lớp L&F mà truyền thành phần này.

boolean isBorderPaintedFlat(): Lấy giá trị của thuộc tính borderPaintedFlat.

protected String paramString(): Trả về một biểu diễn chuỗi của JCheckBox này.

void setBorderPaintedFlat(boolean b): Thiết lập thuộc tính borderPaintedFlat, mà cung cấp một hint tới L&F tới bề mặt của đường viền của checkbox.

void updateUI(): Phục hồi thuộc tính UI về một giá trị từ L&F hiện tại.

5.2.6.4 Chương trình ví dụ về lớp JCheckBox

JCheckBoxExam1.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class JCheckBoxExam1 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JCheckBoxExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        JCheckBoxExam1 swingControlDemo = new JCheckBoxExam1();
        swingControlDemo.showCheckBoxDemo();
    }

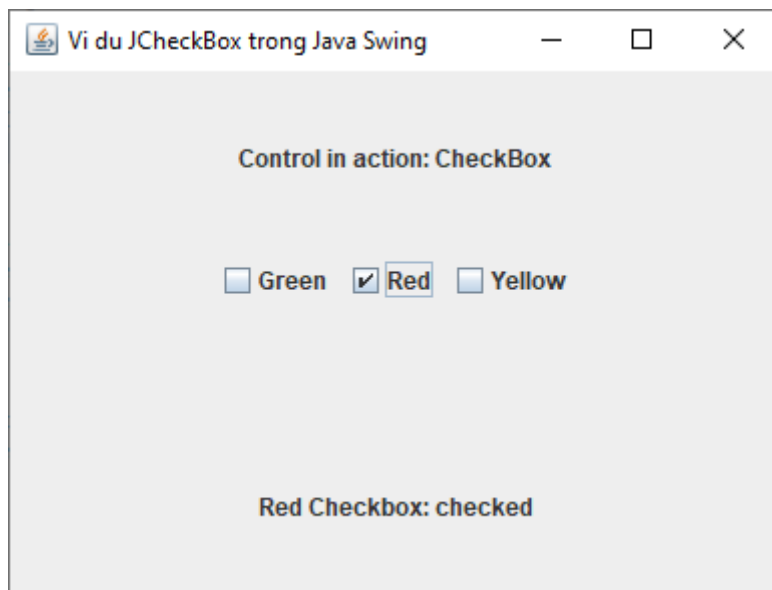
    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JCheckBox trong Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
    }
}
```

```
mainFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
});
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setSize(350, 100);
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
}

private void showCheckBoxDemo() {
    headerLabel.setText("Control in action: CheckBox");
    final JCheckBox chkApple = new JCheckBox("Green");
    final JCheckBox chkMango = new JCheckBox("Red");
    final JCheckBox chkPeer = new JCheckBox("Yellow");
    chkApple.setMnemonic(KeyEvent.VK_C);
    chkMango.setMnemonic(KeyEvent.VK_M);
    chkPeer.setMnemonic(KeyEvent.VK_P);
    chkApple.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Green Checkbox: "
                + (e.getStateChange() == 1 ? "checked" : "unchecked"));
        }
    });
    chkMango.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Red Checkbox: "
                + (e.getStateChange() == 1 ? "checked" : "unchecked"));
        }
    });
    chkPeer.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            statusLabel.setText("Yellow Checkbox: "
```

```
        + (e.getStateChange() == 1 ? "checked" : "unchecked"));
    }
});
controlPanel.add(chkApple);
controlPanel.add(chkMango);
controlPanel.add(chkPeer);
mainFrame.setVisible(true);
}
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.7 Lớp JRadioButton trong Java Swing

Lớp **JRadioButton** trong Java Swing là một trình triển khai của một radio button, một item mà có thể được lựa chọn hoặc không, và hiển thị trạng thái của nó tới người dùng. Lớp này nên được thêm vào trong ButtonGroup để chỉ lựa chọn một radio button.

Dưới đây là cú pháp khai báo cho lớp **javax.swing.JRadioButton**:

```
public class JRadioButton extends JToggleButton
    implements Accessible
```

5.2.7.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.AbstractButton
- javax.swing.JToggleButton
- javax.swing.JComponent
- java.awt.Container

- java.awt.Component
- java.lang.Object

5.2.7.1 Các constructor được sử dụng phổ biến của lớp JRadioButton:

JRadioButton(): Tạo một unselected radiobutton không có text.

JRadioButton(String s): Tạo một unselected radiobutton với text đã cho.

JRadioButton(String s, boolean selected): Tạo một radiobutton với text đã cho và trạng thái là selected.

5.2.7.2 Các phương thức được sử dụng phổ biến của lớp JRadioButton

AccessibleContext getAccessibleContext(): Lấy AccessibleContext được liên kết với JRadioButton này.

String getUIClassID(): Trả về tên của lớp L&F mà truyền thành phần này.

protected String paramString(): Trả về biểu diễn chuỗi của JRadioButton này.

void updateUI(): Phục hồi thuộc tính UI về một giá trị từ L&F hiện tại.

5.2.7.3 Chương trình ví dụ về lớp JRadioButton

JRadioButtonExam1.java

```
package vn.plpsoft.swing;

import javax.swing.ButtonGroup;
import javax.swing.JFrame;
import javax.swing.JRadioButton;

public class JRadioButtonExam1 {
    JFrame frame;

    JRadioButtonExam1() {
        frame = new JFrame();

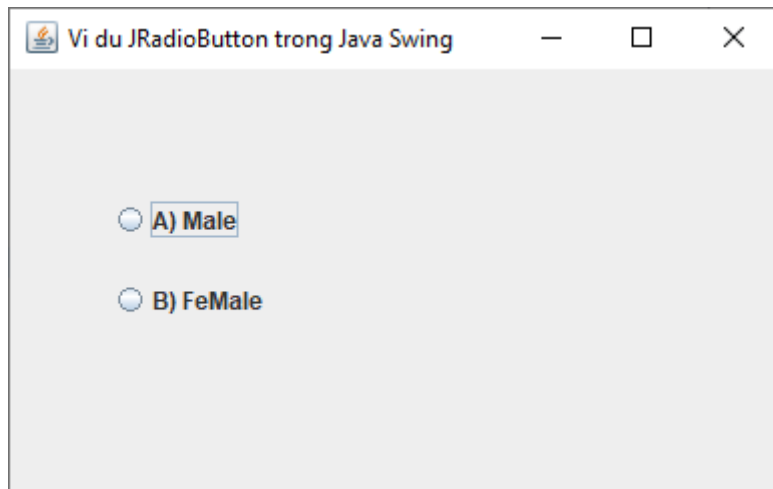
        JRadioButton radioBtn1 = new JRadioButton("A) Male");
        JRadioButton radioBtn2 = new JRadioButton("B) FeMale");
        radioBtn1.setBounds(50, 60, 170, 30);
        radioBtn2.setBounds(50, 100, 170, 30);

        ButtonGroup bg = new ButtonGroup();
        bg.add(radioBtn1);
```

```
        bg.add( radioBtn2 );

        frame.add( radioBtn1 );
        frame.add( radioBtn2 );
        frame.setTitle( "Vi du JRadioButton trong Java Swing" );
        frame.setSize( 400, 250 );
        frame.setLayout( null );
        frame.setVisible( true );
    }

    public static void main( String[] args ) {
        new JRadioButtonExam1 ();
    }
}
```



5.2.7.4 Chương trình ví dụ khác về lớp JRadioButton

JRadioButtonExam2.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.ButtonGroup;
```



```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;

public class JRadioButtonExam2 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

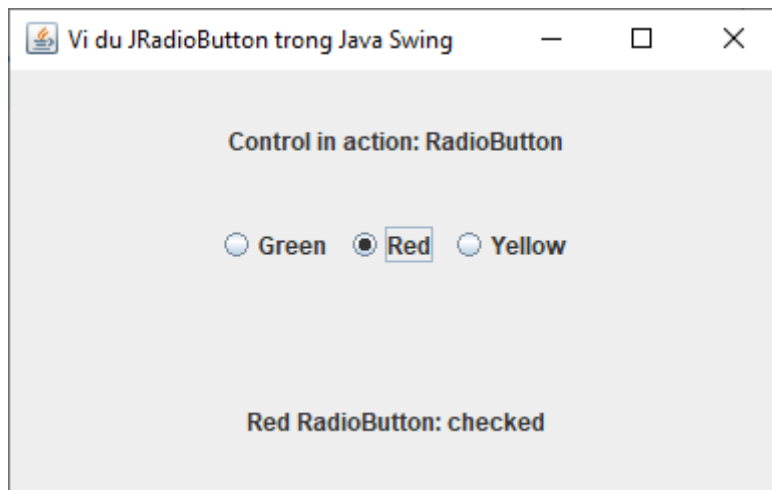
    public JRadioButtonExam2() {
        prepareGUI();
    }

    public static void main(String[] args) {
        JRadioButtonExam2 swingControlDemo = new
JRadioButtonExam2();
        swingControlDemo.showRadioButtonDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JRadioButton trong Java Swing");
        mainFrame.setSize(400, 250);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
    }
}
```

```
        mainFrame.setVisible(true);
    }

    private void showRadioButtonDemo() {
        headerLabel.setText("Control in action: RadioButton");
        final JRadioButton radApple = new JRadioButton("Green", true);
        final JRadioButton radMango = new JRadioButton("Red");
        final JRadioButton radPeer = new JRadioButton("Yellow");
        radApple.setMnemonic(KeyEvent.VK_C);
        radMango.setMnemonic(KeyEvent.VK_M);
        radPeer.setMnemonic(KeyEvent.VK_P);
        radApple.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                statusLabel.setText("Green RadioButton: "
                    + (e.getStateChange() == 1 ? "checked" : "unchecked"));
            }
        });
        radMango.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                statusLabel.setText("Red RadioButton: "
                    + (e.getStateChange() == 1 ? "checked" : "unchecked"));
            }
        });
        radPeer.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent e) {
                statusLabel.setText("Yellow RadioButton: "
                    + (e.getStateChange() == 1 ? "checked" : "unchecked"));
            }
        }); // Group the radio buttons.
        ButtonGroup group = new ButtonGroup();
        group.add(radApple);
        group.add(radMango);
        group.add(radPeer);
        controlPanel.add(radApple);
        controlPanel.add(radMango);
        controlPanel.add(radPeer);
        mainFrame.setVisible(true);
    }
}
```



5.2.8 Lớp ButtonGroup trong Java Swing

Lớp ButtonGroup có thể được sử dụng để nhóm nhiều lớp lại với nhau, để mà tại một thời điểm, chỉ có một nút được lựa chọn. Ví dụ:

```
package vn.plpsoft.swing;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JRadioButton;

public class JRadioButtonExam3 extends JFrame implements
ActionListener {
    JRadioButton rb1, rb2;
    JButton b;

    JRadioButtonExam3() {

        rb1 = new JRadioButton("Green");
        rb1.setBounds(100, 50, 100, 30);

        rb2 = new JRadioButton("Red");
        rb2.setBounds(100, 100, 100, 30);
```

```
        ButtonGroup bg = new ButtonGroup();
        bg.add(rb1);
        bg.add(rb2);

        b = new JButton("click");
        b.setBounds(100, 150, 80, 30);
        b.addActionListener(this);

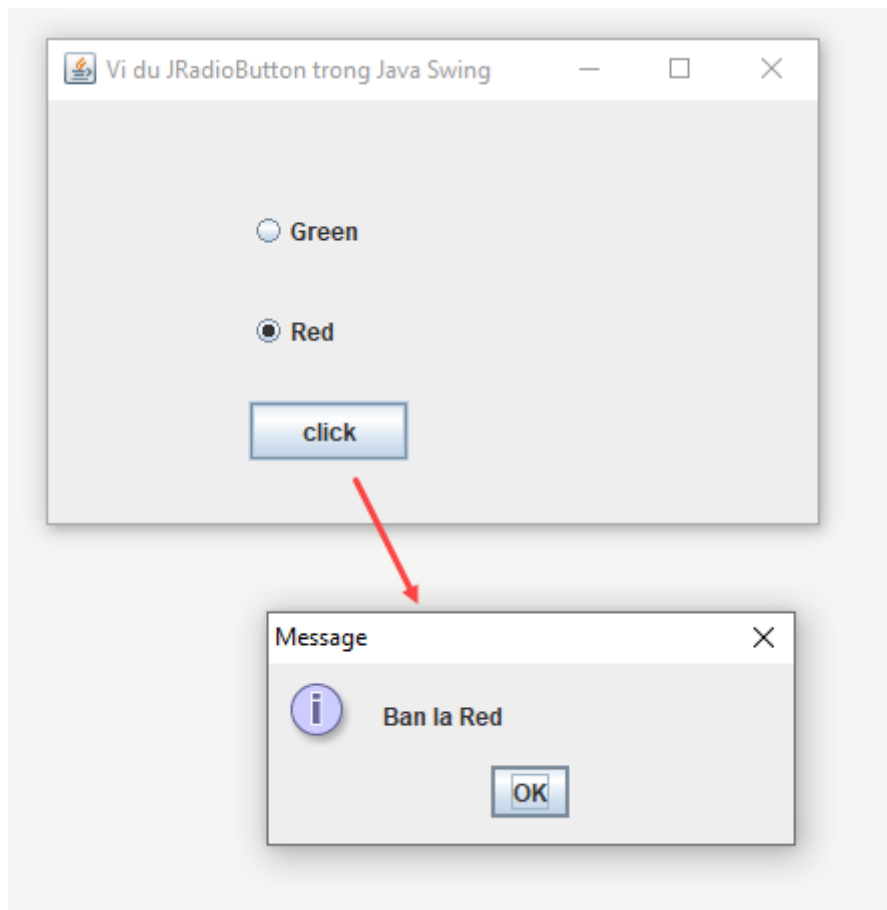
        add(rb1);
        add(rb2);
        add(b);

        setTitle("Vi du JRadioButton trong Java Swing");
        setSize(400, 250);
        setLayout(null);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if (rb1.isSelected()) {
            JOptionPane.showMessageDialog(this, "Ban la Green");
        }
        if (rb2.isSelected()) {
            JOptionPane.showMessageDialog(this, "Ban la Red");
        }
    }

    public static void main(String args[]) {
        new JRadioButtonExam3();
    }
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.9 Lớp JList trong Java Swing

Lớp **JList** trong Java Swing là một thành phần mà hiển thị một danh sách các đối tượng và cho phép người dùng lựa chọn một hoặc nhiều item. Một Model riêng rẽ, `ListModel`, duy trì các nội dung của list. Cú pháp khai báo của lớp **`javax.swing.JList`** là: [?](#)

```
public class JList
    extends JComponent
        implements Scrollable, Accessible
```

5.2.9.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `javax.swing.JComponent`
- `java.awt.Container`
- `java.awt.Component`
- `java.lang.Object`

5.2.9.2 Lớp JList có các trường sau:

- **static int HORIZONTAL_WRAP**: Trình bày một layout theo phong cách như một tờ báo ("newspaper style") với các ô tràn theo chiều ngang và sau đó là chiều dọc.
- **static int VERTICAL**: Chỉ một cách bố trí các ô theo chiều dọc, trong một cột đơn; đây là layout mặc định.
- **static int VERTICAL_WRAP**: Chỉ một layout theo phong cách như một tờ báo "newspaper style" với các ô tràn theo chiều dọc và sau đó là chiều ngang.

5.2.9.3 Các constructor của lớp JList trong Java Swing

JList(): Xây dựng một JList với một model là empty, read-only.

JList(ListModel dataModel): Xây dựng một JList mà hiển thị các phần tử từ model đã cho và non-null.

JList(Object[] listData): Xây dựng một JList mà hiển thị các phần tử trong mảng đã cho.

JList(Vector listData): Xây dựng một JList mà hiển thị các phần tử trong Vector đã cho.

5.2.9.4 Các phương thức của lớp JList trong Java Swing

STT	Phương thức & Mô tả
1	void addListSelectionListener(ListSelectionListener listener) Thêm một Listener tới list, để được thông báo mỗi khi xuất hiện một thay đổi tới selection; đây là cách ưu tiên để nghe các trạng thái của thay đổi
2	void addSelectionInterval(int anchor, int lead) Thiết lập selection thành sự kết hợp của khoảng interval đã cho với selection hiện tại
3	void clearSelection() Xóa selection sau khi gọi phương thức này, isSelectedEmpty sẽ trả về true
4	protected ListSelectionModel createSelectionModel() Trả về một instance của DefaultListSelectionModel; được gọi trong khi xây dựng để khởi tạo thuộc tính selection model của list
5	void ensureIndexIsVisible(int index) Cuộn danh sách bên trong một viewport đang bao quanh để làm cho ô đã cho là hoàn toàn nhìn thấy

6	protected void fireSelectionValueChanged(int firstIndex, int lastIndex, boolean isAdjusting) Thông báo cho ListSelectionListeners để thêm trực tiếp tới danh sách các thay đổi của selection được tạo tới selection model
7	AccessibleContext getAccessibleContext() Lấy AccessibleContext được liên kết với JList này
8	int getFirstVisibleIndex() Trả về chỉ mục nhỏ nhất của list mà có thể nhìn thấy hiện tại
9	int getLastVisibleIndex() Trả về chỉ mục lớn nhất của list mà có thể nhìn thấy hiện tại
10	Dimension getPreferredSize() Tính toán kích cỡ của viewport cần thiết để hiển thị các hàng visibleRowCount
11	int getScrollableUnitIncrement(Rectangle visibleRect, int orientation, int direction) Trả về khoảng cách để cuộn để trưng bày hàng trước đó hoặc khối tiếp theo (với cuộn theo chiều dọc) hoặc cột (với cuộn theo chiều ngang)
12	void removeListSelectionListener(ListSelectionListener listener) Xóa một selection listener từ list này
13	void removeSelectionInterval(int index0, int index1) Thiết lập selection để thiết lập sự khác nhau của interval đã cho và selection hiện tại
14	void setDragEnabled(boolean b) Tắt hoặc bật bộ xử lý hoạt động drag mặc định
15	void setDropMode(DropMode dropMode) Thiết lập drop mode cho thành phần này
16	void setFixedCellHeight(int height) Thiết lập một giá trị cố định để được sử dụng cho chiều cao của mỗi ô trong list
17	void setLayoutOrientation(int layoutOrientation)

	Định nghĩa cách các ô trong list được bố trí
18	void setListData(Object[] listData) Xây dựng một read-only ListModel từ một mảng các đối tượng và gọi setModel với model này
19	void setListData(Vector listData) Xây dựng một read-only ListModel từ một Vector và gọi setModel với model này
20	void setModel(ListModel model) Thiết lập model mà biểu diễn các nội dung hoặc "value" của list, thông báo sự thay đổi thuộc tính tới listener, và sau đó xóa selection của list
21	void setPrototypeCellValue(Object prototypeCellValue) Thiết lập thuộc tính prototypeCellValue, và sau đó (nếu giá trị mới là non-null) thì tính toán các thuộc tính fixedCellWidth và fixedCellHeight bởi yêu cầu thành phần cell renderer cho giá trị đã cho (và chỉ mục 0) từ cell renderer đó, và sử dụng kích cỡ của thành phần đó
22	void setSelectedIndex(int index) Lựa chọn một ô đơn
23	void setSelectedIndices(int[] indices) Thay đổi selection thành là tập hợp các chỉ mục được xác định bởi mảng đã cho
24	void setSelectedValue(Object anObject, boolean shouldScroll) Thiết lập đối tượng đã cho từ list
25	void setSelectionBackground(Color selectionBackground) Thiết lập màu được sử dụng để vẽ màu nền background của item được chọn, mà Call Renderer có thể sử dụng để điền vào ô đã chọn
26	void setSelectionForeground(Color selectionForeground) Thiết lập màu được sử dụng để vẽ foreground của item được chọn, mà Call Renderer có thể sử dụng để truyền text và đồ họa
27	void setSelectionInterval(int anchor, int lead) Thiết lập interval đã cho

28	void setSelectionMode(int selectionMode) Thiết lập selection mode cho list
29	void setSelectionModel(ListSelectionModel selectionModel) Thiết lập selectionModel cho list tới một trình triển khai non-null ListSelectionModel
30	void setUI(ListUI ui) Thiết lập ListUI, đối tượng L&F mà truyền đối tượng này
31	void setValueIsAdjusting(boolean b) Thiết lập thuộc tính valueIsAdjusting của selection model
32	void setVisibleRowCount(int visibleRowCount) Thiết lập thuộc tính visibleRowCount, mà có ý nghĩa khác nhau phụ thuộc vào hướng bố trí layout orientation: Với hướng bố trí VERTICAL, phương thức này thiết lập số hàng ưu tiên để hiển thị (không yêu cầu cuộn); với các hướng khác, phương thức này tác động đến việc bao các ô
33	void updateUI() Phục hồi thuộc tính ListUI bởi thiết lập nó tới giá trị được cung cấp bởi L&F hiện tại

5.2.9.5 Chương trình ví dụ lớp JList

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.DefaultListModel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
```

```
import javax.swing.JScrollPane;
import javax.swing.ListSelectionModel;

public class JListExam1 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JListExam1 () {
        prepareGUI ();
    }

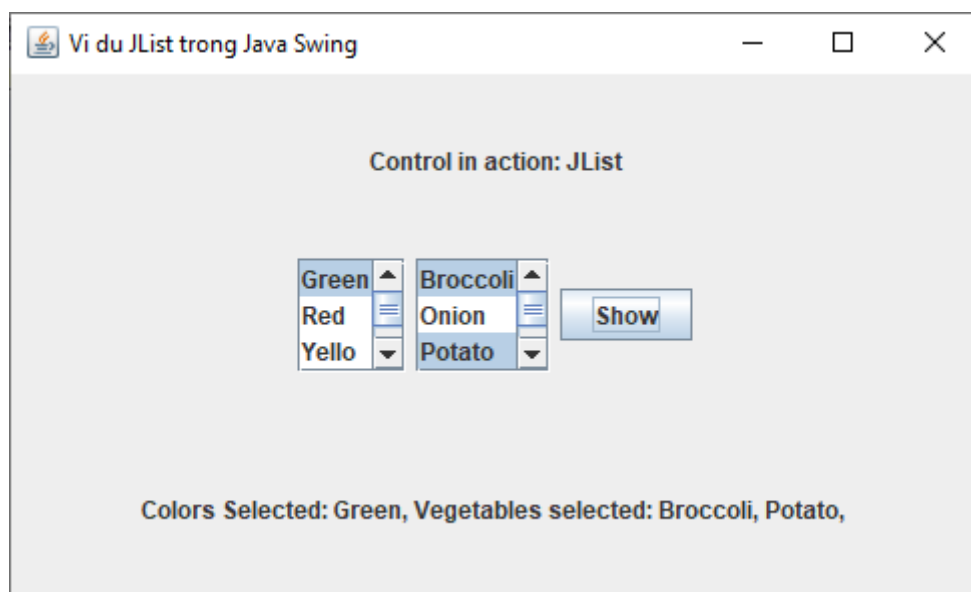
    public static void main(String[] args) {
        JListExam1 jListExam1 = new JListExam1 ();
        jListExam1.showListDemo ();
    }

    private void prepareGUI () {
        mainFrame = new JFrame("Vi du JList trong Java Swing");
        mainFrame.setSize(500, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter () {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
}
```

```
private void showListDemo() {
    headerLabel.setText("Control in action: JList");
    final DefaultListModel colorsName = new
DefaultListModel();
    colorsName.addElement("Green");
    colorsName.addElement("Red");
    colorsName.addElement("Yellow");
    colorsName.addElement("Black");
    final JList fruitList = new JList(colorsName);
fruitList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
    fruitList.setSelectedIndex(0);
    fruitList.setVisibleRowCount(3);
    JScrollPane fruitListScrollPane = new
JScrollPane(fruitList);
    final DefaultListModel vegName = new DefaultListModel();
    vegName.addElement("Broccoli");
    vegName.addElement("Onion");
    vegName.addElement("Potato");
    vegName.addElement("Tomato");
    final JList vegList = new JList(vegName);
vegList.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SE
LECTION);
    vegList.setSelectedIndex(0);
    vegList.setVisibleRowCount(3);
    JScrollPane vegListScrollPane = new JScrollPane(vegList);
    JButton showButton = new JButton("Show");
    showButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String data = "";
            if (fruitList.getSelectedIndex() != -1) {
                data = "Colors Selected: " +
fruitList.getSelectedValue();
                statusLabel.setText(data);
            }
            if (vegList.getSelectedIndex() != -1) {
                data += ", Vegetables selected: ";
                for (Object vegetable :
vegList.getSelectedValues()) {
                    data += vegetable + ", ";
                }
            }
        }
    });
}
```

```
        statusLabel.setText(data);  
    }  
});  
controlPanel.add(fruitListScrollPane);  
controlPanel.add(vegListScrollPane);  
controlPanel.add(showButton);  
  
mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình Java trên cho kết quả như sau:



5.2.10 Lớp JComboBox trong Java Swing

Lớp **JComboBox** trong Java Swing là một thành phần mà kết hợp một button, một trường có thể chỉnh sửa và một drop-down list. Tại một thời điểm chỉ có một item có thể được lựa chọn từ list. Cú pháp khai báo cho lớp **javax.swing.JComboBox** là:

```
public class JComboBox  
    extends JComponent  
        implements ItemSelectable, ListDataListener,  
            ActionListener, Accessible
```

5.2.10.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container

- `java.awt.Component`
- `java.lang.Object`

5.2.10.1 Lớp `JComboBox` bao gồm các trường sau:

- `protected String actionCommand`
- `protected ComboBoxModel dataModel`
- `protected ComboBoxEditor editor`
- `protected boolean isEditable`
- `protected JComboBox.KeySelectionManager keySelectionManager`
- `protected boolean lightWeightPopupEnabled`
- `protected int maximumRowCount`
- `protected ListCellRenderer renderer`
- `protected Object selectedItemReminder`

5.2.10.2 Các constructor được sử dụng phổ biến của lớp `JComboBox`

`JComboBox()`: Tạo một `JComboBox` với data model mặc định.

`JComboBox(Object[] items)`: Tạo một `JComboBox` mà chứa các phần tử trong mảng đã cho.

`JComboBox(Vector items)`: Tạo một `JComboBox` mà chứa các phần tử trong Vector đã cho.

5.2.10.3 Các phương thức được sử dụng phổ biến của lớp `JComboBox`

`public void addItem(Object anObject)`: được sử dụng để thêm một item tới list.

`public void removeItem(Object anObject)`: được sử dụng để xóa một item từ list.

`public void removeAllItems()`: được sử dụng để xóa tất cả item từ list.

`public void setEditable(boolean b)`: được sử dụng để xác định xem có hay không `JComboBox` là editable.

`public void addActionListener(ActionListener a)`: được sử dụng để thêm `ActionListener`.

`public void addItemListener(ItemListener i)`: được sử dụng để thêm `ItemListener`.

5.2.10.4 Chương trình ví dụ đơn giản đầu tiên về lớp `JComboBox` trong Java Swing

```
package vn.plpsoft.swing;

import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;

public class JComboBoxExam1 {
    JFrame f;

    JComboBoxExam1() {
        f = new JFrame("Vi du combobox - java swing");

        String city[] = { "Ha Noi", "Vinh Phuc", "Da Nang",
                           "TP. Ho Chi Minh", "Nha Trang" };

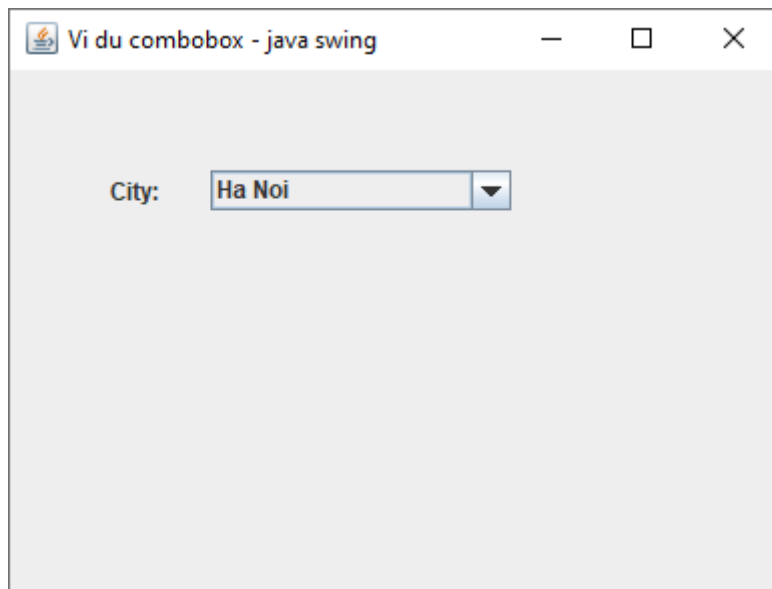
        JComboBox cb = new JComboBox(city);
        cb.setBounds(100, 50, 150, 20);
        f.add(cb);

        JLabel cityLabel = new JLabel("City: ");
        cityLabel.setBounds(50, 50, 80, 20);
        f.add(cityLabel);

        f.setLayout(null);
        f.setSize(400, 300);
        f.setVisible(true);
    }

    public static void main(String[] args) {
        new JComboBoxExam1();
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.10.5 Một chương trình khác về lớp JComboBox

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;

public class JComboBoxExam2 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
```

```
public JComboBoxExam2 () {
    prepareGUI ();
}

public static void main (String[] args) {
    JComboBoxExam2 swingDemo = new JComboBoxExam2 ();
    swingDemo.showComboboxDemo ();
}

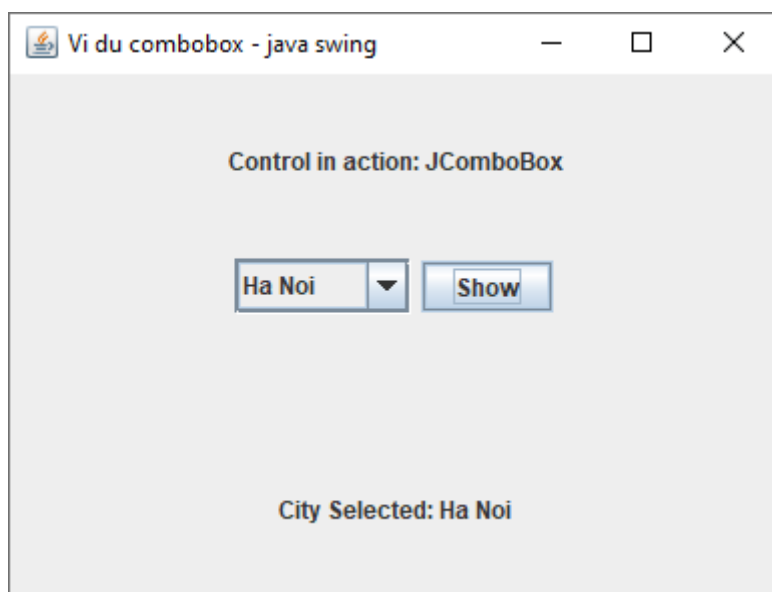
private void prepareGUI () {
    mainFrame = new JFrame ("Vi du combobox - java swing");
    mainFrame.setSize (400, 400);
    mainFrame.setLayout (new GridLayout (3, 1));
    mainFrame.addWindowListener (new WindowAdapter () {
        public void windowClosing (WindowEvent windowEvent) {
            System.exit (0);
        }
    });
    headerLabel = new JLabel ("", JLabel.CENTER);
    statusLabel = new JLabel ("", JLabel.CENTER);
    statusLabel.setSize (350, 100);
    controlPanel = new JPanel ();
    controlPanel.setLayout (new FlowLayout ());
    mainFrame.add (headerLabel);
    mainFrame.add (controlPanel);
    mainFrame.add (statusLabel);
    mainFrame.setVisible (true);
}

private void showComboboxDemo () {
    headerLabel.setText ("Control in action: JComboBox");
    final DefaultComboBoxModel cityName = new
DefaultComboBoxModel ();
    cityName.addElement ("Ha Noi");
    cityName.addElement ("TP. HCM");
    cityName.addElement ("Da Nang");
    cityName.addElement ("Hai Phong");
    final JComboBox fruitCombo = new JComboBox (cityName);
    fruitCombo.setSelectedIndex (0);
    JScrollPane fruitListScrollPane = new
```



```
JScrollPane(fruitCombo);  
JButton showButton = new JButton("Show");  
showButton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        String data = "";  
        if (fruitCombo.getSelectedIndex() != -1) {  
            data = "City Selected: " +  
fruitCombo.getItemAt(  
                fruitCombo.getSelectedIndex());  
        }  
        statusLabel.setText(data);  
    }  
});  
controlPanel.add(fruitListScrollPane);  
controlPanel.add(showButton);  
mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.11 Lớp JTextField trong Java Swing

Lớp **JTextField** trong Java Swing là một thành phần cho phép sửa đổi một dòng text đơn. Dưới đây là cú pháp khai báo của lớp `javax.swing.JTextField`:

```
public class JTextField extends JtextComponent  
    implements SwingConstants
```

5.2.11.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.text.JTextComponent
- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

Lớp `JTextField` có trường **static String notifyAction**. Trường này là tên của action để gửi thông báo rằng các nội dung của trường này đã được chấp nhận.

5.2.11.2 Các constructor của lớp `JTextField` trong Java Swing

`JTextField()`: Xây dựng một `TextField` mới.

`JTextField(Document doc, String text, int columns)`: Xây dựng một `JTextField` mới mà sử dụng mô hình lưu trữ text đã cho và số cột đã cho.

`JTextField(int columns)`: Xây dựng một `TextField` mới và trống với số cột đã cho.

`JTextField(String text)`: Xây dựng một `TextField` mới được khởi tạo với text đã cho.

`JTextField(String text, int columns)`: Xây dựng một `TextField` mới được khởi tạo với text và các cột đã cho.

5.2.11.3 Các phương thức được sử dụng phổ biến của lớp `JTextField` trong Java Swing

STT	Phương thức & Mô tả
1	<code>void setActionCommand(String command)</code> Thiết lập chuỗi lệnh được sử dụng cho action event
2	<code>void setColumns(int columns)</code> Thiết lập số cột trong <code>TextField</code> này, và sau đó làm mất hiệu lực layout đó
3	<code>void setDocument(Document doc)</code> Liên kết editor với một tài liệu text
4	<code>void setFont(Font f)</code> Thiết lập font hiện tại
5	<code>void setHorizontalAlignment(int alignment)</code> Thiết lập căn chỉnh ngang cho text

6	void setScrollOffset(int scrollOffset) Thiết lập scroll offset, giá trị pixel
7	protected void actionPerformed(ActionEvent action, String propertyName) Cập nhật trạng thái của textfield trong phản hồi các thay đổi của thuộc tính trong action liên kết với
8	void addActionListener(ActionListener l) Thêm action listener đã cho để nhận các action event từ textfield này
9	protected void configurePropertiesFromAction(ActionEvent a) Thiết lập các thuộc tính của textfield này để kết nối chúng trong Action đã cho
10	protected PropertyChangeListener createActionPropertyChangeListener(ActionEvent a) Tạo và trả về PropertyChangeListener mà chịu trách nhiệm nghe các thay đổi từ Action đã cho và cập nhật các thuộc tính thích hợp
11	protected Document createDefaultModel() Tạo trình triển khai mặc định của model để được sử dụng tại sự xây dựng nếu không được cung cấp tường minh
12	Action[] getActions() Gọi danh sách lệnh cho trình soạn thảo Editor
13	void postActionEvent() Xử lý action event xảy ra trên textfield này bằng cách gửi chúng tới bất cứ đối tượng ActionListener đã được đăng ký nào
14	void removeActionListener(ActionListener l) Xóa action listener đã cho để nó không bao giờ nhận action event từ textfield này nữa
15	void scrollRectToVisible(Rectangle r) Cuốn trường này sang trái hoặc phải
16	void setAction(ActionEvent a) Thiết lập Action cho ActionEvent source

5.2.11.4 Chương trình ví dụ lớp JTextField

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.JTextField;

public class JTextFieldExam {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JTextFieldExam() {
        prepareGUI();
    }

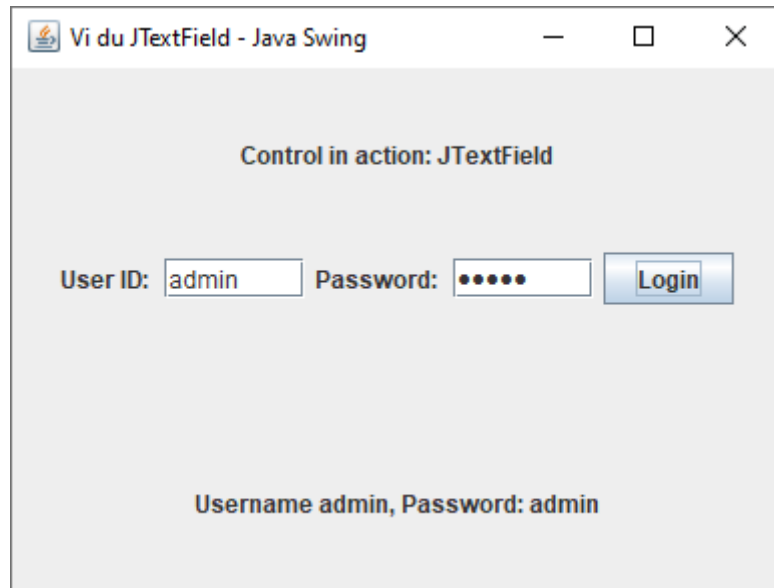
    public static void main(String[] args) {
        JTextFieldExam swingDemo = new JTextFieldExam();
        swingDemo.showTextFieldDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JTextField - Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
    }
}
```

```
mainFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
});
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setSize(350, 100);
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
}

private void showTextFieldDemo() {
    headerLabel.setText("Control in action: JTextField");
    JLabel nameLabel = new JLabel("User ID: ", JLabel.RIGHT);
    JLabel passwordLabel = new JLabel("Password: ",
JLabel.CENTER);
    final JTextField userText = new JTextField(6);
    final JPasswordField passwordText = new JPasswordField(6);
    JButton loginButton = new JButton("Login");
    loginButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String data = "Username " + userText.getText();
            data += ", Password: " + new
String(passwordText.getPassword());
            statusLabel.setText(data);
        }
    });
    controlPanel.add(nameLabel);
    controlPanel.add(userText);
    controlPanel.add(passwordLabel);
    controlPanel.add(passwordText);
    controlPanel.add(loginButton);
    mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.12 Lớp JTextArea trong Java Swing

Lớp **JTextArea** trong Java Swing được sử dụng để tạo một khu vực dành cho text. Nó là một khu vực gồm nhiều dòng và chỉ hiển thị thuần text. Dưới đây là cú pháp khai báo của lớp `javax.swing.JTextArea`:

```
public class JTextArea extends JTextComponent
```

5.2.12.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `javax.swing.text.JTextComponent`
- `javax.swing.JComponent`
- `java.awt.Container`
- `java.awt.Component`
- `java.lang.Object`

5.2.12.2 Các constructor của lớp JTextArea trong Java Swing

`JTextArea()`: Tạo một TextArea mới.

`JTextArea(String s)`: Tạo một TextArea mới với text đã cho.

`JTextArea(int row, int column)`: Tạo một TextArea trống, mới với số hàng và cột đã cho.

`JTextArea(String s, int row, int column)`: Tạo một TextArea mới với text, số hàng và cột đã cho.

5.2.12.3 Các phương thức được sử dụng phổ biến của lớp JTextArea trong Java Swing

1. **public void setRows(int rows):** Được sử dụng để thiết lập số hàng đã cho.
 2. **public void setColumns(int cols):** Được sử dụng để thiết lập số cột đã cho.
 3. **public void setFont(Font f):** Được sử dụng để thiết lập font đã cho.
 4. **public void insert(String s, int position):** Được sử dụng để chèn text vào vị trí đã cho.
 5. **public void append(String s):** Được sử dụng để phụ thêm text đã cho vào cuối tài liệu.
-

5.2.12.4 Chương trình ví dụ đơn giản đầu tiên về lớp JTextArea

```
package vn.plpsoft.swing;

import java.awt.Color;

import javax.swing.JFrame;
import javax.swing.JTextArea;

public class JTextAreaExam1 {
    JTextArea area;
    JFrame f;

    JTextAreaExam1() {
        f = new JFrame("Vi du JTextAreaExam - Java Swing");

        area = new JTextArea(300, 300);
        area.setBounds(10, 30, 300, 300);

        area.setBackground(Color.gray);
        area.setForeground(Color.white);

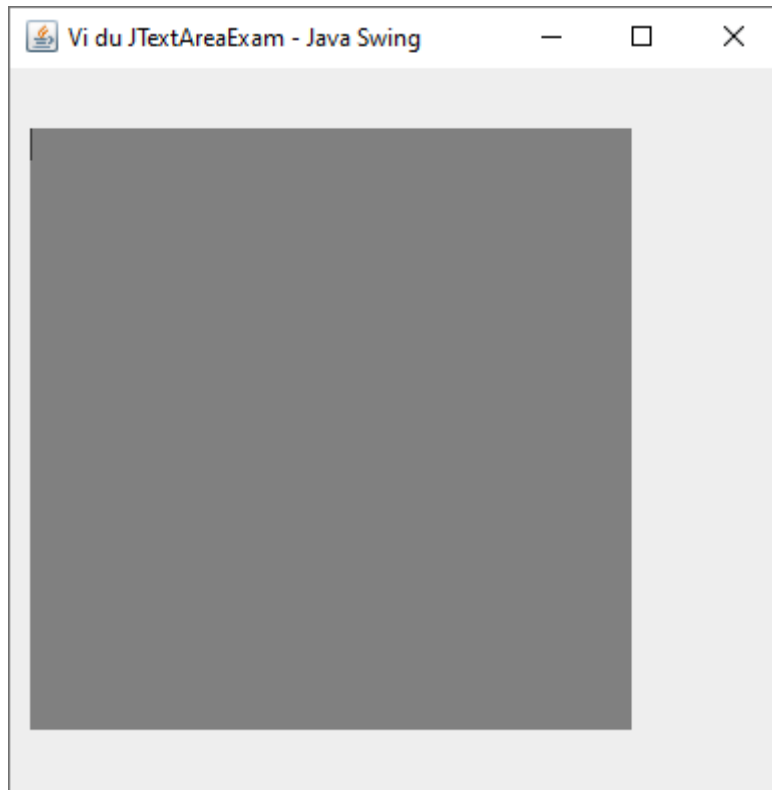
        f.add(area);

        f.setSize(400, 400);
        f.setLayout(null);
        f.setVisible(true);
    }

    public static void main(String[] args) {
        new JTextAreaExam1();
    }
}
```

```
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.12.5 Chương trình ví dụ khác về lớp JTextArea

```
package vn.plpsoft.swing;  
  
import java.awt.FlowLayout;  
import java.awt.GridLayout;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
  
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
import javax.swing.JScrollPane;  
import javax.swing.JTextArea;  
  
public class JTextAreaExam2 {
```



```
private JFrame mainFrame;
private JLabel headerLabel;
private JLabel statusLabel;
private JPanel controlPanel;

public JTextAreaExam2 () {
    prepareGUI ();
}

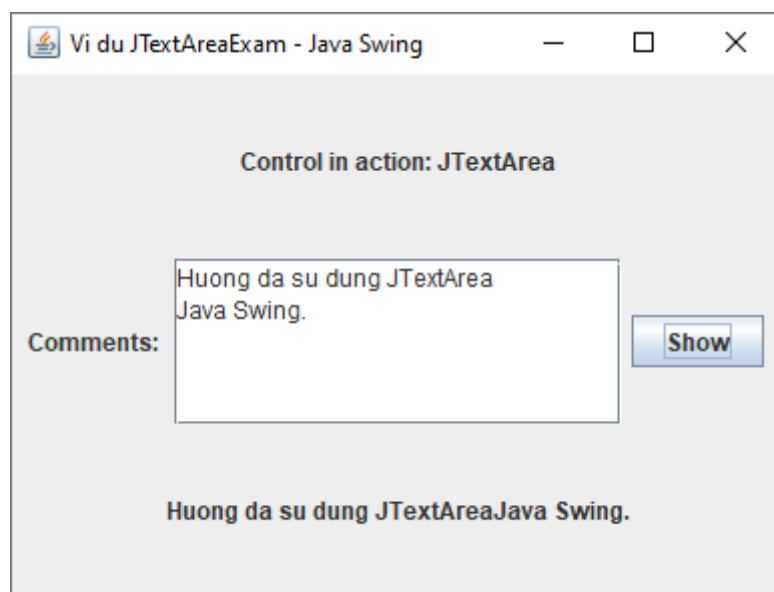
public static void main(String[] args) {
    JTextAreaExam2 swingDemo = new JTextAreaExam2 ();
    swingDemo.showTextAreaDemo ();
}

private void prepareGUI () {
    mainFrame = new JFrame("Vi du JTextAreaExam - Java Swing");
    mainFrame.setSize(400, 300);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter () {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showTextAreaDemo () {
    headerLabel.setText("Control in action: JTextArea");
    JLabel commentlabel = new JLabel("Comments: ",
JLabel.RIGHT);
```

```
final JTextArea commentTextArea = new JTextArea(  
    "Huong da su dung JTextArea\n" + "Java Swing.", 5, 20);  
JScrollPane scrollPane = new JScrollPane(commentTextArea);  
JButton showButton = new JButton("Show");  
showButton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        statusLabel.setText(commentTextArea.getText());  
    }  
});  
controlPanel.add(commentLabel);  
controlPanel.add(scrollPane);  
controlPanel.add(showButton);  
mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.13 Lớp ImageIcon trong Java Swing

Lớp **ImageIcon** trong Java Swing là một trình triển khai của **Icon** Interface, để vẽ các Icon từ các **Image**. Dưới đây là cú pháp khai báo của lớp `javax.swing.ImageIcon`:

```
public class ImageIcon extends Object  
    implements Icon, Serializable, Accessible
```

5.2.13.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.lang.Object`

5.2.13.2 Lớp `Imagelcon` có các trường sau:

- **protected static Component component**
- **protected static MediaTracker tracker**

5.2.13.3 Các constructor của lớp `Imagelcon` trong lớp `Java Swing`

`Imagelcon()`: Tạo một image icon chưa được khởi tạo.

`Imagelcon(byte[] imageData)`: Tạo một `Imagelcon` từ một mảng byte mà đã được đọc từ một image file chứa một định dạng hình ảnh được hỗ trợ, như GIF, JPEG.

`Imagelcon(Image image)`: Tạo một `Imagelcon` từ một đối tượng `Image`.

`Imagelcon(String filename)`: Tạo `Imagelcon` từ file đã cho.

`Imagelcon(URL location)`: Tạo `Imagelcon` từ URL đã cho.

5.2.13.4 Các phương thức của lớp `Imagelcon` trong `Java Swing`

STT	Phương thức & Mô tả
1	<code>Image getImage()</code> Trả về hình ảnh của icon này
2	<code>int getImageLoadStatus()</code> Trả về trạng thái của hoạt động tải hình ảnh này
3	<code>ImageObserver getImageObserver()</code> Trả về trình quan sát observer cho hình ảnh này
4	<code>protected void loadImage(Image image)</code> Tải hình ảnh, trả về chỉ khi hình ảnh được tải
5	<code>void paintIcon(Component c, Graphics g, int x, int y)</code> Sơn màu icon
6	<code>void setDescription(String description)</code> Thiết lập sự miêu tả của hình ảnh

7	void setImage(Image image) Thiết lập hình ảnh được hiển thị bởi icon này
8	String toString() Trả về một biểu diễn chuỗi của hình ảnh

5.2.13.5 Ví dụ ImageIcon trong JLabel

Ví dụ thêm ảnh vào một label, trong ví dụ này tôi cài đặt ảnh có đường dẫn D:\usr\lock.png

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class ImageIconExam {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

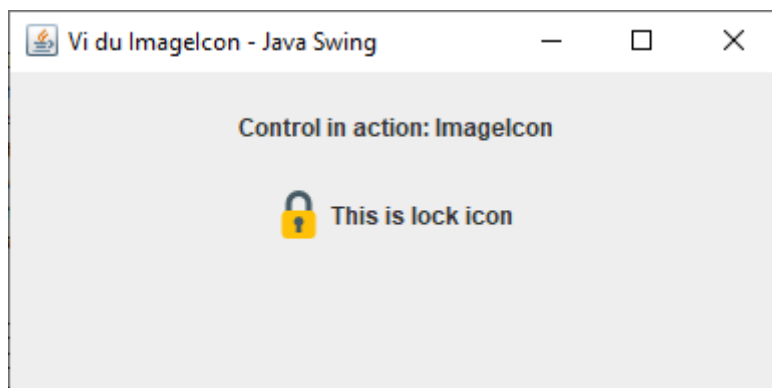
    public ImageIconExam() {
        prepareGUI();
    }

    public static void main(String[] args) {
        ImageIconExam swingControlDemo = new ImageIconExam();
        swingControlDemo.showImageIconDemo();
    }
}
```

```
private void prepareGUI() {
    mainFrame = new JFrame("Vi du ImageIcon - Java Swing");
    mainFrame.setSize(400, 200);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(300, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showImageIconDemo() {
    headerLabel.setText("Control in action: ImageIcon");
    ImageIcon icon = new ImageIcon("D:\\usr\\lock.png", "Lock");
    JLabel commentlabel = new JLabel("This is lock icon", icon,
JLabel.CENTER);
    controlPanel.add(commentlabel);
    mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.13.6 Ví dụ vẽ icon

```
package vn.plpsoft.swing;

import java.awt.Container;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.Graphics;

import javax.swing.GroupLayout;
import javax.swing.ImageIcon;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;

class DrawingPanel extends JPanel {
    private ImageIcon icon;

    public DrawingPanel() {

        loadImage();
        initPanel();
    }

    private void loadImage() {

        icon = new ImageIcon("book-80.png");
    }

    private void initPanel() {

        int w = icon.getIconWidth();
        int h = icon.getIconHeight();
        setPreferredSize(new Dimension(w, h));
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
    }
}
```

```
        icon.paintIcon(this, g, 0, 0);
    }
}

public class ImageIconExam1 extends JFrame {

    public ImageIconExam1() {
        initUI();
    }

    private void initUI() {
        DrawingPanel dpnl = new DrawingPanel();

        createLayout(dpnl);

        setTitle("Image");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }

    private void createLayout(JComponent... arg) {

        Container pane = getContentPane();
        GroupLayout gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setHorizontalGroup(gl.createSequentialGroup().addComponent(
            arg[0]));

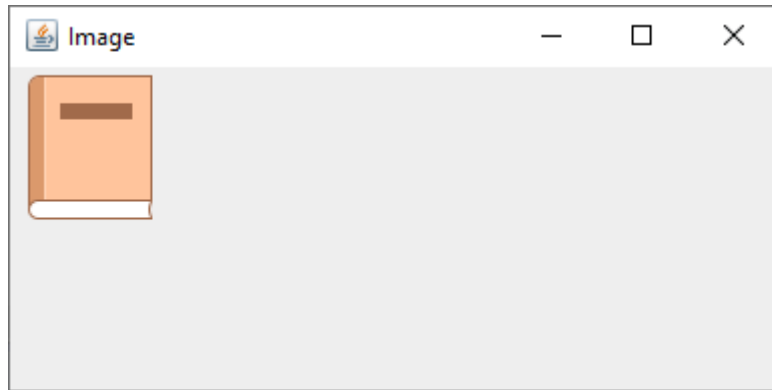
        gl.setVerticalGroup(gl.createParallelGroup().addComponent(a
            rg[0]));

        pack();
    }

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            JFrame f = new ImageIconExam1();
```

```
        f.setSize(400, 200);  
        f.setVisible(true);  
    });  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.13.7 Ví dụ thay đổi kích thước Icon

1. Ví dụ vẽ icon

```
package vn.plpsoft.swing;  
  
import java.awt.Container;  
import java.awt.EventQueue;  
import java.awt.Image;  
  
import javax.swing.GroupLayout;  
import javax.swing.ImageIcon;  
import javax.swing.JComponent;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
  
public class ImageIconExam2 extends JFrame {  
  
    public ImageIconExam2() {  
        initUI();  
    }  
  
    private void initUI() {  
  
        ImageIcon originalIcon = new ImageIcon("Vietnam-Flag-
```



```
icon.png");

    JLabel originalLabel = new JLabel(originalIcon);

    int width = originalIcon.getIconWidth() / 2;
    int height = originalIcon.getIconHeight() / 2;

    Image scaled = scaleImage(originalIcon.getImage(), width, height);

    ImageIcon scaledIcon = new ImageIcon(scaled);

    JLabel newLabel = new JLabel(scaledIcon);

    createLayout(originalLabel, newLabel);

    setTitle("Thay doi kích thước icon");
    setLocationRelativeTo(null);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
}

private Image scaleImage(Image image, int w, int h) {
    Image scaled = image.getScaledInstance(w, h,
    Image.SCALE_SMOOTH);
    return scaled;
}

private void createLayout(JComponent... arg) {
    Container pane = getContentPane();
    GroupLayout gl = new GroupLayout(pane);
    pane.setLayout(gl);

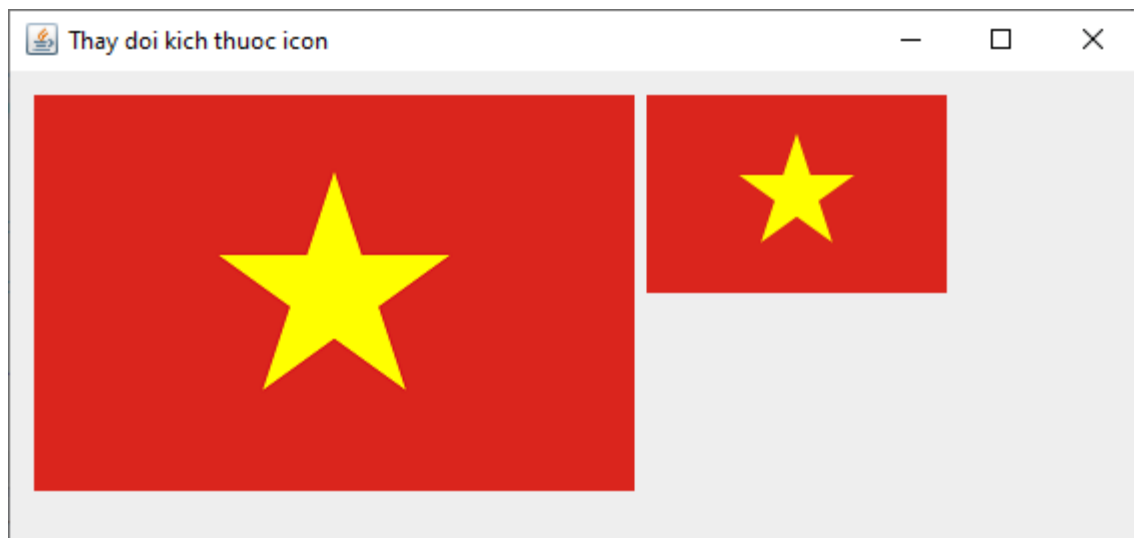
    gl.setAutoCreateContainerGaps(true);
    gl.setAutoCreateGaps(true);

    gl.setHorizontalGroup(gl.createSequentialGroup()
        .addComponent(arg[0]).addComponent(arg[1]));

    gl.setVerticalGroup(gl.createParallelGroup()
        .addComponent(arg[0]).addComponent(arg[1]));
```

```
        pack();  
    }  
  
    public static void main(String[] args) {  
        EventQueue.invokeLater(() -> {  
            ImageIconExam2 f = new ImageIconExam2();  
            f.setSize(500, 400);  
            f.setVisible(true);  
        });  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



2. Ví dụ ImageIcon trong JButton

```
package vn.plpsoft.swing;  
  
import java.awt.Container;  
import java.awt.EventQueue;  
  
import javax.swing.GroupLayout;  
import javax.swing.ImageIcon;  
import javax.swing.JButton;  
import javax.swing.JComponent;  
import javax.swing.JFrame;  
  
public class ImageIconExam3 extends JFrame {  
    public ImageIconExam3() {
```

```
        initUI();
    }

    private void initUI() {

        ImageIcon quitIcon = new ImageIcon("setup.png");
        ImageIcon saveIcon = new ImageIcon("save.png");
        ImageIcon homeIcon = new ImageIcon("home.png");

        JButton quitBtn = new JButton(quitIcon);
        JButton saveBtn = new JButton(saveIcon);
        JButton homeBtn = new JButton(homeIcon);

        createLayout(quitBtn, saveBtn, homeBtn);

        setTitle("Vi du icon trong JButtons");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    private void createLayout(JComponent... arg) {

        Container pane = getContentPane();
        GroupLayout gl = new GroupLayout(pane);
        pane.setLayout(gl);

        gl.setAutoCreateContainerGaps(true);
        gl.setAutoCreateGaps(true);

        gl.setHorizontalGroup(gl.createSequentialGroup()
            .addComponent(arg[0])
            .addComponent(arg[1])
            .addComponent(arg[2])
        );

        gl.setVerticalGroup(gl.createParallelGroup()
            .addComponent(arg[0])
            .addComponent(arg[1])
            .addComponent(arg[2])
        );
    }
}
```

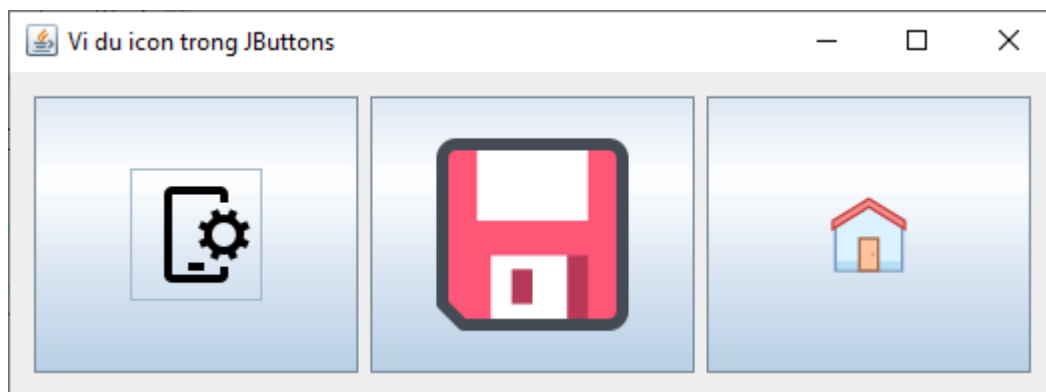
```
);

gl.linkSize(arg[0], arg[1], arg[2]);

pack();
}

public static void main(String[] args) {
    EventQueue.invokeLater(() -> {
        ImageIconExam3 f = new ImageIconExam3();
        f.setVisible(true);
    });
}
```

Chạy chương trình trên cho kết quả như sau:



3. Ví dụ ImageIcon trong tiêu đề JFrame

```
package vn.plpsoft.swing;

import java.awt.EventQueue;

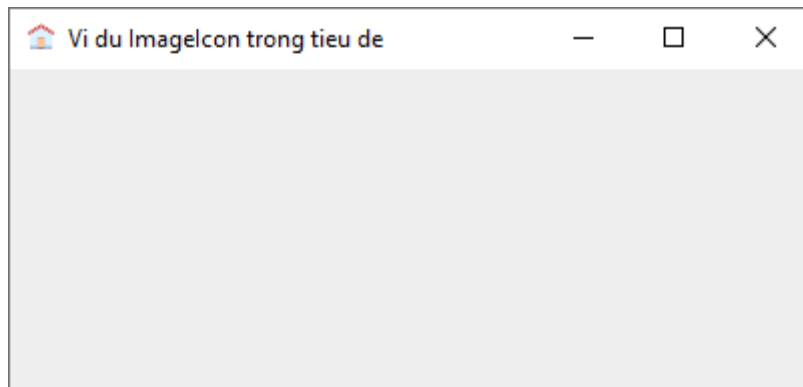
import javax.swing.ImageIcon;
import javax.swing.JFrame;

public class ImageIconExam4 extends JFrame {

    public ImageIconExam4() {
        initUI();
    }
}
```

```
private void initUI() {  
  
    ImageIcon webIcon = new ImageIcon("home.png");  
  
    setIconImage(webIcon.getImage());  
  
    setTitle("Vi du ImageIcon trong tiêu đề");  
    setSize(300, 200);  
    setLocationRelativeTo(null);  
    setDefaultCloseOperation(EXIT_ON_CLOSE);  
}  
  
public static void main(String[] args) {  
  
    EventQueue.invokeLater(() -> {  
        ImageIconExam4 f = new ImageIconExam4();  
        f.setVisible(true);  
    });  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.14 Lớp JScrollbar trong Java Swing

Lớp **JScrollbar** trong Java Swing là một trình triển khai của thanh cuộn scrollbar. Dưới đây là cú pháp khai báo của lớp `javax.swing.JScrollbar`:

```
public class JScrollbar  
    extends JComponent  
        implements Adjustable, Accessible
```

5.2.14.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.lang.Object`

5.2.14.2 Lớp `JScrollbar` bao gồm các trường sau:

- **`protected int blockIncrement`**
- **`protected BoundedRangeModel model`**: Đây là model biểu diễn giá trị nhỏ nhất, lớn nhất, phạm vi và giá trị hiện tại của scrollbar.
- **`protected int orientation`**
- **`protected int unitIncrement`**

5.2.14.3 Các constructor của lớp `JScrollBar` trong Java Swing

`JScrollBar()`: Tạo một thanh cuộn scrollbar theo chiều dọc với các giá trị khởi tạo ban đầu.

`JScrollBar(int orientation)`: Tạo một thanh cuộn scrollbar với hướng orientation đã cho và các giá trị khởi tạo ban đầu.

`JScrollBar(int orientation, int value, int extent, int min, int max)`: Tạo một thanh cuộn scrollbar với hướng orientation, giá trị value, phạm vi extent, minimum và maximum.

5.2.14.4 Các phương thức của lớp `JScrollBar` trong Java Swing

STT	Phương thức & Mô tả
1	<code>void removeAdjustmentListener(AdjustmentListener l)</code> Xóa một <code>AdjustmentEvent</code> listener
2	<code>void setBlockIncrement(int blockIncrement)</code> Thiết lập thuộc tính <code>blockIncrement</code>
3	<code>void setEnabled(boolean x)</code> Kích hoạt thành phần để mà vị trí nút bấm có thể được thay đổi
4	<code>void setMaximum(int maximum)</code> Thiết lập thuộc tính maximum của model
5	<code>void setMinimum(int minimum)</code> Thiết lập thuộc tính minimum của model

6	void setModel(BoundedRangeModel newModel) Thiết lập model mà xử lý 4 thành phần cơ bản của scrollbar là: minimum, maximum, value, extent
7	void setOrientation(int orientation) Thiết lập hướng orientation của scrollbar là VERTICAL hoặc HORIZONTAL
8	void setUI(ScrollBarUI ui) Thiết lập đối tượng L&F mà truyền thành phần này
9	void setUnitIncrement(int unitIncrement) Thiết lập thuộc tính unitIncrement
10	void setValue(int value) Thiết lập giá trị value của scrollbar
11	void setValueIsAdjusting(boolean b) Thiết lập thuộc tính valueIsAdjusting của model
12	void setValues(int newValue, int newExtent, int newMin, int newMax) Thiết lập 4 thuộc tính BoundedRangeModel sau khi ép buộc các tham số tuân theo các ràng buộc thông thường
13	void setVisibleAmount(int extent) Thiết lập thuộc tính extent của model
14	void updateUI() Ghi đè phương thức JComponent.updateUI
15	void addAdjustmentListener(AdjustmentListener l) Thêm một AdjustmentListener
16	protected void fireAdjustmentValueChanged(int id, int type, int value) Thông báo cho listener rằng mô hình của scrollbar đã thay đổi
17	int getBlockIncrement(int direction) Trả về lượng để thay đổi giá trị của scrollbar, đã được cung cấp một yêu cầu (thường là "page")

18	int getMaximum() Giá trị mở rộng lớn nhất của scrollbar
19	Dimension getMaximumSize() Scrollbar là linh động theo trục cuộn của nó và cứng nhắc với trục khác
20	BoundedRangeModel getModel() Trả về data model mà xử lý 4 thuộc tính nền tảng của scrollbar: minimum, maximum, value, extent.

5.2.14.5 Chương trình ví dụ lớp JScrollBar

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.AdjustmentEvent;
import java.awt.event.AdjustmentListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollBar;

public class JScrollBarExam1 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JScrollBarExam1() {
        prepareGUI();
    }

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



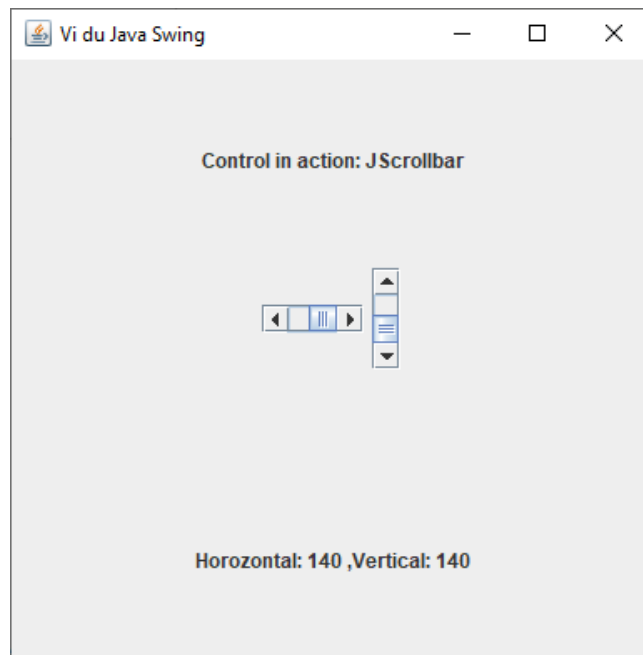
```
JScrollBarExam1 demo = new JScrollBarExam1();
demo.showScrollbarDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame("Vi du Java Swing");
    mainFrame.setSize(400, 400);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showScrollbarDemo() {
    headerLabel.setText("Control in action: JScrollbar");
    final JScrollBar horizontalScroller = new
JScrollBar(JScrollBar.HORIZONTAL);
    final JScrollBar verticalScroller = new JScrollBar();
    verticalScroller.setOrientation(JScrollBar.VERTICAL);
    horizontalScroller.setMaximum(150);
    horizontalScroller.setMinimum(1);
    verticalScroller.setMaximum(150);
    verticalScroller.setMinimum(1);
    horizontalScroller.addAdjustmentListener(new
AdjustmentListener() {
        @Override
        public void adjustmentValueChanged(AdjustmentEvent e) {
            statusLabel.setText(
                "Horozontal: " + horizontalScroller.getValue() +
```

```
        " ,Vertical: " + verticalScroller.getValue());  
    }  
});  
verticalScroller.addAdjustmentListener(new AdjustmentListener()  
{  
    @Override  
    public void adjustmentValueChanged(AdjustmentEvent e) {  
        statusLabel.setText(  
            "Horozontal: " + horizontalScroller.getValue() +  
            " ,Vertical: " + verticalScroller.getValue());  
    }  
});  
controlPanel.add(horizontalScroller);  
controlPanel.add(verticalScroller);  
mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.15 Lớp JOptionPane trong Java Swing

Lớp **JOptionPane** trong Java Swing là một thành phần cung cấp các phương thức chuẩn để gọi một hộp thoại dialog chuẩn cho một giá trị hoặc thông báo người dùng về một cái gì đó. Dưới đây là cú pháp khai báo của lớp `javax.swing.JOptionPane`:

```
public class JOptionPane
```

```
extends JComponent  
implements Accessible
```

5.2.15.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.2.15.2 Các constructor của lớp JOptionPane trong Java Swing

1. **JOptionPane()**: Tạo một JOptionPane với một thông điệp kiểm tra (test message).

2. **JOptionPane(Object message)**: Tạo một instance của JOptionPane để hiển thị một message bởi sử dụng kiểu thông điệp thuần message và các tùy chọn option mặc định được phân phối bởi UI.

3. **JOptionPane(Object, int messageType)**: Tạo một instance của JOptionPane để hiển thị một thông điệp với kiểu thông điệp đã cho và các tùy chọn mặc định.

4. **JOptionPane(Object message, int messageType, int optionType)**: Tạo một instance của JOptionPane để hiển thị một thông điệp với kiểu thông điệp đã cho và các tùy chọn mặc định.

5. **JOptionPane(Object message, int messageType, int optionType, Icon icon)**: Tạo một instance của JOptionPane để hiển thị một thông điệp với kiểu thông điệp, tùy chọn và icon đã cho.

6. **JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options)**: Tạo một instance của JOptionPane để hiển thị một thông điệp với kiểu thông điệp, tùy chọn và icon đã cho.

7. **JOptionPane(Object message, int messageType, int optionType, Icon icon, Object[] options, Object initialValue)**: Tạo một instance của JOptionPane để hiển thị một thông điệp với kiểu thông điệp, tùy chọn và icon đã cho với giá trị tùy chọn được lựa chọn ban đầu đã được xác định.

5.2.15.3 Các phương thức được sử dụng phổ biến của lớp JOptionPane

STT	Phương thức & Mô tả
1	void selectInitialValue() Yêu cầu giá trị khởi tạo ban đầu để được lựa chọn, mà sẽ thiết lập focus tới giá trị đó

2	void setIcon(Icon newIcon) Thiết lập icon để hiển thị
3	void setInitialSelectionValue(Object newValue) Thiết lập giá trị input mà được hiển thị ban đầu như là selected tới người dùng
4	void setInitialValue(Object newInitialValue) Thiết lập giá trị ban đầu để được kích hoạt. Đây là thành phần mà có focus khi pane được hiển thị ban đầu
5	void setInputValue(Object newValue) Thiết lập giá trị input mà là selected hoặc input bởi người dùng
6	void setMessage(Object newMessage) Thiết lập đối tượng message của option pane
7	void setMessageType(int newType) Thiết lập kiểu thông điệp của option pane
8	void setOptions(Object[] newOptions) Thiết lập các tùy chọn mà pane này hiển thị
9	void setOptionType(int newType) Thiết lập các tùy chọn để hiển thị
10	static void setRootFrame(Frame newRootFrame) Thiết lập frame để sử dụng cho các phương thức lớp mà chưa được cung cấp frame nào
11	void setSelectionValues(Object[] newValues) Thiết lập các giá trị selection cho một pane mà cung cấp cho người dùng một danh sách item để lựa chọn từ đó
12	void setUI(OptionPaneUI ui) Thiết lập đối tượng UI mà triển khai L&F cho thành phần này
13	void setValue(Object newValue) Thiết lập giá trị mà người dùng đã lựa chọn

14	void setWantsInput(boolean new Value) Thiết lập thuộc tính wantsInput
15	static int showConfirmDialog(Component parentComponent, Object message) Hiển thị một hộp thoại với các tùy chọn Yes, No và Cancel với title là Select an Option
16	static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType) Hiển thị một hộp thoại, với số tùy chọn được xác định bởi tham số optionType
17	static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType) Hiển thị một hộp thoại, với số tùy chọn được xác định bởi tham số optionType, và tham số messageType xác định icon để hiển thị
18	static int showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon) Hiển thị một hộp thoại với icon đã cho, với số tùy chọn được xác định bởi tham số optionType
19	static String showInputDialog(Component parentComponent, Object message, Object initialSelectionValue) Hiển thị một hộp thoại dạng question-message yêu cầu input từ người dùng được tạo ra từ parentComponent
20	static String showInputDialog(Component parentComponent, Object message, String title, int messageType) Hiển thị một hộp thoại dạng question-message yêu cầu input từ người dùng được tạo ra từ parentComponent với hộp thoại có title và messageType
21	static Object showInternalInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialSelectionValue) Gợi ý người dùng nhập input trong một hộp thoại nội tại, ở đây sự lựa chọn ban đầu, sự lựa chọn có thể có, và tất cả tùy chọn khác có thể được xác định
22	JDialog createDialog(Component parentComponent, String title) Tạo và trả về một JDialog mới mà bao quanh optionpane này được căn chỉnh vào giữa parentComponent trong frame của parentComponent

23	JDialog createDialog(String title) Tạo và trả về một JDialog mới (không phải là cha) với title đã cho
24	JInternalFrame createInternalFrame(Component parentComponent, String title) Tạo và trả về một instance của JInternalFrame

5.2.15.4 Chương trình ví dụ lớp JOptionPane

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

public class JOptionPaneExam1 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JOptionPaneExam1 () {
        prepareGUI ();
    }

    public static void main(String[] args) {
        JOptionPaneExam1 demo = new JOptionPaneExam1 ();
        demo.showDialogDemo ();
    }
}
```

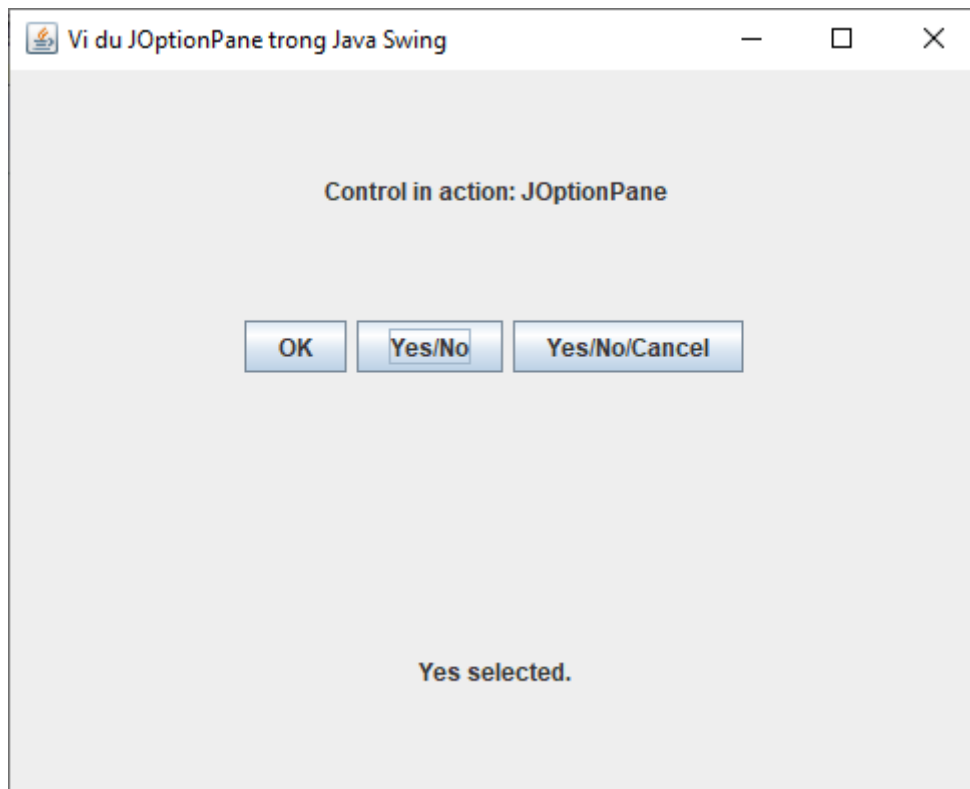
```
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JOptionPane trong Java Swing");
        mainFrame.setSize(500, 400);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showDialogDemo() {
        headerLabel.setText("Control in action: JOptionPane");
        JButton okButton = new JButton("OK");
        JButton javaButton = new JButton("Yes/No");
        JButton cancelButton = new JButton("Yes/No/Cancel");
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(mainFrame,
                    "Welcome to Plpsoft.Vn");
            }
        });
        javaButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int output =
JOptionPane.showConfirmDialog(mainFrame,
                    "Click any button", "Plpsoft.Vn",
                    JOptionPane.YES_NO_OPTION);
            }
        });
    }
}
```

```
        if (output == JOptionPane.YES_OPTION) {
            statusLabel.setText("Yes selected.");
        } else if (output == JOptionPane.NO_OPTION) {
            statusLabel.setText("No selected.");
        }
    }
});
cancelButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int output =
JOptionPane.showConfirmDialog(mainFrame,
                                "Click any button", "Plpsoft.Vn",
                                JOptionPane.YES_NO_CANCEL_OPTION,
                                JOptionPane.INFORMATION_MESSAGE);
        if (output == JOptionPane.YES_OPTION) {
            statusLabel.setText("Yes selected.");
        } else if (output == JOptionPane.NO_OPTION) {
            statusLabel.setText("No selected.");
        } else if (output == JOptionPane.CANCEL_OPTION) {
            statusLabel.setText("Cancel selected.");
        }
    }
});
controlPanel.add(okButton);
controlPanel.add(javaButton);
controlPanel.add(cancelButton);
mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.16 Lớp JFileChooser trong Java Swing

Lớp **JFileChooser** trong Java Swing là một thành phần cung cấp một kỹ thuật đơn giản cho người dùng để lựa chọn một file. Cú pháp khai báo cho lớp `Javax.swing.JFileChooser` là:

```
public class JFileChooser
    extends JComponent
        implements Accessible
```

5.2.16.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `javax.swing.JComponent`
- `java.awt.Container`
- `java.awt.Component`
- `java.lang.Object`

5.2.16.2 Các constructor của lớp JFileChooser trong Java Swing

1. `JFileChooser()`: Xây dựng một `JFileChooser` trở tới thư mục mặc định của người dùng.
2. `JFileChooser(File currentDirectory)`: Xây dựng một `JFileChooser` bởi sử dụng File đã cho như là path.

3. `JFileChooser(File currentDirectory, FileSystemView fsv)`: Xây dựng một `JFileChooser` bởi sử dụng thư mục hiện tại đã cung cấp và `FileSystemView`.
4. `JFileChooser(FileSystemView fsv)`: Xây dựng một `JFileChooser` bởi sử dụng `FileSystemView` đã cho.
5. `JFileChooser(String currentDirectoryPath)`: Xây dựng một `JFileChooser` bởi sử dụng path đã cho.
6. `JFileChooser(String currentDirectoryPath, FileSystemView fsv)`: Xây dựng một `JFileChooser` bởi sử dụng thư mục hiện tại đã cung cấp và `FileSystemView`.

5.2.16.3 Các phương thức của lớp `JFileChooser` trong Java Swing

STT	Phương thức & Mô tả
1	<code>void removeActionListener(ActionListener l)</code> Gỡ bỏ một <code>ActionListener</code> từ file chooser
2	<code>boolean removeChoosableFileFilter(FileFilter f)</code> Xóa một trình lọc filter từ danh sách các trình lọc file có thể lựa chọn của người dùng
3	<code>void rescanCurrentDirectory()</code> Nói cho UI rằng cần quét lại danh sách file của nó từ thư mục hiện tại
4	<code>void resetChoosableFileFilters()</code> Phục hồi danh sách trình lọc file có thể lựa chọn về trạng thái ban đầu của nó
5	<code>void setAcceptAllFileFilterUsed(boolean b)</code> Xác định xem <code>AcceptAll FileFilter</code> có được sử dụng như là một lựa chọn có sẵn không trong danh sách trình lọc có thể chọn
6	<code>void setAccessory(JComponent newAccessory)</code> Thiết lập thành phần phụ thêm
7	<code>void setApproveButtonMnemonic(char mnemonic)</code> Thiết lập mnemonic của nút xác nhận bởi sử dụng một ký tự
8	<code>void setApproveButtonMnemonic(int mnemonic)</code> Thiết lập mnemonic của nút xác nhận bởi sử dụng một keycode dạng số

9	void setApproveButtonText(String approveButtonText) Thiết lập text được sử dụng trong ApproveButton trong FileChooserUI
10	void setApproveButtonToolTipText(String toolTipText) Thiết lập tooltip text được sử dụng trong ApproveButton
11	void setControlButtonsAreShown(boolean b) Thiết lập thuộc tính chỉ rằng có hay không các nút xác nhận và hủy bỏ được hiển thị trong file chooser
12	void setCurrentDirectory(File dir) Thiết lập thư mục hiện tại
13	void setDialogTitle(String dialogTitle) Thiết lập chuỗi trong thanh tiêu đề của cửa sổ JFileChooser
14	void setDialogType(int dialogType) Thiết lập kiểu của dialog này
15	void setDragEnabled(boolean b) Thiết lập thuộc tính dragEnabled property, mà phải là true để kích hoạt trình xử lý hoạt động drag tự động (phần đầu tiên của hoạt động drag và drop) trên thành phần này
16	void setFileFilter(FileFilter filter) Thiết lập trình lọc file hiện tại
17	void setFileHidingEnabled(boolean b) Bật hoặc tắt chế độ ẩn file
18	void setFileSelectionMode(int mode) Thiết lập JFileChooser để cho phép người dùng: chỉ lựa chọn file, chỉ thư mục hoặc cả hai
19	void setFileSystemView(FileSystemView fsv) Thiết lập file system view mà JFileChooser sử dụng để truy cập và tạo các nguồn file system, chẳng hạn như tìm đĩa mềm và lấy danh sách các root drive
20	void setFileView(FileView fileView)

	Thiết lập file view để được sử dụng để lấy thông tin UI, chẳng hạn như icon mà biểu diễn một file hoặc miêu tả kiểu của file
21	void setMultiSelectionEnabled(boolean b) Thiết lập file chooser để cho phép lựa chọn multi-file
22	void setSelectedFile(File file) Thiết lập file được lựa chọn
23	void setSelectedFiles(File[] selectedFiles) Thiết lập danh sách các file được lựa chọn nếu file chooser được thiết lập dạng multi-file
24	protected void setup(FileSystemView view) Thực hiện khởi tạo và thiết lập constructor chung
25	int showDialog(Component parent, String approveButtonText) Popup một hộp thoại file chooser tùy biến với một nút xác nhận tùy biến
26	int showOpenDialog(Component parent) Hiển thị một hộp thoại "Open File"
27	int showSaveDialog(Component parent) Hiển thị một hộp thoại "Save File"
28	void updateUI() Phục hồi thuộc tính UI về giá trị của L&F hiện tại
29	boolean accept(File f) Trả về true nếu file nên được hiển thị
30	void addActionListener(ActionListener l) Thêm một ActionListener tới file chooser
31	void addChoosableFileFilter(FileFilter filter) Thêm một bộ lọc filter tới danh sách bộ lọc các file có thể lựa chọn của người dùng
32	void approveSelection() Được gọi bởi UI khi người dùng nhấn nút xác nhận (được gán nhãn "Open" hoặc

	"Save", theo mặc định)
33	void cancelSelection() Được gọi bởi UI khi người dùng chọn nút Cancel
34	void changeToParentDirectory() Các thay đổi tới thư mục để được thiết lập tới thư mục cha của thư mục hiện tại
35	protected JDialog createDialog(Component parent) Tạo và trả về một JDialog mới đang bao hộp thoại này được căn chỉnh vào giữa frame của thành phần cha
36	void ensureFileIsVisible(File f) Đảm bảo rằng file đã cho là nhìn thấy, không bị ẩn
37	protected void fireActionPerformed(String command) Thông báo cho tất cả Listener mà đã đăng ký nhận thông báo trên kiểu sự kiện (event type) này
38	FileFilter getAcceptAllFileFilter() Trả về trình lọc file là AcceptAll

5.2.16.4 Chương trình ví dụ lớp JFileChooser

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class JFileChooserExam1 {
```

```
private JFrame mainFrame;
private JLabel headerLabel;
private JLabel statusLabel;
private JPanel controlPanel;

public JFileChooserExam1() {
    prepareGUI();
}

public static void main(String[] args) {
    JFileChooserExam1 demo = new JFileChooserExam1();
    demo.showFileChooserDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame("Vi du JFileChooser trong Java Swing");
    mainFrame.setSize(500, 250);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showFileChooserDemo() {
    headerLabel.setText("Control in action: JFileChooser");
    final JFileChooser fileDialog = new JFileChooser();
    JButton showFileDialogButton = new JButton("Open File");
```

```
showFileDialogButton.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        int returnVal =
fileDialog.showOpenDialog(mainFrame);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            java.io.File file =
fileDialog.getSelectedFile();
            statusLabel.setText("File Selected :"+
file.getName());
        } else {
            statusLabel.setText("Open command cancelled by user.");
        }
    }
});
controlPanel.add(showFileDialogButton);
mainFrame.setVisible(true);
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.17 Lớp JProgressBar trong Java Swing

Lớp **JProgressBar** trong Java Swing là một thành phần hiển thị tiến trình của tác vụ. Cú pháp khai báo của lớp `javax.swing.JProgressBar` là:

```
public class JProgressBar
    extends JComponent
        implements SwingConstants, Accessible
```

5.2.17.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.2.17.2 Các constructor được sử dụng phổ biến của lớp JProgressBar

JProgressBar(): được sử dụng để tạo một thanh tiến trình ngang nhưng không có chuỗi text nào.

JProgressBar(int min, int max): được sử dụng để tạo một thanh tiến trình ngang với các giá trị minimum và maximum đã cho.

JProgressBar(int orient): được sử dụng để tạo một thanh tiến trình với orientation đã cho, có thể là Vertical hoặc Horizontal bởi sử dụng hằng SwingConstants.VERTICAL và SwingConstants.HORIZONTAL.

JProgressBar(int orient, int min, int max): được sử dụng để tạo một thanh tiến trình với orientation, giá trị minimum và maximum đã cho.

5.2.17.3 Các phương thức được sử dụng phổ biến của lớp JProgressbar

1. public void setStringPainted(boolean b): được sử dụng để xác định xem chuỗi có nên được hiển thị.

2. public void setString(String s): được sử dụng để thiết lập giá trị tới chuỗi tiến trình.

3. public void setOrientation(int orientation): được sử dụng để thiết lập orientation, có thể là vertical hoặc horizontal bởi sử dụng hằng SwingConstants.VERTICAL và SwingConstants.HORIZONTAL ...

4. public void setValue(int value): được sử dụng để thiết lập giá trị hiện tại trên thanh tiến trình.

5.2.17.4 Chương trình ví dụ đơn giản đầu tiên về lớp JProgressBar

```
package vn.plpsoft.swing;

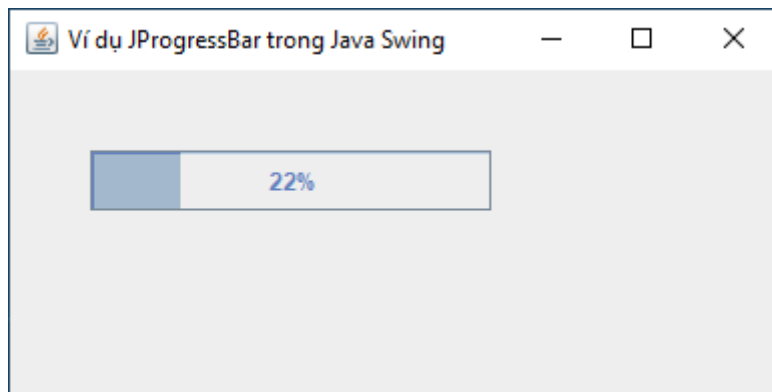
import javax.swing.JFrame;
import javax.swing.JProgressBar;
import javax.swing.WindowConstants;

public class JProgressBarExam1 extends JFrame {
    JProgressBar jb;
    int i = 0, num = 0;
```



```
JProgressBarExam1() {  
    jb = new JProgressBar(0, 2000);  
    jb.setBounds(40, 40, 200, 30);  
  
    jb.setValue(0);  
    jb.setStringPainted(true);  
  
    add(jb);  
    this.setTitle("Ví dụ JProgressBar trong Java Swing");  
    this.setSize(400, 200);  
    this.setLayout(null);  
    this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
}  
  
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) {  
    JProgressBarExam1 m = new JProgressBarExam1();  
    m.setVisible(true);  
    m.iterate();  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.17.5 Chương trình ví dụ khác về lớp JProgressBar

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JProgressBar;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;
import javax.swing.WindowConstants;

public class JProgressBarExam2 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JProgressBarExam2 () {
        prepareGUI ();
    }
```

```
}

public static void main(String[] args) {
    JProgressBarExam2 demo = new JProgressBarExam2();
    demo.showProgressBarDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame("Vi du JProgressBar trong Java Swing");
    mainFrame.setSize(400, 300);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

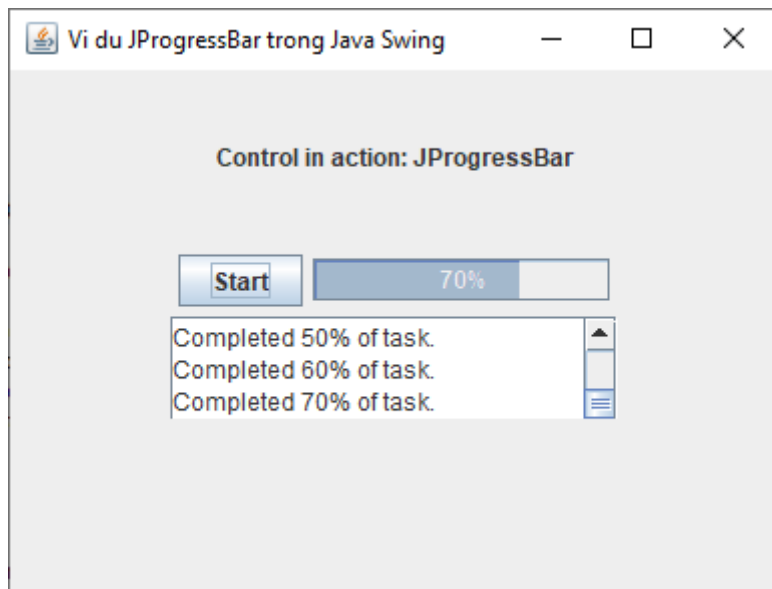
private JProgressBar progressBar;
private Task task;
private JButton startButton;
private JTextArea outputTextArea;

private void showProgressBarDemo() {
    headerLabel.setText("Control in action: JProgressBar");
    progressBar = new JProgressBar(0, 100);
    progressBar.setValue(0);
    progressBar.setStringPainted(true);
    startButton = new JButton("Start");
```

```
outputTextArea = new JTextArea("", 5, 20);
JScrollPane scrollPane = new JScrollPane(outputTextArea);
startButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        task = new Task();
        task.start();
    }
});
controlPanel.add(startButton);
controlPanel.add(progressBar);
controlPanel.add(scrollPane);
mainFrame.setVisible(true);
}

private class Task extends Thread {
    public Task() {
    }

    public void run() {
        for (int i = 0; i <= 100; i += 10) {
            final int progress = i;
            SwingUtilities.invokeLater(new Runnable() {
                public void run() {
                    progressBar.setValue(progress);
                    outputTextArea.setText(
                        outputTextArea.getText() +
                        String.format("Completed %d%% of
task.\n", progress));
                }
            });
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
            }
        }
    }
}
```



5.2.18 Lớp JSlider trong Java Swing

Lớp **JSlider** trong Java Swing được sử dụng để tạo con trượt slider. Bởi sử dụng JSlider, một người dùng có thể lựa chọn một giá trị từ một dãy cụ thể. Cú pháp khai báo của lớp JSlider như sau:

```
public class JSlider
    extends JComponent
        implements SwingConstants, Accessible
```

5.2.18.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.2.18.2 Các constructor được sử dụng phổ biến của lớp JSlider

JSlider(): tạo một slider với giá trị khởi tạo là 50 và dãy giá trị là từ 0 tới 100.

JSlider(int orientation): tạo một slider với orientation đã cho được thiết lập bởi hoặc JSlider.HORIZONTAL hoặc JSlider.VERTICAL với dãy từ 0 tới 100 và giá trị khởi tạo là 50.

JSlider(int min, int max): tạo một thanh slider ngang bởi sử dụng giá trị min và max đã cho.

JSlider(int min, int max, int value): tạo một thanh slider ngang bởi sử dụng giá trị min, max và value đã cho.

JSlider(int orientation, int min, int max, int value): tạo một slider bởi sử dụng orientation, min, max và value đã cho.

5.2.18.3 Các phương thức được sử dụng phổ biến của lớp JSlider

public void setMinorTickSpacing(int n): được sử dụng để thiết lập khoảng cách tick nhỏ nhất cho slider.

public void setMajorTickSpacing(int n): được sử dụng để thiết lập khoảng cách tick lớn nhất cho slider.

public void setPaintTicks(boolean b): được sử dụng để xác định xem tick mark có được sơn màu hay không.

public void setPaintLabels(boolean b): được sử dụng để xác định xem label có được sơn màu hay không.

public void setPaintTracks(boolean b): được sử dụng để xác định xem track có được sơn màu hay không.

5.2.18.4 Chương trình ví dụ đầu tiên về lớp JSlider

```
package vn.plpsoft.swing;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.WindowConstants;

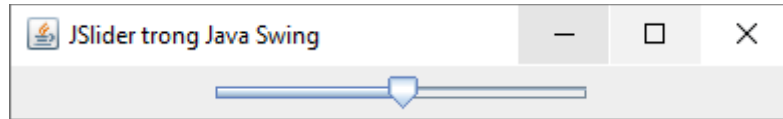
public class JSliderExam1 extends JFrame {

    public JSliderExam1() {
        JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 50, 25);
        JPanel panel = new JPanel();
        panel.add(slider);

        add(panel);
        setTitle("JSlider trong Java Swing");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public static void main(String s[]) {
        JSliderExam1 frame = new JSliderExam1();
        frame.pack();
        frame.setVisible(true);
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.18.5 Ví dụ khác về lớp JSlider mà tô màu các tick

```
package vn.plpsoft.swing;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.WindowConstants;

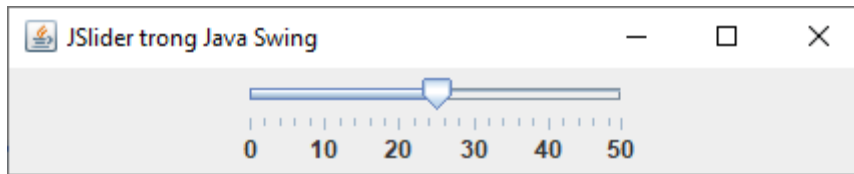
public class JSliderExam2 extends JFrame {
    public JSliderExam2() {
        JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 50, 25);
        slider.setMinorTickSpacing(2);
        slider.setMajorTickSpacing(10);

        slider.setPaintTicks(true);
        slider.setPaintLabels(true);

        JPanel panel = new JPanel();
        panel.add(slider);
        add(panel);
        setTitle("JSlider trong Java Swing");
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public static void main(String s[]) {
        JSliderExam2 frame = new JSliderExam2();
        frame.pack();
        frame.setVisible(true);
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.18.6 Chương trình ví dụ khác về lớp JSlider

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSlider;
import javax.swing.WindowConstants;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class JSliderExam3 {

    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JSliderExam3() {
        prepareGUI();
    }

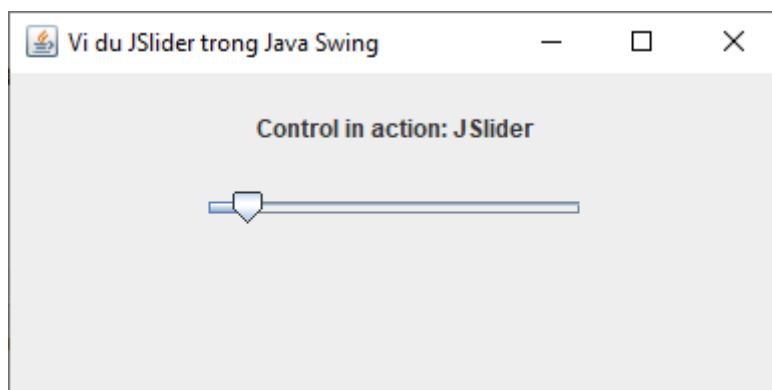
    public static void main(String[] args) {
        JSliderExam3 demo = new JSliderExam3();
        demo.showSliderDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JSlider trong Java Swing");
```



```
mainFrame.setSize(400, 200);
mainFrame.setLayout(new GridLayout(3, 1));
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setSize(350, 100);
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}
private void showSliderDemo() {
    headerLabel.setText("Control in action: JSlider");
    JSlider slider = new JSlider(JSlider.HORIZONTAL, 0, 100, 10);
    slider.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
            statusLabel.setText("Value : " + ((JSlider)
e.getSource()).getValue());
        }
    });
    controlPanel.add(slider);
    mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.2.19 Lớp JSpinner trong Java Swing

Lớp **JSpinner** trong Java Swing là một thành phần cho phép người dùng lựa chọn một số hoặc một giá trị đối tượng từ một dãy đã qua sắp xếp bởi sử dụng một trường đầu vào. Cú pháp khai báo của lớp `javax.swing.JSpinner` là:

```
public class JSpinner
    extends JComponent
        implements Accessible
```

5.2.19.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `javax.swing.JComponent`
- `java.awt.Container`
- `java.awt.Component`
- `java.lang.Object`

5.2.19.2 Các constructor của lớp JSpinner trong Java Swing

JSpinner(): Xây dựng một spinner với một Integer `SpinnerNumberModel` với giá trị khởi tạo 0 và không có giới hạn lớn nhất và nhỏ nhất.

JSpinner(SpinnerModel model): Xây dựng một spinner đầy đủ với một cặp nút `next/previous` và một editor cho `SpinnerModel`.

5.2.19.3 Các phương thức của lớp JSpinner trong Java Swing

STT	Phương thức & Mô tả
1	<code>void removeChangeListener(ChangeListener listener)</code> Xóa một <code>ChangeListener</code> từ spinner này
2	<code>void setEditor(JComponent editor)</code> Thay đổi <code>JComponent</code> mà hiển thị giá trị hiện tại của <code>SpinnerModel</code>
3	<code>void setModel(SpinnerModel model)</code> Thay đổi model mà biểu diễn giá trị của spinner này
4	<code>void setUI(SpinnerUI ui)</code> Thiết lập đối tượng L&F mà truyền spinner này

5	void setValue(Object value) Thay đổi giá trị hiện tại của model, đặt trung của giá trị này là được hiển thị bởi Editor
6	void updateUI() Phục hồi thuộc tính UI về giá trị L&F hiện tại
7	void addChangeListener(ChangeListener listener) Thêm một listener tới list mà được thông báo mỗi khi xuất hiện một thay đổi tới model
8	void commitEdit() Ký thác giá trị đã sửa đổi hiện tại tới SpinnerModel
9	protected JComponent createEditor(SpinnerModel model) Phương thức này được gọi bởi các constructor để tạo JComponent mà hiển thị giá trị hiện tại của dãy
10	protected void fireStateChanged() Gửi một ChangeEvent, có source là JSpinner này, tới mỗi ChangeListener
11	AccessibleContext getAccessibleContext() Lấy AccessibleContext cho JSpinner
12	ChangeListener[] getChangeListeners() Trả về một mảng tất cả ChangeListeners được thêm tới JSpinner này với phương thức addChangeListener()
13	JComponent getEditor() Trả về thành phần mà hiển thị và thay đổi giá trị của model

5.2.19.4 Chương trình ví dụ lớp JSpinner trong Java Swing

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.SpinnerModel;
import javax.swing.SpinnerNumberModel;
import javax.swing.WindowConstants;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class JSpinnerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JSpinnerExam1() {
        prepareGUI();
    }

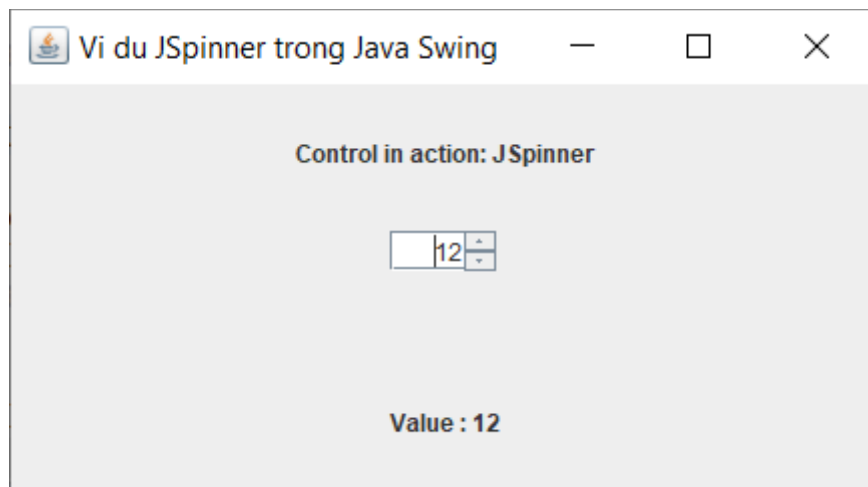
    public static void main(String[] args) {
        JSpinnerExam1 demo = new JSpinnerExam1();
        demo.showSpinnerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JSpinner trong Java Swing");
        mainFrame.setSize(450, 250);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
    }
}
```

```
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showSpinnerDemo() {
    headerLabel.setText("Control in action: JSpinner");
    SpinnerModel spinnerModel = new SpinnerNumberModel(
        10, // initial value
        0, // min
        100, // max
        1); // step
    JSpinner spinner = new JSpinner(spinnerModel);
    spinner.addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
            statusLabel.setText("Value : " +
                ((JSpinner) e.getSource()).getValue());
        }
    });
    controlPanel.add(spinner);
    mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.3 Nhóm lớp chứa (Container)

5.3.1 Lớp JPanel trong Java Swing

JPanel trong Java Swing được sử dụng để tạo ra các container nơi chứa các phần tử khác. Container là thành phần chủ chốt trong các thành phần của SWING GUI. Một Container cung cấp một không gian, là nơi đặt một thành phần. Một Container trong AWT chính là một Component và nó có thêm khả năng để thêm các thành phần khác vào chính nó.

Khi xem xét về Container, bạn cần chú ý các điểm sau:

- Các lớp con của Container được gọi là Container. Một số ví dụ về các lớp con của Container là JPanel, JFrame và JWindow.
- Container chỉ có thể thêm Component vào chính nó
- Một layout mặc định có mặt trong mỗi container. Layout này có thể bị ghi đè bởi sử dụng phương thức `setLayout()`.

Chương này chúng ta sẽ tìm hiểu về **JPanel**. Lớp JPanel là một container chung và gọn nhẹ. Cú pháp khai báo cho lớp **`javax.swing.JPanel`** là:

```
public class JPanel
    extends JComponent
        implements Accessible
```

5.3.1.1 Lớp này kế thừa các phương thức từ các lớp sau

- `javax.swing.JComponent`
 - `java.awt.Container`
 - `java.awt.Component`
 - `java.lang.Object`
-

5.3.1.2 Lớp này bao gồm các constructor sau:

1. **`JPanel()`**: Tạo một JPanel mới với một double buffer và một Flow Layout.
 2. **`JPanel(boolean isDoubleBuffered)`**: Tạo một JPanel mới với Flow Layout và trình đệm đã xác định.
 3. **`JPanel(LayoutManager layout)`**: Tạo một JPanel mới với Layout Manager đã cho
 4. **`JPanel(LayoutManager layout, boolean isDoubleBuffered)`**: Tạo một JPanel mới với Layout Manager đã cho và trình đệm đã xác định.
-

5.3.1.3 Các phương thức của lớp JPanel

1. **AccessibleContext getAccessibleContext():** Lấy AccessibleContext được liên kết với JPanel này.
2. **PanelUI getUI():** Trả về đối tượng L&F mà truyền thành phần này
3. **String getUIClassID():** Trả về một chuỗi xác định tên của lớp L&F mà truyền thành phần này
4. **protected String paramString():** Trả về một biểu diễn chuỗi của JPanel này
5. **void setUI(PanelUI ui):** Thiết lập đối tượng L&F mà truyền thành phần này
6. **void updateUI():** Phục hồi thuộc tính UI về một giá trị Look và Feel hiện tại.

5.3.1.4 Chương trình ví dụ về lớp JPanel trong Java Swing

SwingContainerDemo.java

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class JPanelExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;

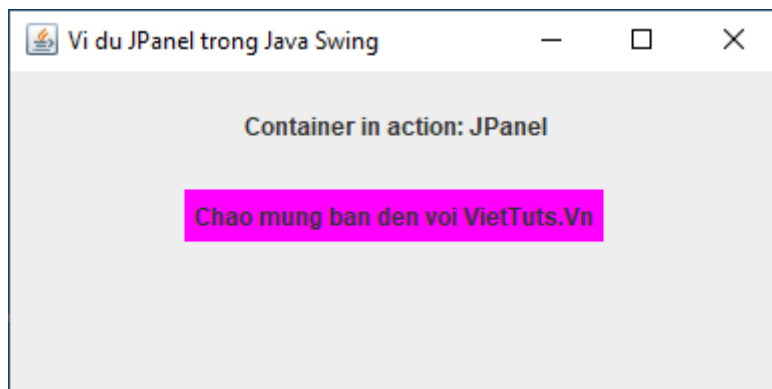
    public JPanelExam1 () {
        prepareGUI ();
    }

    public static void main(String[] args) {
        JPanelExam1 demo = new JPanelExam1 ();
        demo.showJPanelDemo ();
    }
}
```

```
private void prepareGUI() {
    mainFrame = new JFrame("Vi du JPanel trong Java Swing");
    mainFrame.setSize(400, 200);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    msglabel = new JLabel("Chao mung ban den voi Plpsoft.Vn",
JLabel.CENTER);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_
ON_CLOSE);
}

private void showJPanelDemo() {
    headerLabel.setText("Container in action: JPanel");
    JPanel panel = new JPanel();
    panel.setBackground(Color.magenta);
    panel.setLayout(new FlowLayout());
    panel.add(msglabel);
    controlPanel.add(panel);
    mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.2 Lớp JFrame trong Java Swing

Container là thành phần chủ chốt trong các thành phần của SWING GUI. Một Container cung cấp một không gian, là nơi đặt một thành phần. Một Container trong AWT chính là một Component và nó có thêm khả năng để thêm các thành phần khác vào chính nó.

Khi xem xét về Container, bạn cần chú ý các điểm sau:

- Các lớp con của Container được gọi là Container. Một số ví dụ về các lớp con của Container là JPanel, JFrame và JWindow.
- Container chỉ có thể thêm Component vào chính nó.
- Một layout mặc định có mặt trong mỗi container. Layout này có thể bị ghi đè bởi sử dụng phương thức `setLayout()`.

Chương này chúng ta sẽ tìm hiểu về **JFrame**. Lớp JFrame là một lớp kế thừa từ `java.awt.Frame` mà bổ sung các hỗ trợ cho cấu trúc thành phần JFC/Swing. Cú pháp khai báo cho lớp **`javax.swing.JFrame`** là:

```
public class JFrame
    extends Frame
    implements WindowConstants, Accessible, RootPaneContainer
```

5.3.2.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.awt.Frame`
- `java.awt.Window`
- `java.awt.Container`
- `java.awt.Component`
- `java.lang.Object`

5.3.2.2 Lớp JFrame này có các constructor sau:

1. **`JFrame()`**: Xây dựng một Frame mới, ban đầu là không nhìn thấy (invisible).
2. **`JFrame(GraphicsConfiguration gc)`**: Tạo một Frame trong `GraphicsConfiguration` đã cho của một thiết bị màn hình và một title trống.
3. **`JFrame(String title)`**: Tạo một Frame mới, ban đầu là không nhìn thấy (invisible) với title đã cho.
4. **`JFrame(String title, GraphicsConfiguration gc)`**: Tạo một Frame với title đã cho và `GraphicsConfiguration` đã cho của một thiết bị màn hình.

5.3.2.3 Chương trình ví dụ về lớp JFrame trong Java Swing

SwingContainerDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class JFrameExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;

    public JFrameExam1 () {
        prepareGUI ();
    }

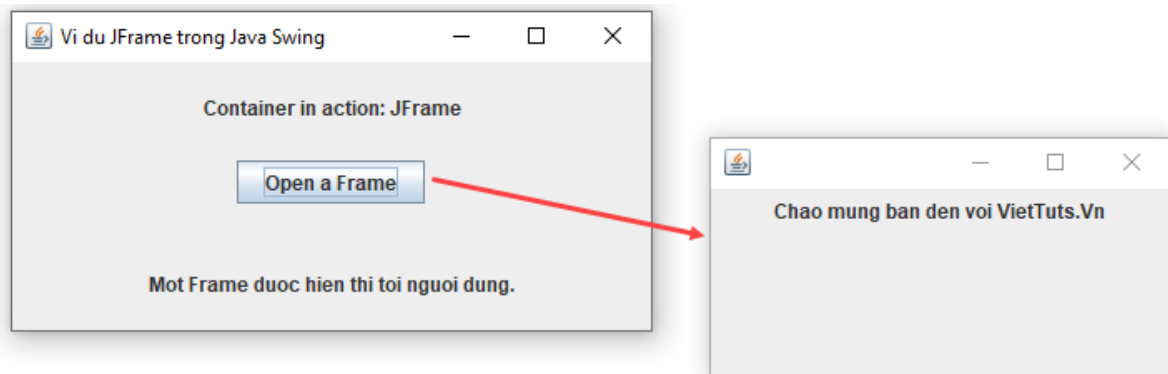
    public static void main(String[] args) {
        JFrameExam1 demo = new JFrameExam1 ();
        demo.showJFrameDemo ();
    }

    private void prepareGUI () {
        mainFrame = new JFrame("Vi du JFrame trong Java Swing");
        mainFrame.setSize(400, 200);
        mainFrame.setLayout(new GridLayout(3, 1));
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_
ON_CLOSE);
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
    }
}
```

```
        statusLabel.setSize(350, 100);
        msglabel = new JLabel("Chao mung ban den voi Plpsoft.Vn",
JLabel.CENTER);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showJFrameDemo() {
        headerLabel.setText("Container in action: JFrame");
        final JFrame frame = new JFrame();
        frame.setSize(300, 100);
        frame.setLayout(new FlowLayout());
        frame.add(msglabel);
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                frame.dispose();
            }
        });
        JButton okButton = new JButton("Open a Frame");
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                statusLabel.setText("Mot Frame duoc hien thi toi
nguai dung.");
                frame.setVisible(true);
            }
        });
        controlPanel.add(okButton);
        mainFrame.setVisible(true);
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.3 Lớp JWindow trong Java Swing

Container là thành phần chủ chốt trong các thành phần của SWING GUI. Một Container cung cấp một không gian, là nơi đặt một thành phần. Một Container trong AWT chính là một Component và nó có thêm khả năng để thêm các thành phần khác vào chính nó.

Khi xem xét về Container, bạn cần chú ý các điểm sau:

- Các lớp con của Container được gọi là Container. Một số ví dụ về các lớp con của Container là JPanel, JFrame và JWindow.
- Container chỉ có thể thêm Component vào chính nó.
- Một layout mặc định có mặt trong mỗi container. Layout này có thể bị ghi đè bởi sử dụng phương thức `setLayout()`.

Chương này chúng ta sẽ tìm hiểu về **JWindow**. Lớp JWindow trong Java là một container mà có thể được hiển thị nhưng không có thanh tiêu đề hoặc các nút quản lý cửa sổ. Cú pháp khai báo cho lớp `javax.swing.JWindow` là:

```
public class JWindow
    extends Window
        implements Accessible, RootPaneContainer
```

5.3.3.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.awt.Window`
- `java.awt.Container`
- `java.awt.Component`
- `java.lang.Object`

5.3.3.2 Lớp JWindow có các constructor sau:

1. **JWindow():** Tạo một window mà không xác định khung sở hữu nó (owner frame).
2. **JWindow(Frame owner):** Tạo một window với owner frame đã cho.

3. JWindow(GraphicsConfiguration gc): Tạo một window với GraphicsConfiguration đã cho của một thiết bị màn hình.

4. JWindow(Window owner): Tạo một window với cửa sổ sở hữu nó đã cho (owner window).

5. JWindow(Window owner, GraphicsConfiguration gc): Tạo một window với cửa sổ sở hữu nó đã cho (owner window) và GraphicsConfiguration đã cho của một thiết bị màn hình.

5.3.3.3 Chương trình ví dụ về lớp JWindow

SwingContainerDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.Graphics;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JWindow;

public class JWindowExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JWindowExam1 () {
        prepareGUI ();
    }

    public static void main (String[] args) {
        JWindowExam1 demo = new JWindowExam1 ();
        demo.showJWindowDemo ();
    }
}
```

```
}

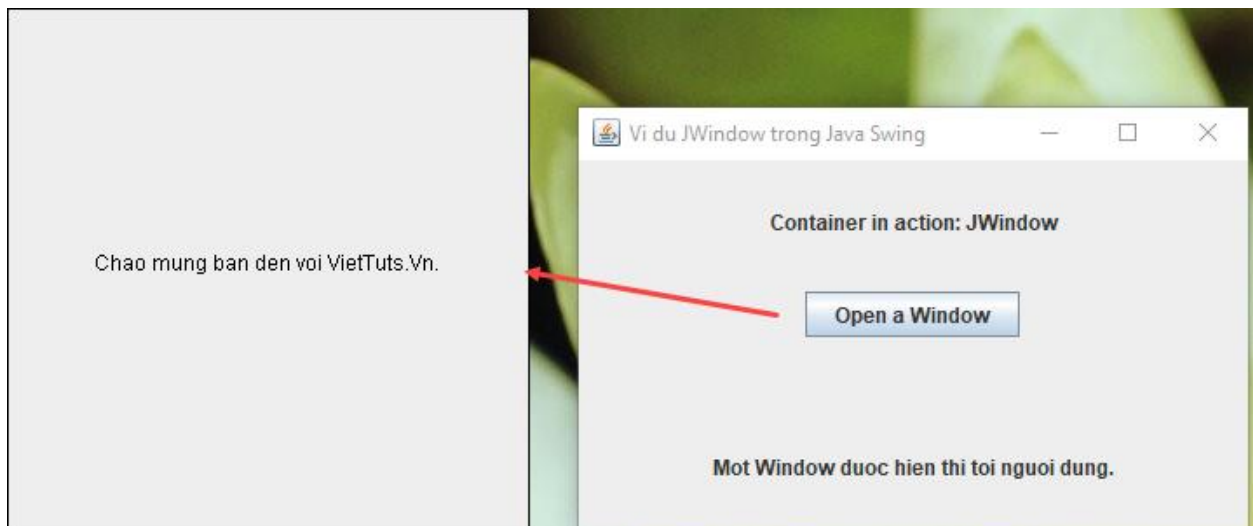
private void prepareGUI() {
    mainFrame = new JFrame("Vi du JWindow trong Java Swing");
    mainFrame.setSize(400, 250);
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            System.exit(0);
        }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showJWindowDemo() {
    headerLabel.setText("Container in action: JWindow");
    final MessageWindow window = new MessageWindow(mainFrame,
        "Chao mung ban den voi Plpsoft.Vn.");
    JButton okButton = new JButton("Open a Window");
    okButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            window.setVisible(true);
        }
    });
    statusLabel.setText("Mot Window duoc hien thi toi nguoi dung.");
    controlPanel.add(okButton);
    mainFrame.setVisible(true);
}

class MessageWindow extends JWindow {
    private String message;
```

```
public MessageWindow(JFrame parent, String message) {  
    super(parent);  
    this.message = message;  
    setSize(300, 300);  
    setLocationRelativeTo(parent);  
}  
  
public void paint(Graphics g) {  
    super.paint(g);  
    g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);  
    g.drawString(message, 50, 150);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.4 Lớp JMenuBar trong Java Swing

Mỗi cửa sổ window có một thanh trình đơn (menu bar) được liên kết với nó. Thanh trình đơn này gồm các lựa chọn có sẵn tới người dùng cuối cùng. Các điều khiển Menu và MenuItem là lớp con của lớp MenuComponent.

Lớp **JMenuBar** cung cấp một trình triển khai của một thanh trình đơn (menu bar). Dưới đây là cú pháp khai báo cho lớp javax.swing.JMenuBar:

```
public class JMenuBar  
    extends JComponent  
        implements Accessible, MenuElement
```

5.3.4.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.3.4.2 Constructor của lớp JMenuBar

JMenuBar(): Tạo một thanh trình đơn mới.

5.3.4.3 Chương trình ví dụ về lớp JMenuBar

SwingMenuDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.WindowConstants;

public class JMenuBarExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
```



```
private JPanel controlPanel;

public JMenuBarExam1() {
    prepareGUI();
}

public static void main(String[] args) {
    JMenuBarExam1 demo = new JMenuBarExam1();
    demo.showMenuDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame("Vi du JMenuBar trong Java Swing");
    mainFrame.setSize(400, 300);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT
_ON_CLOSE);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showMenuDemo() {
    // Tao mot menu bar
    final JMenuBar menuBar = new JMenuBar(); // Tao cac menu
    JMenu fileMenu = new JMenu("File");
    JMenu editMenu = new JMenu("Edit");
    final JMenu aboutMenu = new JMenu("About");
    final JMenu linkMenu = new JMenu("Links");

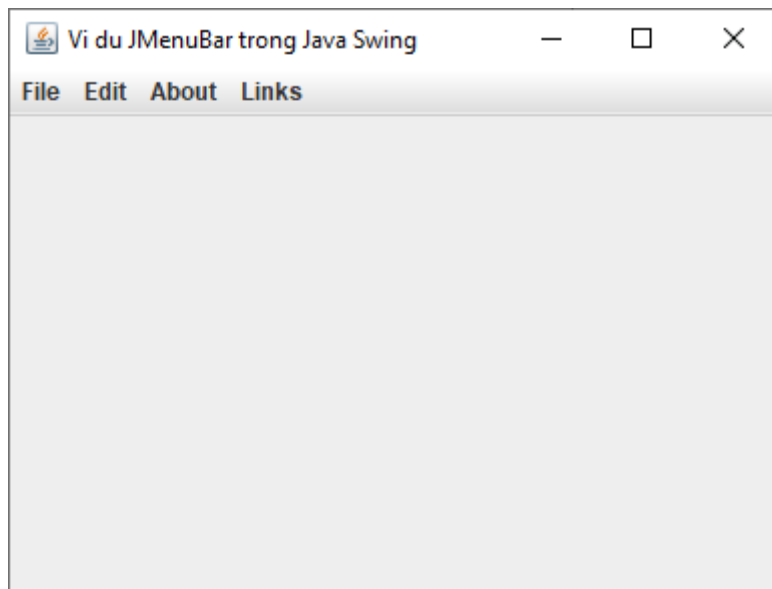
    // tao cac item
    JMenuItem newItem = new JMenuItem("New");
    newItem.setMnemonic(KeyEvent.VK_N);
```

```
newMenuItem.setActionCommand("New");
JMenuItem openMenuItem = new JMenuItem("Open");
openMenuItem.setActionCommand("Open");
JMenuItem saveMenuItem = new JMenuItem("Save");
saveMenuItem.setActionCommand("Save");
JMenuItem exitMenuItem = new JMenuItem("Exit");
exitMenuItem.setActionCommand("Exit");
JMenuItem cutMenuItem = new JMenuItem("Cut");
cutMenuItem.setActionCommand("Cut");
JMenuItem copyMenuItem = new JMenuItem("Copy");
copyMenuItem.setActionCommand("Copy");
JMenuItem pasteMenuItem = new JMenuItem("Paste");
pasteMenuItem.setActionCommand("Paste");
MenuItemListener menuItemListener = new
MenuItemListener();
newMenuItem.addActionListener(menuItemListener);
openMenuItem.addActionListener(menuItemListener);
saveMenuItem.addActionListener(menuItemListener);
exitMenuItem.addActionListener(menuItemListener);
cutMenuItem.addActionListener(menuItemListener);
copyMenuItem.addActionListener(menuItemListener);
pasteMenuItem.addActionListener(menuItemListener);
final JCheckBoxMenuItem showWindowMenu = new
JCheckBoxMenuItem("Show About", true);
showWindowMenu.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        if (showWindowMenu.getState()) {
            menuBar.add(aboutMenu);
        } else {
            menuBar.remove(aboutMenu);
        }
    }
});
final JRadioButtonMenuItem showLinksMenu = new
JRadioButtonMenuItem("Show Links", true);
showLinksMenu.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        if (menuBar.getMenu(3) != null) {
            menuBar.remove(linkMenu);
            mainFrame.repaint();
        }
    }
});
```

```
        } else {
            menuBar.add(linkMenu);
            mainFrame.repaint();
        }
    }
}); // Them item toi cac menu
fileMenu.add(newMenuItem);
fileMenu.add(openMenuItem);
fileMenu.add(saveMenuItem);
fileMenu.addSeparator();
fileMenu.add(showWindowMenu);
fileMenu.addSeparator();
fileMenu.add(showLinksMenu);
fileMenu.addSeparator();
fileMenu.add(exitMenuItem);
editMenu.add(cutMenuItem);
editMenu.add(copyMenuItem);
editMenu.add(pasteMenuItem); // Them menu toi menubar
menuBar.add(fileMenu);
menuBar.add(editMenu);
menuBar.add(aboutMenu);
menuBar.add(linkMenu); // Them menubar toi frame
mainFrame.setJMenuBar(menuBar);
mainFrame.setVisible(true);
}

class MenuItemListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        statusLabel.setText(e.getActionCommand() + "
JMenuItem clicked.");
    }
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.5 Lớp JMenuItem trong Java Swing

Mỗi cửa sổ window có một thanh trình đơn (menu bar) được liên kết với nó. Thanh trình đơn này gồm các lựa chọn có sẵn tới người dùng cuối cùng. Các điều khiển Menu và MenuItem là lớp con của lớp MenuComponent.

Lớp **JMenuItem** trong Java biểu diễn item thực sự trong một menu. Tất cả item trong một menu nên kế thừa từ lớp JMenuItem, hoặc một trong các lớp con của nó. Cú pháp khai báo cho lớp javax.swing.JMenuItem là:

```
public class JMenuItem
    extends AbstractButton
        implements Accessible, MenuElement
```

5.3.5.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JAbstractButton
- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.3.5.2 Lớp JMenuItem gồm các constructor sau:

1. **JMenuItem()**: Tạo một JMenuItem không có text hoặc icon.
2. **JMenuItem(Action a)**: Tạo một JMenuItem có các thuộc tính được nhận từ Action đã cho.
3. **JMenuItem(Icon icon)**: Tạo một JMenuItem với icon đã cho.
4. **JMenuItem(String text)**: Tạo một JMenuItem với text đã cho.

5. JMenuItem(String text, Icon icon): Tạo một JMenuItem với text và icon đã cho.

6. JMenuItem(String text, int mnemonic): Tạo một JMenuItem với text đã cho và mnemonic.

5.3.5.3 Chương trình ví dụ lớp JMenuItem

SwingMenuDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;

import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.WindowConstants;

public class JMenuItemExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JMenuItemExam1 () {
        prepareGUI();
    }

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

```
JMenuItemExam1 demo = new JMenuItemExam1();
demo.showMenuDemo();
}

private void prepareGUI() {
mainFrame = new JFrame("Vi du JMenuItem trong Java Swing");
mainFrame.setSize(400, 300);
mainFrame.setLayout(new GridLayout(3, 1));
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT
_ON_CLOSE);
statusLabel.setSize(350, 100);
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
}

private void showMenuDemo() {
// Tao mot menu bar
final JMenuBar menuBar = new JMenuBar(); // Tao cac menu
JMenu fileMenu = new JMenu("File");
JMenu editMenu = new JMenu("Edit");
final JMenu aboutMenu = new JMenu("About");
final JMenu linkMenu = new JMenu("Links");

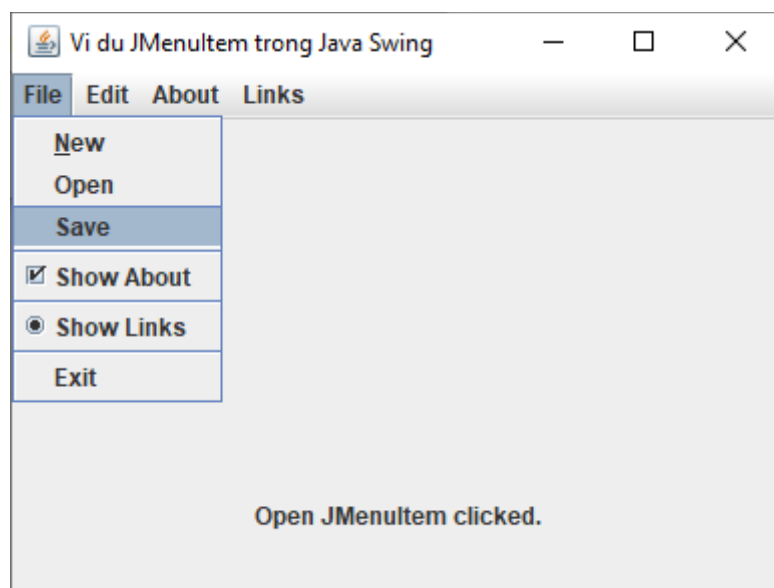
// tao cac item
JMenuItem newItem = new JMenuItem("New");
newItem.setMnemonic(KeyEvent.VK_N);
newItem.setActionCommand("New");
JMenuItem openMenuItem = new JMenuItem("Open");
openMenuItem.setActionCommand("Open");
JMenuItem saveMenuItem = new JMenuItem("Save");
saveMenuItem.setActionCommand("Save");
JMenuItem exitMenuItem = new JMenuItem("Exit");
exitMenuItem.setActionCommand("Exit");
```

```
JMenuItem cutMenuItem = new JMenuItem("Cut");
cutMenuItem.setActionCommand("Cut");
JMenuItem copyMenuItem = new JMenuItem("Copy");
copyMenuItem.setActionCommand("Copy");
JMenuItem pasteMenuItem = new JMenuItem("Paste");
pasteMenuItem.setActionCommand("Paste");
MenuItemListener menuItemListener = new
MenuItemListener();
newMenuItem.addActionListener(menuItemListener);
openMenuItem.addActionListener(menuItemListener);
saveMenuItem.addActionListener(menuItemListener);
exitMenuItem.addActionListener(menuItemListener);
cutMenuItem.addActionListener(menuItemListener);
copyMenuItem.addActionListener(menuItemListener);
pasteMenuItem.addActionListener(menuItemListener);
final JCheckBoxMenuItem showWindowMenu = new
JCheckBoxMenuItem("Show About", true);
showWindowMenu.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        if (showWindowMenu.getState()) {
            menuBar.add(aboutMenu);
        } else {
            menuBar.remove(aboutMenu);
        }
    }
});
final JRadioButtonMenuItem showLinksMenu = new
JRadioButtonMenuItem("Show Links", true);
showLinksMenu.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        if (menuBar.getMenu(3) != null) {
            menuBar.remove(linkMenu);
            mainFrame.repaint();
        } else {
            menuBar.add(linkMenu);
            mainFrame.repaint();
        }
    }
}); // Them item toi cac menu
fileMenu.add(newMenuItem);
```

```
fileMenu.add(openMenuItem);
fileMenu.add(saveMenuItem);
fileMenu.addSeparator();
fileMenu.add(showWindowMenu);
fileMenu.addSeparator();
fileMenu.add(showLinksMenu);
fileMenu.addSeparator();
fileMenu.add(exitMenuItem);
editMenu.add(cutMenuItem);
editMenu.add(copyMenuItem);
editMenu.add(pasteMenuItem); // Them menu toi menubar
menuBar.add(fileMenu);
menuBar.add(editMenu);
menuBar.add(aboutMenu);
menuBar.add(linkMenu); // Them menubar toi frame
mainFrame.setJMenuBar(menuBar);
mainFrame.setVisible(true);
}

class MenuItemListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        statusLabel.setText(e.getActionCommand() + "
JMenuItem clicked.");
    }
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.6 Lớp JMenu trong Java Swing

Mỗi cửa sổ window có một thanh trình đơn (menu bar) được liên kết với nó. Thanh trình đơn này gồm các lựa chọn có sẵn tới người dùng cuối cùng. Các điều khiển Menu và MenuItem là lớp con của lớp MenuComponent.

Lớp **JMenu** trong Java biểu diễn thành phần pull-down menu mà được triển khai từ một thanh trình đơn. Cú pháp khai báo cho lớp javax.swing.JMenu là:

```
public class JMenu
    extends JMenuItem
        implements Accessible, MenuElement
```

5.3.6.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JAbstractButton
- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

Lớp JMenu này có trường **protected JMenu.WinListener popupListener**.

5.3.6.2 Các constructor của lớp JMenu

1. JMenu(): Xây dựng một JMenu mới không có text.

2. JMenu(Action a): Xây dựng một menu có các thuộc tính được nhận từ Action đã cho.

3. JMenu(String s): Xây dựng một JMenu mới với chuỗi s đã cho (như là text của nó).

4. JMenu(String s, boolean b): Xây dựng một JMenu mới với chuỗi s đã cho (như là text của nó) và một giá trị boolean để xác định có hay không một tear-off menu.

5.3.6.3 Chương trình ví dụ lớp JMenu

SwingMenuDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
```

```
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;

import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.WindowConstants;

public class JMenuItemExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JMenuItemExam1 () {
        prepareGUI();
    }

    public static void main(String[] args) {
        JMenuItemExam1 demo = new JMenuItemExam1();
        demo.showMenuDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JMenuItem trong Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT
_ON_CLOSE);
        statusLabel.setSize(350, 100);
```

```
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showMenuDemo() {
        // Tao mot menu bar
        final JMenuBar menuBar = new JMenuBar(); // Tao cac menu
        JMenu fileMenu = new JMenu("File");
        JMenu editMenu = new JMenu("Edit");
        final JMenu aboutMenu = new JMenu("About");
        final JMenu linkMenu = new JMenu("Links");

        // tao cac item
        JMenuItem newItem = new JMenuItem("New");
        newItem.setMnemonic(KeyEvent.VK_N);
        newItem.setActionCommand("New");
        JMenuItem openMenuItem = new JMenuItem("Open");
        openMenuItem.setActionCommand("Open");
        JMenuItem saveMenuItem = new JMenuItem("Save");
        saveMenuItem.setActionCommand("Save");
        JMenuItem exitMenuItem = new JMenuItem("Exit");
        exitMenuItem.setActionCommand("Exit");
        JMenuItem cutMenuItem = new JMenuItem("Cut");
        cutMenuItem.setActionCommand("Cut");
        JMenuItem copyMenuItem = new JMenuItem("Copy");
        copyMenuItem.setActionCommand("Copy");
        JMenuItem pasteMenuItem = new JMenuItem("Paste");
        pasteMenuItem.setActionCommand("Paste");
        MenuItemListener menuItemListener = new
MenuItemListener();

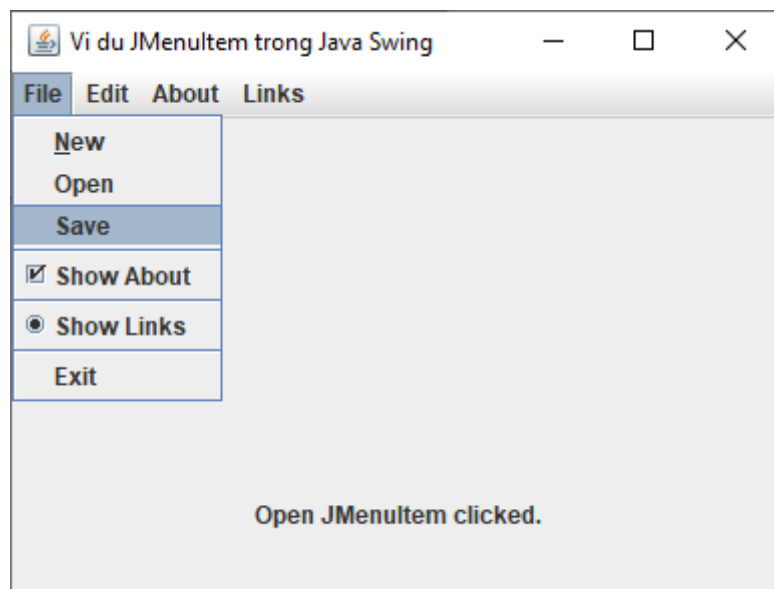
        newItem.addActionListener(menuItemListener);
        openMenuItem.addActionListener(menuItemListener);
        saveMenuItem.addActionListener(menuItemListener);
        exitMenuItem.addActionListener(menuItemListener);
        cutMenuItem.addActionListener(menuItemListener);
```

```
copyMenuItem.addActionListener(menuItemListener);
pasteMenuItem.addActionListener(menuItemListener);
final JCheckBoxMenuItem showWindowMenu = new
JCheckBoxMenuItem("Show About", true);
showWindowMenu.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        if (showWindowMenu.getState()) {
            menuBar.add(aboutMenu);
        } else {
            menuBar.remove(aboutMenu);
        }
    }
});
final JRadioButtonMenuItem showLinksMenu = new
JRadioButtonMenuItem("Show Links", true);
showLinksMenu.addItemListener(new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        if (menuBar.getMenu(3) != null) {
            menuBar.remove(linkMenu);
            mainFrame.repaint();
        } else {
            menuBar.add(linkMenu);
            mainFrame.repaint();
        }
    }
}); // Them item toi cac menu
fileMenu.add(newMenuItem);
fileMenu.add(openMenuItem);
fileMenu.add(saveMenuItem);
fileMenu.addSeparator();
fileMenu.add(showWindowMenu);
fileMenu.addSeparator();
fileMenu.add(showLinksMenu);
fileMenu.addSeparator();
fileMenu.add(exitMenuItem);
editMenu.add(cutMenuItem);
editMenu.add(copyMenuItem);
editMenu.add(pasteMenuItem); // Them menu toi menubar
menuBar.add(fileMenu);
menuBar.add(editMenu);
```

```
        menuBar.add (aboutMenu) ;
        menuBar.add (linkMenu) ; // Them menubar toi frame
        mainFrame.setJMenuBar (menuBar) ;
        mainFrame.setVisible (true) ;
    }

    class MenuItemListener implements ActionListener {
        public void actionPerformed (ActionEvent e) {
            statusLabel.setText (e.getActionCommand () + "
JMenuItem clicked.");
        }
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.7 Lớp JCheckboxMenuItem trong Java Swing

Mỗi cửa sổ window có một thanh trình đơn (menu bar) được liên kết với nó. Thanh trình đơn này gồm các lựa chọn có sẵn tới người dùng cuối cùng. Các điều khiển Menu và MenuItem là lớp con của lớp MenuComponent.

Lớp **JCheckboxMenuItem** trong Java biểu diễn một checkbox mà có thể được bao trong một menu. Việc lựa chọn checkbox đó trong menu làm thay đổi trạng thái của điều khiển từ **on** thành **off** hoặc từ **off** thành **on**. Cú pháp khai báo cho lớp javax.swing.JCheckboxMenuItem là:

```
public class JCheckBoxMenuItem
    extends JMenuItem
        implements SwingConstants, Accessible
```

5.3.7.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JMenuItem
 - javax.swing.JAbstractButton
 - javax.swing.JComponent
 - java.awt.Container
 - java.awt.Component
 - java.lang.Object
-

5.3.7.2 Lớp JCheckboxMenuItem gồm các constructor sau:

1. **JCheckboxMenuItem()**: Tạo một checkbox ban đầu dạng unselected không có text hoặc icon.
 2. **JCheckboxMenuItem(Action a)**: Tạo một menu item có các thuộc tính được lấy từ Action đã cho.
 3. **JCheckboxMenuItem(Icon icon)**: Tạo một checkbox ban đầu dạng unselected với một icon đã cho.
 4. **JCheckboxMenuItem(String text)**: Tạo một checkbox ban đầu dạng unselected với một text đã cho.
 5. **JCheckboxMenuItem(String text, boolean b)**: Tạo một checkbox ban đầu dạng unselected với một text và trạng thái selection đã cho.
 6. **JCheckboxMenuItem(String text, Icon icon)**: Tạo một checkbox ban đầu dạng unselected với một text và icon đã cho.
 7. **JCheckboxMenuItem(String text, Icon icon, boolean b)**: Tạo một checkbox ban đầu dạng unselected với một text, icon và trạng thái selection đã cho.
-

5.3.7.3 Chương trình ví dụ lớp JCheckboxMenuItem

SwingMenuDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;
```

```
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.WindowConstants;

public class JMenuItemExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JMenuItemExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        JMenuItemExam1 demo = new JMenuItemExam1();
        demo.showMenuDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JMenuItem trong Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
    }
}
```

```
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }

    private void showMenuDemo() {
        // Tao mot menu bar
        final JMenuBar menuBar = new JMenuBar(); // Tao cac menu
        JMenu fileMenu = new JMenu("File");
        JMenu editMenu = new JMenu("Edit");
        final JMenu aboutMenu = new JMenu("About");
        final JMenu linkMenu = new JMenu("Links");

        // tao cac item
        JMenuItem newItem = new JMenuItem("New");
        newItem.setMnemonic(KeyEvent.VK_N);
        newItem.setActionCommand("New");
        JMenuItem openMenuItem = new JMenuItem("Open");
        openMenuItem.setActionCommand("Open");
        JMenuItem saveMenuItem = new JMenuItem("Save");
        saveMenuItem.setActionCommand("Save");
        JMenuItem exitMenuItem = new JMenuItem("Exit");
        exitMenuItem.setActionCommand("Exit");
        JMenuItem cutMenuItem = new JMenuItem("Cut");
        cutMenuItem.setActionCommand("Cut");
        JMenuItem copyMenuItem = new JMenuItem("Copy");
        copyMenuItem.setActionCommand("Copy");
        JMenuItem pasteMenuItem = new JMenuItem("Paste");
        pasteMenuItem.setActionCommand("Paste");
        MenuItemListener menuItemListener = new
MenuItemListener();
        newItem.addActionListener(menuItemListener);
        openMenuItem.addActionListener(menuItemListener);
        saveMenuItem.addActionListener(menuItemListener);
        exitMenuItem.addActionListener(menuItemListener);
        cutMenuItem.addActionListener(menuItemListener);
        copyMenuItem.addActionListener(menuItemListener);
        pasteMenuItem.addActionListener(menuItemListener);
        final JCheckBoxMenuItem showWindowMenu = new
JCheckBoxMenuItem("Show About", true);
        showWindowMenu.addItemListener(new ItemListener() {
```

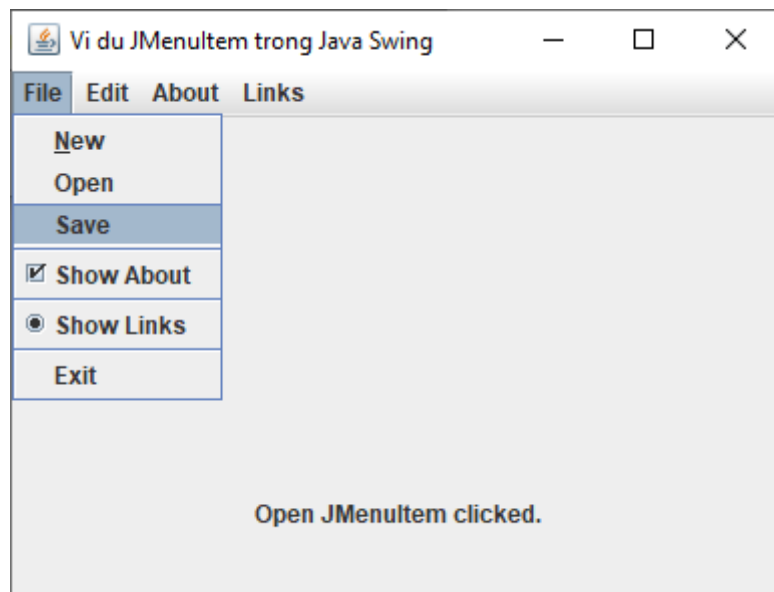


```
        public void itemStateChanged(ItemEvent e) {
            if (showWindowMenu.getState()) {
                menuBar.add(aboutMenu);
            } else {
                menuBar.remove(aboutMenu);
            }
        }
    });

    final JRadioButtonMenuItem showLinksMenu = new
    JRadioButtonMenuItem("Show Links", true);
    showLinksMenu.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            if (menuBar.getMenu(3) != null) {
                menuBar.remove(linkMenu);
                mainFrame.repaint();
            } else {
                menuBar.add(linkMenu);
                mainFrame.repaint();
            }
        }
    }); // Them item toi cac menu
    fileMenu.add(newMenuItem);
    fileMenu.add(openMenuItem);
    fileMenu.add(saveMenuItem);
    fileMenu.addSeparator();
    fileMenu.add(showWindowMenu);
    fileMenu.addSeparator();
    fileMenu.add(showLinksMenu);
    fileMenu.addSeparator();
    fileMenu.add(exitMenuItem);
    editMenu.add(cutMenuItem);
    editMenu.add(copyMenuItem);
    editMenu.add(pasteMenuItem); // Them menu toi menubar
    menuBar.add(fileMenu);
    menuBar.add(editMenu);
    menuBar.add(aboutMenu);
    menuBar.add(linkMenu); // Them menubar toi frame
    mainFrame.setJMenuBar(menuBar);
    mainFrame.setVisible(true);
```

```
}  
  
class JMenuItemListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        statusLabel.setText(e.getActionCommand() + "  
JMenuItem clicked.");  
    }  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.8 Lớp JMenuItem trong Java Swing

Mỗi cửa sổ window có một thanh trình đơn (menu bar) được liên kết với nó. Thanh trình đơn này gồm các lựa chọn có sẵn tới người dùng cuối cùng. Các điều khiển Menu và MenuItem là lớp con của lớp MenuComponent.

Lớp **JRadioButtonMenuItem** trong Java biểu diễn một checkbox mà có thể được bao trong một menu. Việc lựa chọn checkbox đó trong menu làm thay đổi trạng thái của điều khiển từ **on** thành **off** hoặc từ **off** thành **on**. Cú pháp khai báo cho lớp javax.swing.JRadioButtonMenuItem là:

```
public class JRadioButtonMenuItem  
    extends JMenuItem  
        implements Accessible
```

5.3.8.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JMenuItem
- javax.swing.JAbstractButton

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.3.8.2 Lớp JRadioButtonMenuItem gồm các constructor sau:

1. JRadioButtonMenuItem(): Tạo một JRadioButtonMenuItem không có text hoặc icon.

2. JRadioButtonMenuItem(Action a): Tạo một radio button có các thuộc tính được lấy từ Action đã cho.

3. JRadioButtonMenuItem(Icon icon): Tạo một JRadioButtonMenuItem với một icon.

4. JRadioButtonMenuItem(Icon icon, boolean selected): Tạo một JRadioButtonMenuItem với icon và trạng thái lựa chọn selected đã cho, không có text.

5. JRadioButtonMenuItem(String text): Tạo một JRadioButtonMenuItem với text đã cho.

6. JRadioButtonMenuItem(String text, boolean selected): Tạo một JRadioButtonMenuItem với text và trạng thái lựa chọn selected đã cho.

7. JRadioButtonMenuItem(String text, Icon icon): Tạo một JRadioButtonMenuItem với text và icon đã cho.

8. JRadioButtonMenuItem(String text, Icon icon, boolean selected): Tạo một JRadioButtonMenuItem với text, icon và trạng thái lựa chọn selected đã cho.

5.3.8.3 Chương trình ví dụ JRadioButtonMenuItem

JMenuItemExam1.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.KeyEvent;

import javax.swing.JCheckBoxMenuItem;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JRadioButtonMenuItem;
import javax.swing.WindowConstants;

public class JMenuItemExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JMenuItemExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        JMenuItemExam1 demo = new JMenuItemExam1();
        demo.showMenuDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du JMenuItem trong Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT
_ON_CLOSE);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
}
```

```
}

private void showMenuDemo() {
    // Tao mot menu bar
    final JMenuBar menuBar = new JMenuBar(); // Tao cac menu
    JMenu fileMenu = new JMenu("File");
    JMenu editMenu = new JMenu("Edit");
    final JMenu aboutMenu = new JMenu("About");
    final JMenu linkMenu = new JMenu("Links");

    // tao cac item
    JMenuItem newItem = new JMenuItem("New");
    newItem.setMnemonic(KeyEvent.VK_N);
    newItem.setActionCommand("New");
    JMenuItem openMenuItem = new JMenuItem("Open");
    openMenuItem.setActionCommand("Open");
    JMenuItem saveMenuItem = new JMenuItem("Save");
    saveMenuItem.setActionCommand("Save");
    JMenuItem exitMenuItem = new JMenuItem("Exit");
    exitMenuItem.setActionCommand("Exit");
    JMenuItem cutMenuItem = new JMenuItem("Cut");
    cutMenuItem.setActionCommand("Cut");
    JMenuItem copyMenuItem = new JMenuItem("Copy");
    copyMenuItem.setActionCommand("Copy");
    JMenuItem pasteMenuItem = new JMenuItem("Paste");
    pasteMenuItem.setActionCommand("Paste");
    MenuItemListener menuItemListener = new
MenuItemListener();
    newItem.addActionListener(menuItemListener);
    openMenuItem.addActionListener(menuItemListener);
    saveMenuItem.addActionListener(menuItemListener);
    exitMenuItem.addActionListener(menuItemListener);
    cutMenuItem.addActionListener(menuItemListener);
    copyMenuItem.addActionListener(menuItemListener);
    pasteMenuItem.addActionListener(menuItemListener);
    final JCheckBoxMenuItem showWindowMenu = new
JCheckBoxMenuItem("Show About", true);
    showWindowMenu.addItemListener(new ItemListener() {
        public void itemStateChanged(ItemEvent e) {
            if (showWindowMenu.getState()) {
```

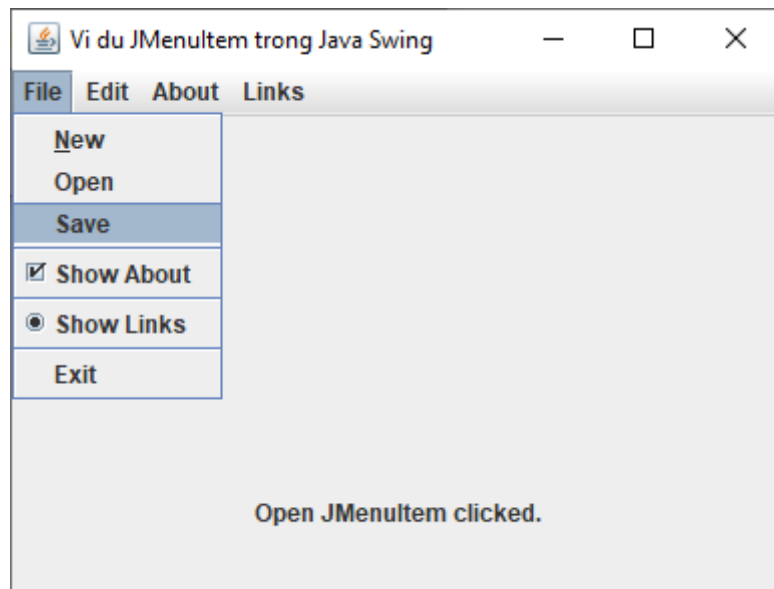
```
        menuBar.add (aboutMenu) ;
    } else {
        menuBar.remove (aboutMenu) ;
    }
}

});

final JRadioButtonMenuItem showLinksMenu = new
JRadioButtonMenuItem("Show Links", true);
showLinksMenu.addItemListener (new ItemListener () {
    public void itemStateChanged (ItemEvent e) {
        if (menuBar.getMenu (3) != null) {
            menuBar.remove (linkMenu) ;
            mainFrame.repaint () ;
        } else {
            menuBar.add (linkMenu) ;
            mainFrame.repaint () ;
        }
    }
}); // Them item toi cac menu
fileMenu.add (newMenuItem) ;
fileMenu.add (openMenuItem) ;
fileMenu.add (saveMenuItem) ;
fileMenu.addSeparator () ;
fileMenu.add (showWindowMenu) ;
fileMenu.addSeparator () ;
fileMenu.add (showLinksMenu) ;
fileMenu.addSeparator () ;
fileMenu.add (exitMenuItem) ;
editMenu.add (cutMenuItem) ;
editMenu.add (copyMenuItem) ;
editMenu.add (pasteMenuItem) ; // Them menu toi menubar
menuBar.add (fileMenu) ;
menuBar.add (editMenu) ;
menuBar.add (aboutMenu) ;
menuBar.add (linkMenu) ; // Them menubar toi frame
mainFrame.setJMenuBar (menuBar) ;
mainFrame.setVisible (true) ;
}
```

```
class JMenuItemListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        statusLabel.setText(e.getActionCommand() + "  
JMenuItem clicked.");  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.3.9 Lớp JPopupMenu trong Java Swing

Mỗi cửa sổ window có một thanh trình đơn (menu bar) được liên kết với nó. Thanh trình đơn này gồm các lựa chọn có sẵn tới người dùng cuối cùng. Các điều khiển Menu và MenuItem là lớp con của lớp MenuComponent.

Lớp JPopupMenu trong Java Swing biểu diễn một menu mà có thể được popup một cách động tại một vị trí đã cho bên trong một thành phần. Cú pháp khai báo cho lớp javax.swing.JPopupMenu là:

```
public class JPopupMenu  
    extends JComponent  
        implements Accessible, MenuElement
```

5.3.9.1 Lớp này kế thừa các phương thức từ các lớp sau:

- javax.swing.JComponent
- java.awt.Container
- java.awt.Component
- java.lang.Object

5.3.9.2 Lớp JPopupMenu gồm các constructor sau:

JPopupMenu(): Xây dựng một JPopupMenu không có "invoker".

JPopupMenu(String label): Xây dựng một JPopupMenu với title đã cho.

5.3.9.3 Chương trình ví dụ lớp JPopupMenu

SwingMenuDemo.java

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.WindowConstants;

public class JPopupMenuExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public JPopupMenuExam1 () {
        prepareGUI ();
    }

    public static void main (String[] args) {
        JPopupMenuExam1 demo = new JPopupMenuExam1 ();
        demo.showPopupMenuDemo ();
    }
}
```

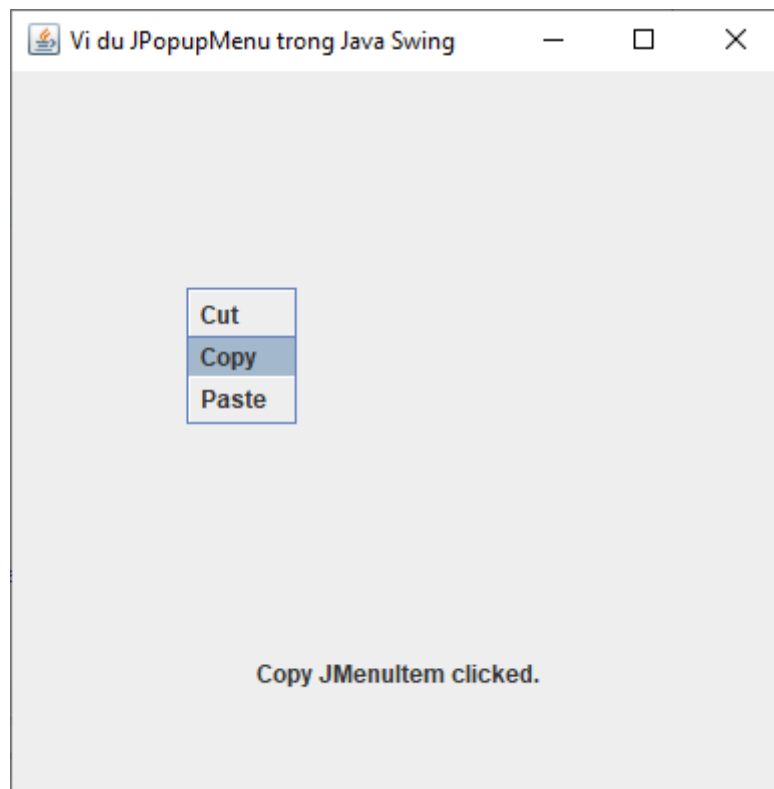


```
private void prepareGUI() {  
    mainFrame = new JFrame("Vi du JPopupMenu trong Java Swing");  
    mainFrame.setSize(400, 400);  
    mainFrame.setLayout(new GridLayout(3, 1));  
    headerLabel = new JLabel("", JLabel.CENTER);  
    statusLabel = new JLabel("", JLabel.CENTER);  
    statusLabel.setSize(350, 100);  
    controlPanel = new JPanel();  
    controlPanel.setLayout(new FlowLayout());  
    mainFrame.add(headerLabel);  
    mainFrame.add(controlPanel);  
    mainFrame.add(statusLabel);  
    mainFrame.setVisible(true);  
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT  
_ON_CLOSE);  
}
```

```
private void showPopupMenuDemo() {  
    final JPopupMenu editMenu = new JPopupMenu("Edit");  
    JMenuItem cutMenuItem = new JMenuItem("Cut");  
    cutMenuItem.setActionCommand("Cut");  
    JMenuItem copyMenuItem = new JMenuItem("Copy");  
    copyMenuItem.setActionCommand("Copy");  
    JMenuItem pasteMenuItem = new JMenuItem("Paste");  
    pasteMenuItem.setActionCommand("Paste");  
    MenuItemListener menuItemListener = new  
MenuItemListener();  
    cutMenuItem.addActionListener(menuItemListener);  
    copyMenuItem.addActionListener(menuItemListener);  
    pasteMenuItem.addActionListener(menuItemListener);  
    editMenu.add(cutMenuItem);  
    editMenu.add(copyMenuItem);  
    editMenu.add(pasteMenuItem);  
    mainFrame.addMouseListener(new MouseAdapter() {  
        public void mouseClicked(MouseEvent e) {  
            editMenu.show(mainFrame, e.getX(), e.getY());  
        }  
    });  
    mainFrame.add(editMenu);  
    mainFrame.setVisible(true);  
}
```

```
    }  
  
    class JMenuItemListener implements ActionListener {  
        public void actionPerformed(ActionEvent e) {  
            statusLabel.setText(e.getActionCommand() + "  
JMenuItem clicked.");  
        }  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.4 Giới thiệu Layout trong Java Swing

Layout (bố cục) nghĩa là sự bố trí sắp xếp các thành phần bên trong Container theo một phương thức nhất định. Nói cách khác, chúng ta đặt các thành phần tại một vị trí cụ thể bên trong **Container**. Tác vụ bố trí này được thực hiện tự động bởi **LayoutManager**. Nếu chúng ta không sử dụng LayoutManager thì các thành phần cũng sẽ được tự động bố trí bởi LayoutManager mặc định. Tuy nhiên, để xử lý một số lượng thành phần lớn với kích cỡ, hình dạng khác nhau và muốn chúng được bố trí theo cách bạn muốn thì việc sử dụng các LayoutManager là cần thiết.

LayoutManager được liên kết với mỗi đối tượng Container. Mỗi LayoutManager là một đối tượng của lớp mà triển khai LayoutManager Interface. Dưới đây là các Interface mà định nghĩa các tính năng của LayoutManager:

- LayoutManager Interface
- LayoutManager2 Interface

5.4.1 Giới thiệu về LayoutManager Interface

LayoutManager Interface được sử dụng để định nghĩa giao diện cho các lớp mà biết cách để bố trí các Container. Cú pháp khai báo cho java.awt.LayoutManager là:

```
public interface LayoutManager
```

LayoutManager Interface này có các phương thức sau:

void addLayoutComponent(String name, Component comp): Nếu layout manager sử dụng một chuỗi mỗi thành phần, thêm thành phần comp tới layout, liên kết nó với chuỗi được xác định bởi tên.

void layoutContainer(Container parent): Bố trí Container đã cho.

Dimension minimumLayoutSize(Container parent): Tính toán các chiều kích cỡ tối thiểu cho Container đã xác định, mà đã cung cấp các thành phần được chứa trong đó.

Dimension preferredLayoutSize(Container parent): Tính toán các chiều kích cỡ được ưu tiên cho Container đã xác định, mà đã cung cấp các thành phần được chứa trong đó.

void removeLayoutComponent(Component comp): Xóa thành phần đã cho từ layout.

5.4.2 Giới thiệu về LayoutManager2 Interface

LayoutManager2 Interface được sử dụng để định nghĩa giao diện cho các lớp mà biết cách bố trí các Container dựa trên một đối tượng ràng buộc Constraint. Cú pháp khai báo cho java.awt.LayoutManager2 là:

```
public interface LayoutManger2
    extends LayoutManager
```

LayoutManager2 Interface bao gồm các phương thức sau:

void addLayoutComponent(Component comp, Object constraints): Thêm thành phần comp đã cho tới layout, bởi sử dụng đối tượng ràng buộc Constraint.

float getLayoutAlignmentX(Container target): Trả về căn chỉnh theo trục x.

float getLayoutAlignmentY(Container target): Trả về căn chỉnh theo trục y.

void invalidateLayout(Container target): Vô hiệu hóa layout, chỉ rằng nếu Layout Manager đã lưu thông tin thì nó nên được loại bỏ.

Dimension maximumLayoutSize(Container target): Tính toán các chiều kích cỡ tối đa cho Container đã xác định, mà đã cung cấp các thành phần chứa trong đó.

5.4.3 Các lớp Layout Manager trong Java Swing

Bảng sau liệt kê danh sách các lớp được sử dụng phổ biến, bạn truy cập link để tìm hiểu chi tiết:

STT	LayoutManager & Mô tả
1	Lớp BoxLayout Lớp BoxLayout, trong java.swing package, được sử dụng để sắp xếp các thành phần hoặc theo chiều dọc hoặc theo chiều ngang
2	Lớp BorderLayout BorderLayout sắp xếp các thành phần để phù hợp với 5 miền: đông, tây, nam, bắc và trung tâm
3	Lớp CardLayout CardLayout xem xét mỗi thành phần trong Container như là một card. Chỉ có một card là nhìn thấy tại một thời điểm
4	Lớp FlowLayout FlowLayout là Layout mặc định. Nó bố trí các thành phần trong luồng định hướng (trong một line, line sau nối tiếp line trước)
5	Lớp GridLayout GridLayout quản lý các thành phần trong lưới hình chữ nhật. Một thành phần được hiển thị trong mỗi hình chữ nhật con.

6	<u>Lớp GridBagLayout</u> GridBagLayout là một lớp quản lý layout linh động. Đối tượng của GridBagLayout căn chỉnh các thành phần theo chiều dọc, ngang hoặc theo baseline của chúng mà không yêu cầu các thành phần phải có cùng kích cỡ.
7	<u>Lớp GroupLayout</u> GroupLayout nhóm các thành phần theo cấu trúc thứ bậc để đặt chúng trong một Container
8	<u>Lớp SpringLayout</u> SpringLayout đặt vị trí các con của Container liên kết với nó tuân theo một tập hợp các ràng buộc.

5.4.3.1 BorderLayout trong Java Swing

Lớp BorderLayout trong Java Swing được sử dụng để sắp xếp các thành phần hoặc theo chiều dọc hoặc theo chiều ngang. Để phục vụ mục đích này, lớp BorderLayout cung cấp 4 hằng:

1. **public static final int X_AXIS**
2. **public static final int Y_AXIS**
3. **public static final int LINE_AXIS**
4. **public static final int PAGE_AXIS**

5.4.3.1.1 Constructor của lớp BorderLayout

BorderLayout(Container c, int axis): tạo một box layout mà sắp xếp các thành phần theo trục đã cho.

5.4.3.1.2 Ví dụ về lớp BorderLayout với trục Y

```
package vn.plpsoft.swing;

import java.awt.Button;
import java.awt.Frame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import javax.swing.BoxLayout;

public class BorderLayoutExam1 extends Frame {
    Button buttons[];

    public BorderLayoutExam1 () {
```

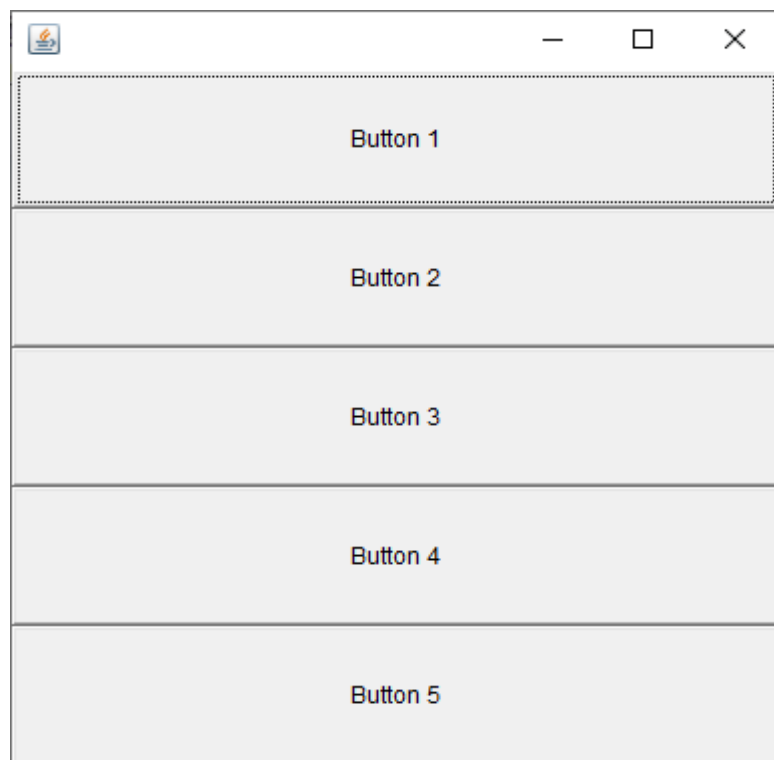
```
buttons = new Button[5];

for (int i = 0; i < 5; i++) {
    buttons[i] = new Button("Button " + (i + 1));
    add(buttons[i]);
}

setLayout(new BorderLayout(this, BorderLayout.Y_AXIS));
setSize(400, 400);
setVisible(true);
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        System.exit(0);
    }
});
}

public static void main(String args[]) {
    BorderLayoutExam1 boxLayout = new BorderLayoutExam1();
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.1.3 Ví dụ về lớp BorderLayout với trục X

```
package vn.plpsoft.swing;

import java.awt.Button;
import java.awt.Frame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.BoxLayout;

public class BorderLayoutExam2 extends Frame {
    Button buttons[];

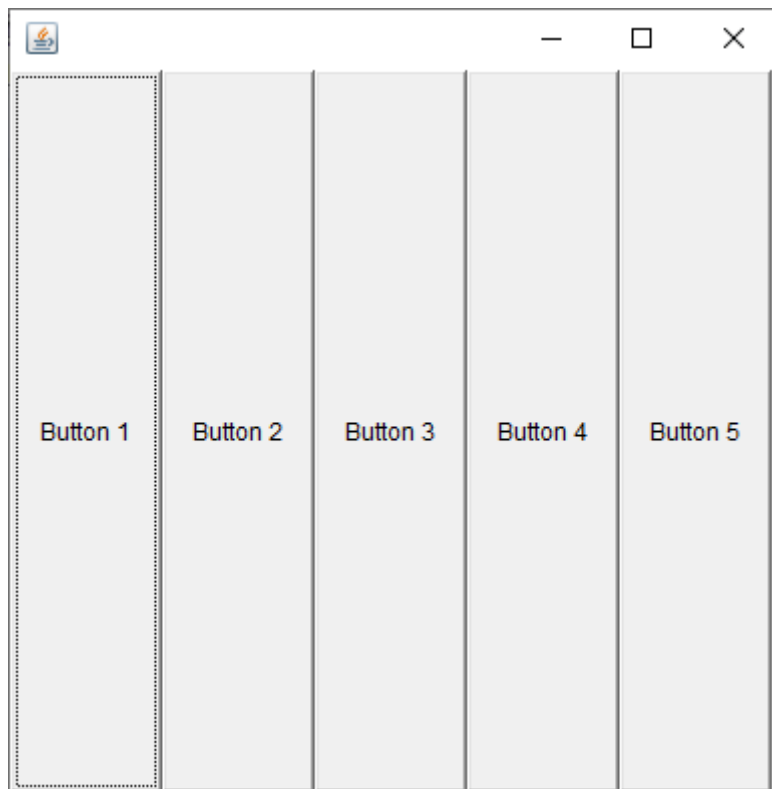
    public BorderLayoutExam2() {
        buttons = new Button[5];

        for (int i = 0; i < 5; i++) {
            buttons[i] = new Button("Button " + (i + 1));
            add(buttons[i]);
        }

        setLayout(new BoxLayout(this, BoxLayout.X_AXIS));
        setSize(400, 400);
        setVisible(true);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent we) {
                System.exit(0);
            }
        });
    }

    public static void main(String args[]) {
        BorderLayoutExam2 boxLayout = new BorderLayoutExam2();
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.2 BorderLayout trong Java Swing

Lớp **BorderLayout** trong Java Swing sắp xếp các thành phần để phù hợp với 5 miền: EAST, WEST, SOUTH, NORTH và CENTER. Nó là layout mặc định của Frame hoặc Window. Mỗi khu vực (miền) chỉ có thể chứa một thành phần và mỗi thành phần trong mỗi khu vực được nhận diện bởi các hằng tương ứng là:

public static final int NORTH

public static final int SOUTH

public static final int EAST

public static final int WEST

public static final int CENTER

Cú pháp để khai báo lớp `Java.awt.BorderLayout` là:

```
public class BorderLayout
    extends Object
        implements LayoutManager2, Serializable
```

5.4.3.2.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.lang.Object`

Lớp `BorderLayout` bao gồm các trường sau:

- **static String AFTER_LAST_LINE**: Giống `PAGE_END`.

- **static String AFTER_LINE_ENDS:** Giống LINE_END.
- **static String BEFORE_FIRST_LINE:** Giống PAGE_START.
- **static String BEFORE_LINE_BEGINS:** Giống LINE_START.
- **static String CENTER:** Đây là ràng buộc bố trí trung tâm (ở giữa container).
- **static String EAST:** Ràng buộc bố trí theo hướng đông (cạnh phải của container).
- **static String LINE_END:** Thành phần tới phần cuối của line direction cho layout
- **static String LINE_START:** Thành phần tới phần bắt đầu của line direction cho layout
- **static String NORTH:** Đây là ràng buộc bố trí theo hướng bắc (phần trên của container).
- **static String PAGE_END:** Thành phần đến sau dòng cuối cùng (last line) của nội dung layout
- **static String PAGE_START:** Thành phần tới trước dòng đầu tiên (first line) của nội dung layout.
- **static String SOUTH:** Ràng buộc bố trí hướng nam (phần dưới của container)
- **static String WEST:** Ràng buộc bố trí hướng tây (cạnh trái của container)

5.4.3.2.2 Lớp BorderLayout bao gồm các constructor sau:

BorderLayout(): Xây dựng một Border Layout mới, ở đó giữa các thành phần là không có khoảng cách.

BorderLayout(int hgap, int vgap): Xây dựng một Border Layout với các khoảng cách gap theo chiều dọc và ngang đã xác định giữa các thành phần.

5.4.3.2.3 Các phương thức của lớp BorderLayout

STT	Phương thức & Mô tả
1	void addLayoutComponent(Component comp, Object constraints) Thêm thành phần comp đã cho tới layout, bởi sử dụng đối tượng Constraint đã xác định.
2	void addLayoutComponent(String name, Component comp) Nếu Layout Manager sử dụng một chuỗi mỗi thành phần, thêm thành phần comp tới layout, liên kết nó với chuỗi đã được xác định bởi tên

3	int getHgap() Trả về khoảng cách theo chiều ngang giữa các thành phần
4	float getLayoutAlignmentX(Container parent) Trả về căn chỉnh theo trục x
5	float getLayoutAlignmentY(Container parent) Trả về căn chỉnh theo trục y
6	int getVgap() Trả về khoảng cách theo chiều dọc giữa các thành phần
7	void invalidateLayout(Container target) Vô hiệu hóa layout, chỉ rằng nếu Layout Manager đã lưu thông tin thì nó nên loại bỏ
8	void layoutContainer(Container target) Bố trí tham số container bởi sử dụng Border Layout này
9	Dimension maximumLayoutSize(Container target) Trả về kích cỡ các chiều tối đa cho layout này, đã cung cấp các thành phần trong target đã cho.
10	Dimension minimumLayoutSize(Container target) Xác định kích cỡ tối thiểu cho target bởi sử dụng layout manager
11	Dimension preferredLayoutSize(Container target) Xác định kích cỡ được ưu tiên cho target bởi sử dụng layout manager, dựa trên các thành phần của container.
12	void removeLayoutComponent(Component comp) Xóa thành phần comp đã cho từ Border Layout này
13	void setHgap(int hgap) Thiết lập khoảng cách theo chiều ngang giữa các thành phần
14	void setVgap(int vgap) Thiết lập khoảng cách theo chiều dọc giữa các thành phần

15	String toString() Trả về một biểu diễn chuỗi của trạng thái Border Layout này
----	---

5.4.3.2.4 Chương trình ví dụ đơn giản về lớp BorderLayout

```
package vn.plpsoft.swing;

import java.awt.BorderLayout;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.WindowConstants;

public class BorderLayoutExam1 {
    JFrame frame;

    BorderLayoutExam1() {
        frame = new JFrame();

        JButton b1 = new JButton("NORTH");
        JButton b2 = new JButton("SOUTH");
        JButton b3 = new JButton("EAST");
        JButton b4 = new JButton("WEST");
        JButton b5 = new JButton("CENTER");

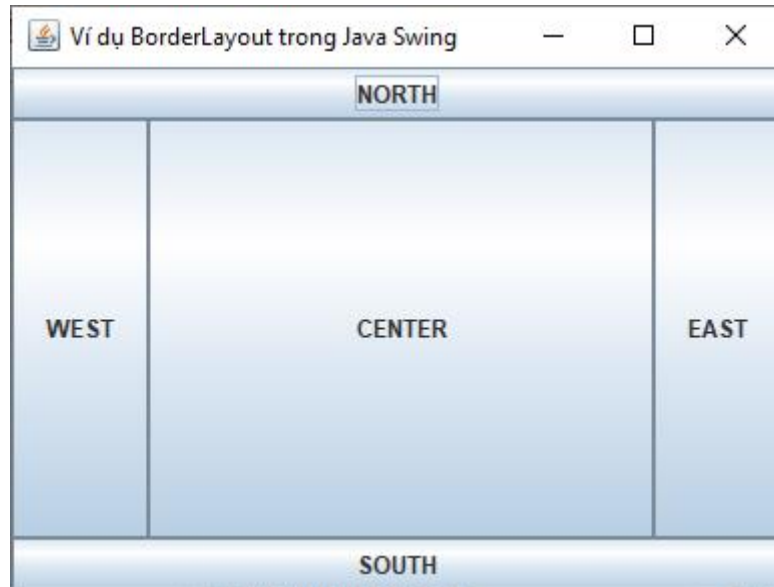
        frame.add(b1, BorderLayout.NORTH);
        frame.add(b2, BorderLayout.SOUTH);
        frame.add(b3, BorderLayout.EAST);
        frame.add(b4, BorderLayout.WEST);
        frame.add(b5, BorderLayout.CENTER);

        frame.setSize(400, 300);
        frame.setVisible(true);
        frame.setTitle("Ví dụ BorderLayout trong Java Swing");
        frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

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

```
        new BorderLayoutExam1 ();  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



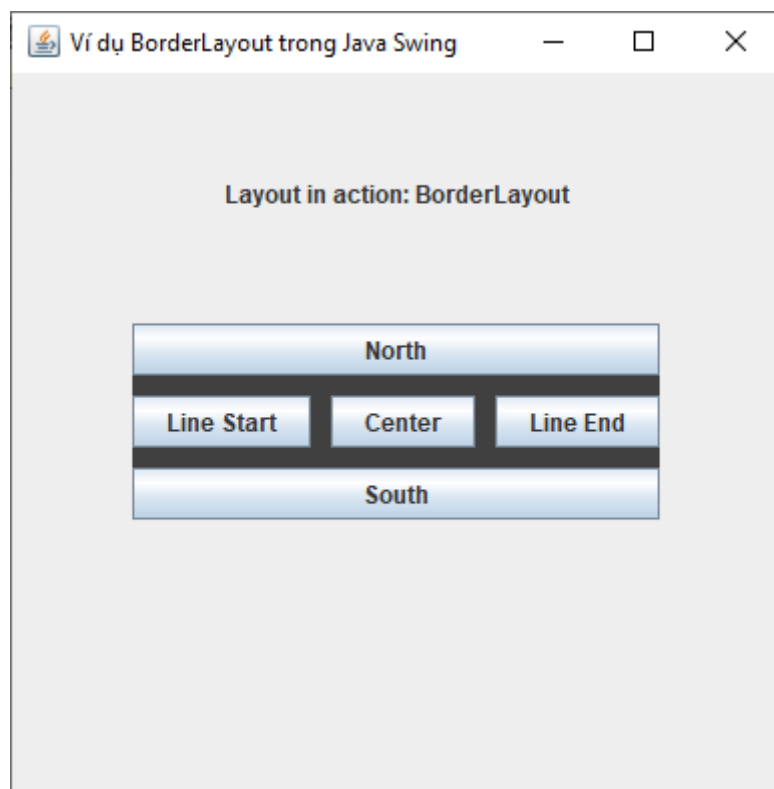
5.4.3.2.5 Chương trình ví dụ khác về lớp BorderLayout

```
package vn.plpsoft.swing;  
  
import java.awt.BorderLayout;  
import java.awt.Color;  
import java.awt.FlowLayout;  
import java.awt.GridLayout;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
  
import javax.swing.JButton;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JPanel;  
import javax.swing.WindowConstants;  
  
public class BorderLayoutExam2 {  
    private JFrame frame;  
    private JLabel headerLabel;  
    private JLabel statusLabel;  
    private JPanel controlPanel;
```

```
public BorderLayoutExam2 () {  
    prepareGUI ();  
}  
  
public static void main (String[] args) {  
    BorderLayoutExam2 demo = new BorderLayoutExam2 ();  
    demo.showBorderLayoutDemo ();  
}  
  
private void prepareGUI () {  
    frame = new JFrame ();  
    frame.setSize (400, 400);  
    frame.setLayout (new GridLayout (3, 1));  
    headerLabel = new JLabel ("", JLabel.CENTER);  
    statusLabel = new JLabel ("", JLabel.CENTER);  
    statusLabel.setSize (350, 100);  
    frame.addWindowListener (new WindowAdapter () {  
        public void windowClosing (WindowEvent windowEvent) {  
            System.exit (0);  
        }  
    });  
    controlPanel = new JPanel ();  
    controlPanel.setLayout (new FlowLayout ());  
    frame.add (headerLabel);  
    frame.add (controlPanel);  
    frame.add (statusLabel);  
    frame.setVisible (true);  
    frame.setTitle ("Ví dụ BorderLayout trong Java Swing");  
    frame.setDefaultCloseOperation (WindowConstants.EXIT_ON_CLOSE);  
}  
  
private void showBorderLayoutDemo () {  
    headerLabel.setText ("Layout in action: BorderLayout");  
    JPanel panel = new JPanel ();  
    panel.setBackground (Color.darkGray);  
    panel.setSize (300, 300);  
    BorderLayout layout = new BorderLayout ();  
    layout.setHgap (10);
```

```
layout.setVgap(10);  
panel.setLayout(layout);  
  
panel.add(new JButton("Center"), BorderLayout.CENTER);  
panel.add(new JButton("Line Start"), BorderLayout.LINE_START);  
panel.add(new JButton("Line End"), BorderLayout.LINE_END);  
panel.add(new JButton("East"), BorderLayout.EAST);  
panel.add(new JButton("West"), BorderLayout.WEST);  
panel.add(new JButton("North"), BorderLayout.NORTH);  
panel.add(new JButton("South"), BorderLayout.SOUTH);  
controlPanel.add(panel);  
frame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.3 CardLayout trong Java Swing

Lớp **CardLayout** trong Java Swing quản lý các thành phần theo một phương thức mà chỉ có một thành phần là nhìn thấy (visible) tại một thời điểm. Nó xem xét mỗi thành phần như là một card, và container hoạt động như là một stack của các card, và đó là lý do tại sao gọi là CardLayout.

Cú pháp khai báo cho lớp `Java.awt.CardLayout` là:

```
public class CardLayout
    extends Object
        implements LayoutManager2, Serializable
```

5.4.3.3.1 Lớp này kế thừa các phương thức từ các lớp sau:

- java.lang.Object
-

5.4.3.3.2 Các constructor của lớp CardLayout

CardLayout(): tạo một Card Layout với các khoảng cách theo chiều dọc và ngang là 0.

CardLayout(int hgap, int vgap): tạo một Card Layout với các khoảng cách gap theo chiều dọc và ngang đã cho

5.4.3.3.3 Các phương thức được sử dụng phổ biến của lớp CardLayout

public void next(Container parent): được sử dụng để lật tới card tiếp theo của container đã cho

public void previous(Container parent): được sử dụng để lật tới card trước đó của container đã cho

public void first(Container parent): được sử dụng để lật tới card đầu tiên của container đã cho

public void last(Container parent): được sử dụng để lật tới card cuối cùng của container đã cho

public void show(Container parent, String name): được sử dụng để lật tới card đã được xác định bởi tên name đã cho.

5.4.3.3.4 Chương trình ví dụ về lớp CardLayout

```
package vn.plpsoft.swing;

import java.awt.CardLayout;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.WindowConstants;
```

```
public class CardLayoutExam1 extends JFrame implements ActionListener {
    CardLayout card;
    JButton b1, b2, b3;
    Container c;

    CardLayoutExam1() {
        c = getContentPane();
        // tao doi tuong CardLayout
        // khong gian chieu ngang la 60 va chieu doc la 50
        card = new CardLayout(60, 50);
        c.setLayout(card);

        b1 = new JButton("Apple");
        b2 = new JButton("Mango");
        b3 = new JButton("Orange");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);

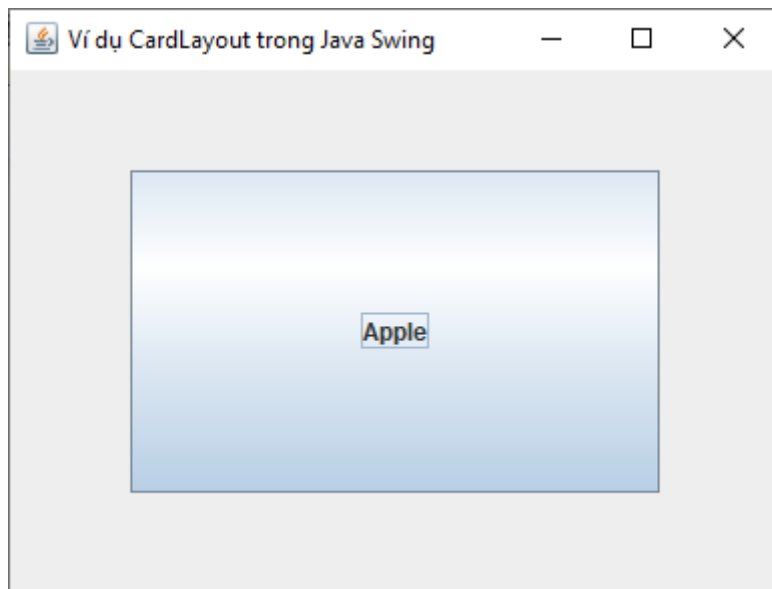
        c.add("a", b1);
        c.add("b", b2);
        c.add("c", b3);

        this.setTitle("Ví dụ CardLayout trong Java Swing");
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public void actionPerformed(ActionEvent e) {
        card.next(c);
    }

    public static void main(String[] args) {
        CardLayoutExam1 cl = new CardLayoutExam1();
        cl.setSize(400, 300);
        cl.setVisible(true);
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.3.5 Ví dụ khác về lớp CardLayout

```
package vn.plpsoft.swing;

import java.awt.CardLayout;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.DefaultComboBoxModel;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

public class CardLayoutExam2 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
```

```
public CardLayoutExam2() {
    prepareGUI();
}

public static void main(String[] args) {
    CardLayoutExam2 demo = new CardLayoutExam2();
    demo.showCardLayoutDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame("Vi du CardLayout trong Java Swing");
    mainFrame.setSize(400, 350);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showCardLayoutDemo() {
    headerLabel.setText("Layout in action: CardLayout");
    final JPanel panel = new JPanel();
    panel.setBackground(Color.DARK_GRAY);
    panel.setSize(300, 300);
    CardLayout layout = new CardLayout();
    layout.setHgap(10);
    layout.setVgap(10);
    panel.setLayout(layout);
    JPanel buttonPanel = new JPanel(new FlowLayout());
    buttonPanel.add(new JButton("OK"));
    buttonPanel.add(new JButton("Cancel"));
    JPanel textBoxPanel = new JPanel(new FlowLayout());
```

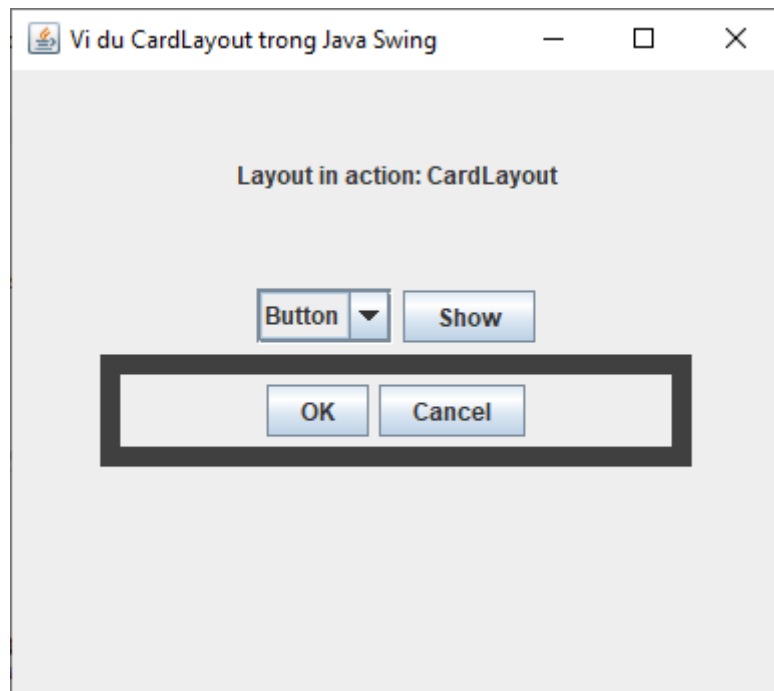
```
textBoxPanel.add(new JLabel("Name:"));
textBoxPanel.add(new JTextField(20));
panel.add("Button", buttonPanel);
panel.add("Text", textBoxPanel);

final DefaultComboBoxModel panelName = new DefaultComboBoxModel();
panelName.addElement("Button");
panelName.addElement("Text");

final JComboBox listCombo = new JComboBox(panelName);
listCombo.setSelectedIndex(0);
JScrollPane listComboScrollPane = new JScrollPane(listCombo);
JButton showButton = new JButton("Show");
showButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String data = "";
        if (listCombo.getSelectedIndex() != -1) {
            CardLayout cardLayout = (CardLayout)
(panel.getLayout());
            cardLayout.show(panel,
                            (String)
listCombo.getItemAt(listCombo.getSelectedIndex()));
        }
        statusLabel.setText(data);
    }
});

controlPanel.add(listComboScrollPane);
controlPanel.add(showButton);
controlPanel.add(panel);
mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.4 FlowLayout trong Java Swing

Lớp **FlowLayout** trong Java Swing được sử dụng để sắp xếp các thành phần trong một line, line sau nối tiếp line trước (trong một luồng từ trái qua phải left-to-right flow). Nó là Layout mặc định của applet hoặc panel. Cú pháp để khai báo lớp `java.awt.FlowLayout` là:

```
public class FlowLayout
    extends Object
        implements LayoutManager, Serializable
```

5.4.3.4.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.lang.Object`

5.4.3.4.2 Lớp FlowLayout này bao gồm các trường sau:

- **static int CENTER**: Giá trị này chỉ rằng mỗi hàng của các thành phần nên được căn chỉnh vào giữa.
- **static int LEADING**: Giá trị này chỉ rằng mỗi hàng của các thành phần nên được căn chỉnh theo cạnh chủ đạo (leading edge) theo hướng của container, ví dụ: căn chỉnh theo cạnh trái theo hướng left-to-right.
- **static int LEFT**: Giá trị này chỉ rằng mỗi hàng của các thành phần nên được căn chỉnh trái.
- **static int RIGHT**: Giá trị này chỉ rằng mỗi hàng của các thành phần nên được căn chỉnh phải.

- **static int TRAILING:** Giá trị này chỉ rằng mỗi hàng của các thành phần nên được căn chỉnh theo cạnh trailing edge theo hướng của container, ví dụ: căn chỉnh theo cạnh phải theo hướng left-to-right.

5.4.3.4.3 Các constructor được sử dụng phổ biến của lớp FlowLayout

FlowLayout(): tạo một Flow Layout với căn chỉnh trung tâm và một khoảng cách gap theo chiều dọc và ngang là 5 đơn vị.

FlowLayout(int align): tạo một Flow Layout với căn chỉnh align đã cho và một khoảng cách gap theo chiều dọc và ngang là 5 đơn vị.

FlowLayout(int align, int hgap, int vgap): tạo một Flow Layout với căn chỉnh align đã cho và một khoảng cách gap theo chiều dọc và ngang đã được xác định.

5.4.3.4.4 Ví dụ đầu tiên về FlowLayout

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.WindowConstants;

public class FlowLayoutExam1 {
    JFrame frame;

    FlowLayoutExam1() {
        frame = new JFrame();

        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");

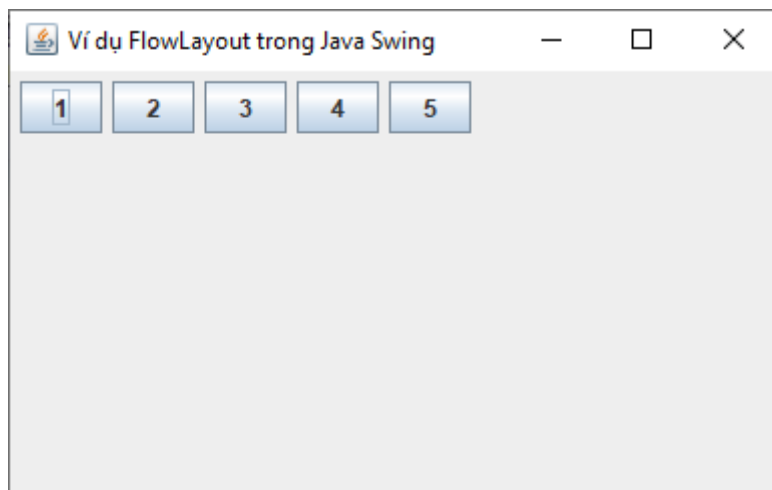
        frame.add(b1);
        frame.add(b2);
        frame.add(b3);
        frame.add(b4);
        frame.add(b5);
    }
}
```

```
// thiết lập flow layout la can chinh trai
frame.setLayout(new FlowLayout(FlowLayout.LEFT));

frame.setSize(400, 250);
frame.setVisible(true);
frame.setTitle("Ví dụ FlowLayout trong Java Swing");
frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CL
OSE);
}

public static void main(String[] args) {
    new FlowLayoutExam1();
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.4.5 Ví dụ khác về FlowLayout

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

```
import javax.swing.WindowConstants;

public class FlowLayoutExam2 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;

    public FlowLayoutExam2() {
        prepareGUI();
    }

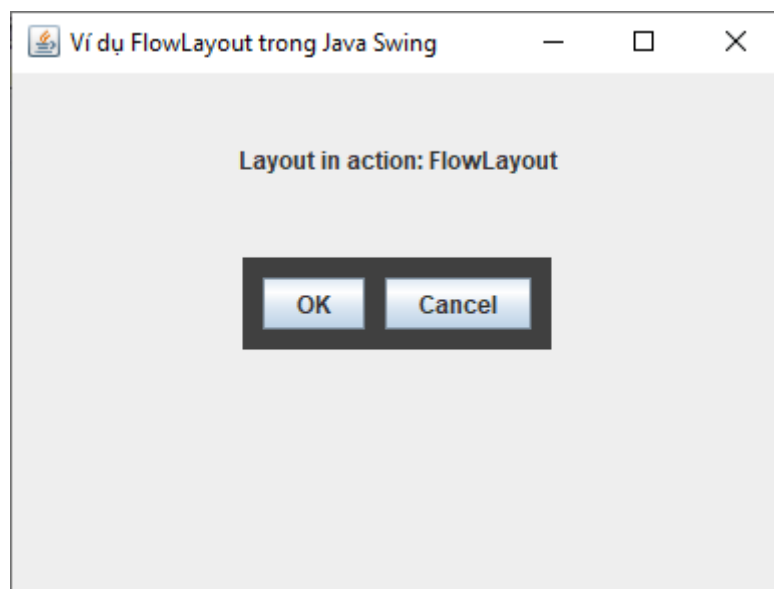
    public static void main(String[] args) {
        FlowLayoutExam2 demo = new FlowLayoutExam2();
        demo.showFlowLayoutDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame();
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ FlowLayout trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON
_CLOSE);
    }

    private void showFlowLayoutDemo() {
        headerLabel.setText("Layout in action: FlowLayout");
        JPanel panel = new JPanel();
```

```
        panel.setBackground(Color.darkGray);  
        panel.setSize(200, 200);  
        FlowLayout layout = new FlowLayout();  
        layout.setHgap(10);  
        layout.setVgap(10);  
        panel.setLayout(layout);  
        panel.add(new JButton("OK"));  
        panel.add(new JButton("Cancel"));  
        controlPanel.add(panel);  
        mainFrame.setVisible(true);  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.5 GridLayout trong Java Swing

Lớp **GridLayout** trong Java Swing sắp xếp các thành phần trong một lưới hình chữ nhật. Một thành phần được hiển thị trong mỗi hình chữ nhật. Cú pháp khai báo cho lớp `java.awt.GridLayout` là:

```
public class GridLayout  
    extends Object  
        implements LayoutManager, Serializable
```

5.4.3.5.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.lang.Object`

5.4.3.5.2 Lớp **GridLayout** gồm các constructor sau:

1. **GridLayout()**: Tạo một grid layout với mặc định là một cột mỗi thành phần, trong một hàng đơn.
2. **GridLayout(int rows, int columns)**: Tạo một grid layout với số hàng và cột đã cho, và không có khoảng cách giữa các thành phần.
3. **GridLayout(int rows, int columns, int hgap, int vgap)**: Tạo một grid layout với các hàng và cột đã cho cùng với các khoảng cách theo chiều dọc và ngang đã xác định.

5.4.3.5.3 Các phương thức của lớp **GridLayout** trong Java Swing

void **addLayoutComponent**(String name, Component comp): Thêm thành phần comp đã cho với tên đã xác định tới layout.

1. void **layoutContainer**(Container parent): Bố trí container đã cho bởi sử dụng layout này.
2. Dimension **minimumLayoutSize**(Container parent): Xác định kích cỡ tối thiểu của tham số container bởi sử dụng Grid Layout.
3. Dimension **preferredLayoutSize**(Container parent): Xác định kích cỡ được ưu tiên của tham số container bởi sử dụng Grid Layout.
4. void **removeLayoutComponent**(Component comp): Xóa thành phần đã cho từ layout.
5. void **setColumns**(int cols): Thiết lập số cột trong layout này tới giá trị đã cho.
6. void **setHgap**(int hgap): Thiết lập khoảng cách theo chiều ngang giữa các thành phần tới giá trị đã cho.
7. void **setRows**(int rows): Thiết lập số hàng trong layout này tới giá trị đã cho.
8. void **setVgap**(int vgap): Thiết lập khoảng cách theo chiều dọc giữa các thành phần tới giá trị đã cho.

5.4.3.5.4 Ví dụ đầu tiên về **GridLayout**

```
package vn.plpsoft.swing;

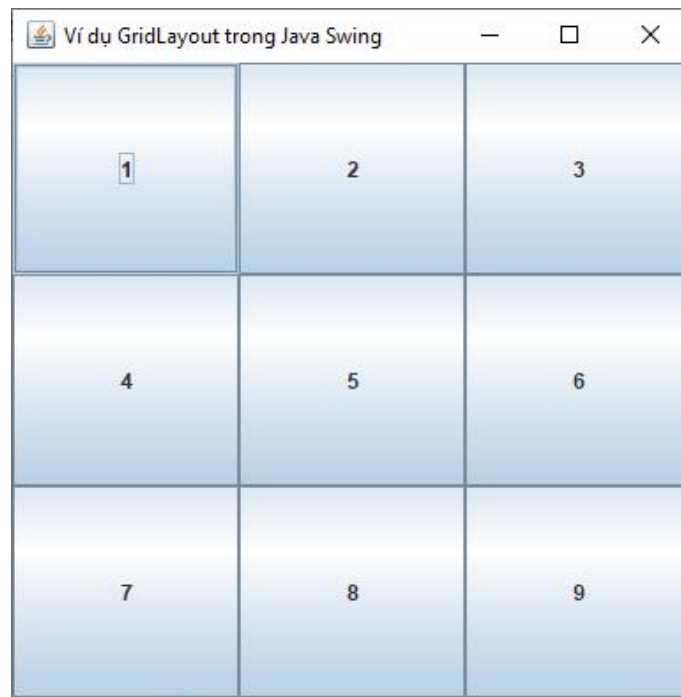
import java.awt.GridLayout;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.WindowConstants;

public class GridLayoutExam1 {
    JFrame frame;
```

```
GridLayoutExam1() {  
    frame = new JFrame();  
  
    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");  
  
    frame.add(b1);  
    frame.add(b2);  
    frame.add(b3);  
    frame.add(b4);  
    frame.add(b5);  
    frame.add(b6);  
    frame.add(b7);  
    frame.add(b8);  
    frame.add(b9);  
  
    // thiết lập 3 hàng và 3 cột cho grid layout  
    frame.setLayout(new GridLayout(3, 3));  
  
    frame.setSize(400, 400);  
    frame.setVisible(true);  
    frame.setTitle("Ví dụ GridLayout trong Java Swing");  
    frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
};  
  
    }  
  
    public static void main(String[] args) {  
        new GridLayoutExam1();  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.5 Ví dụ khác về GridLayout

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class GridLayoutExam2 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;

    public GridLayoutExam2 () {
        prepareGUI ();
    }
}
```

```
public static void main(String[] args) {
    GridLayoutExam2 swingLayoutDemo = new GridLayoutExam2();
    swingLayoutDemo.showGridLayoutDemo();
}

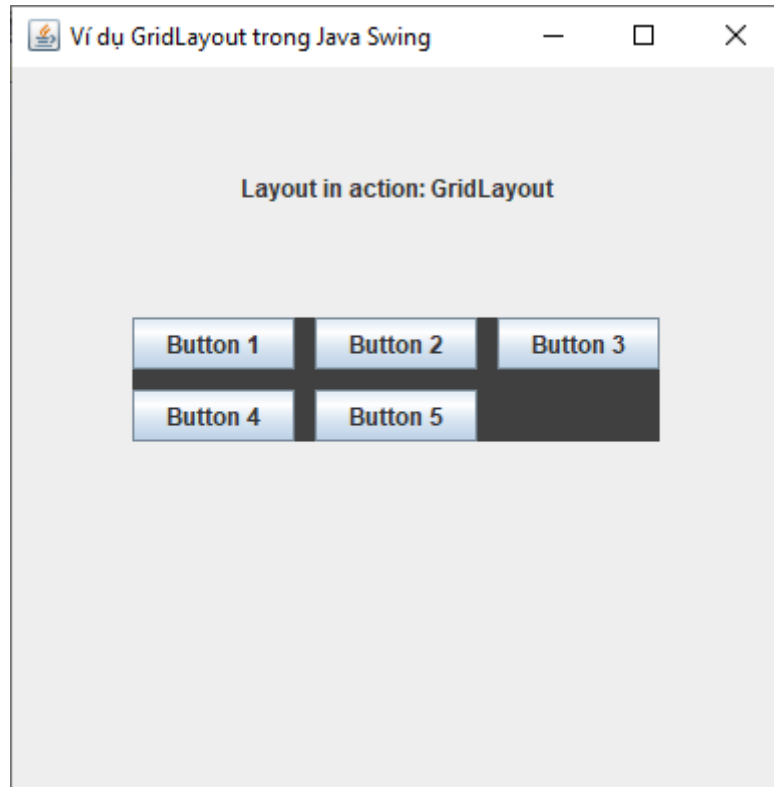
private void prepareGUI() {
    mainFrame = new JFrame();
    mainFrame.setSize(400, 400);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
    mainFrame.setTitle("Ví dụ GridLayout trong Java Swing");
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showGridLayoutDemo() {
    headerLabel.setText("Layout in action: GridLayout");
    JPanel panel = new JPanel();
    panel.setBackground(Color.darkGray);
    panel.setSize(300, 300);
    GridLayout layout = new GridLayout(0, 3);
    layout.setHgap(10);
    layout.setVgap(10);

    panel.setLayout(layout);
    panel.add(new JButton("Button 1"));
    panel.add(new JButton("Button 2"));
    panel.add(new JButton("Button 3"));
    panel.add(new JButton("Button 4"));
    panel.add(new JButton("Button 5"));
```

```
        controlPanel.add(panel);  
        mainFrame.setVisible(true);  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.6 GridBagLayout trong Java Swing

Lớp **GridBagLayout** trong Java Swing là một lớp quản lý layout linh động. Đối tượng của GridBagLayout căn chỉnh các thành phần theo chiều dọc, ngang hoặc theo baseline của chúng mà không yêu cầu các thành phần phải có cùng kích cỡ. Cú pháp khai báo cho lớp **java.awt.GridBagLayout** là:

```
public class GridBagLayout  
    extends Object  
        implements LayoutManager2, Serializable
```

Lớp GridBagLayout có **GridBagLayout()** constructor để tạo một GridBag Layout.

5.4.3.6.1 Lớp này kế thừa các phương thức từ các lớp sau:

- java.lang.Object

5.4.3.6.2 Lớp GridBagLayout này gồm các trường sau:

- **static int DEFAULT_SIZE:** Chỉ kích cỡ từ đó các thành phần hoặc khoảng cách gap nên được sử dụng cho một giá trị dãy cụ thể.
- **static int PREFERRED_SIZE:** Chỉ kích cỡ được ưu tiên từ đó các thành phần hoặc khoảng cách gap nên được sử dụng cho một giá trị dãy cụ thể.

5.4.3.6.3 Ví dụ về lớp GridBagLayout

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class GridBagLayoutExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public GridBagLayoutExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        GridBagLayoutExam1 demo = new GridBagLayoutExam1();
        demo.showGridBagLayoutDemo();
    }

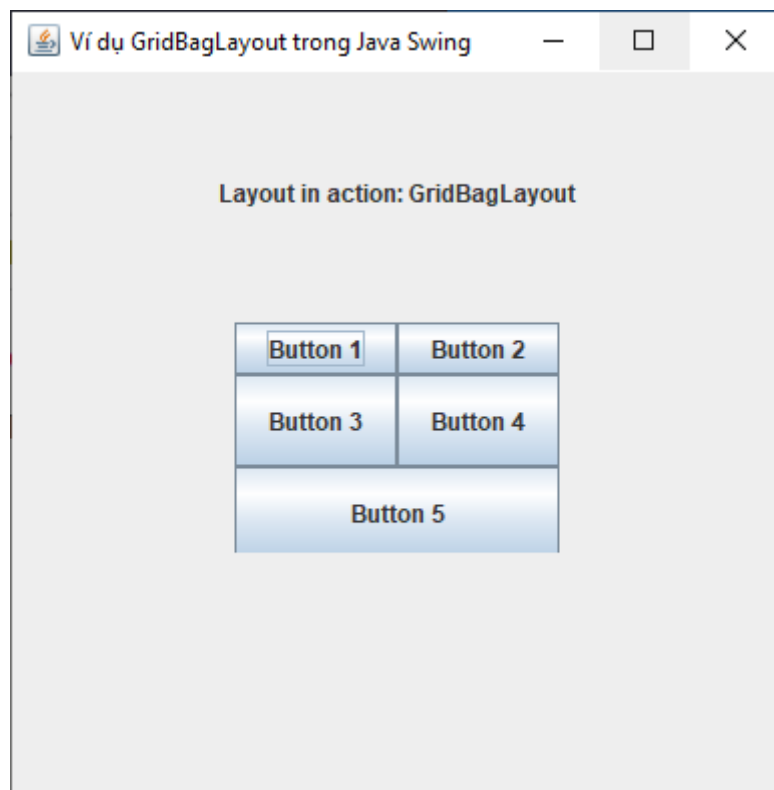
    private void prepareGUI() {
```

```
mainFrame = new JFrame();
mainFrame.setSize(400, 400);
mainFrame.setLayout(new GridLayout(3, 1));
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setSize(350, 100);
mainFrame.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
});
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
mainFrame.setTitle("Ví dụ GridBagLayout trong Java
Swing");
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showGridBagLayoutDemo() {
    headerLabel.setText("Layout in action: GridBagLayout");
    JPanel panel = new JPanel();
    panel.setBackground(Color.darkGray);
    panel.setSize(300, 300);
    GridBagLayout layout = new GridBagLayout();
    panel.setLayout(layout);
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.gridx = 0;
    gbc.gridy = 0;
    panel.add(new JButton("Button 1"), gbc);
    gbc.gridx = 1;
    gbc.gridy = 0;
    panel.add(new JButton("Button 2"), gbc);
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipady = 20;
```

```
gbc.gridx = 0;
gbc.gridy = 1;
panel.add(new JButton("Button 3"), gbc);
gbc.gridx = 1;
gbc.gridy = 1;
panel.add(new JButton("Button 4"), gbc);
gbc.gridx = 0;
gbc.gridy = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.gridwidth = 2;
panel.add(new JButton("Button 5"), gbc);
controlPanel.add(panel);
mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.7 GroupLayout trong Java Swing

Lớp **GroupLayout** trong Java Swing nhóm các thành phần theo cấu trúc thứ bậc để đặt chúng trong một Container. Cú pháp khai báo của lớp `javax.swing.GroupLayout` là:

```
public class GroupLayout
```



```
extends Object
```

```
implements LayoutManager2
```

Lớp GroupLayout này có một constructor là **GroupLayout(Container host)** để tạo một GroupLayout cho container đã cho.

5.4.3.7.1 Lớp này kế thừa các phương thức từ các lớp sau:

- java.lang.Object

5.4.3.7.2 Lớp GroupLayout có các trường sau:

- **static int DEFAULT_SIZE**: Chỉ kích cỡ từ đó các thành phần hoặc khoảng cách gap nên được sử dụng cho một giá trị dãy cụ thể.
- **static int PREFERRED_SIZE**: Chỉ kích cỡ được ưu tiên từ đó các thành phần hoặc khoảng cách gap nên được sử dụng cho một giá trị dãy cụ thể.

5.4.3.7.3 Ví dụ lớp GroupLayout

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;

import javax.swing.GroupLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class GroupLayoutExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public GroupLayoutExam1() {
        prepareGUI();
    }
}
```

```
}

public static void main(String[] args) {
    GroupLayoutExam1 demo = new GroupLayoutExam1();
    demo.showGroupLayoutDemo();
}

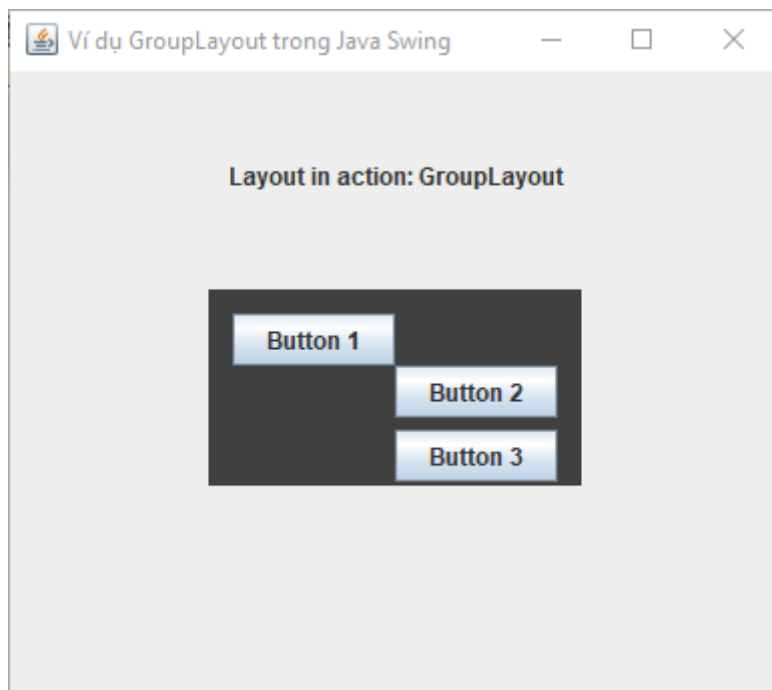
private void prepareGUI() {
    mainFrame = new JFrame();
    mainFrame.setSize(400, 350);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
    mainFrame.setTitle("Ví dụ GroupLayout trong Java Swing");
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showGroupLayoutDemo() {
    headerLabel.setText("Layout in action: GroupLayout");
    JPanel panel = new JPanel();
    panel.setBackground(Color.darkGray);
    panel.setSize(200, 200);
    GroupLayout layout = new GroupLayout(panel);
    layout.setAutoCreateGaps(true);
    layout.setAutoCreateContainerGaps(true);
    JButton btn1 = new JButton("Button 1");
    JButton btn2 = new JButton("Button 2");
    JButton btn3 = new JButton("Button 3");
    layout.setHorizontalGroup(layout.createSequentialGroup()
        .addComponent(btn1)
        .addGroup(layout.createSequentialGroup().addGroup
(layout
```

```
.createParallelGroup(GroupLayout.Alignment.LEADING)
    .addComponent(btn2).addComponent(btn3))) );

layout.setVerticalGroup(
    layout.createSequentialGroup().addComponent(btn1)
    .addComponent(btn2).addComponent(btn3));
panel.setLayout(layout);
controlPanel.add(panel);
mainFrame.setVisible(true);
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.4.3.8 GroupLayout trong Java Swing

Lớp **SpringLayout** trong Java Swing đặt vị trí các con của Container liên kết với nó tuân theo một tập hợp các ràng buộc. Cú pháp để khai báo lớp `javax.swing.SpringLayout` là:

```
public class SpringLayout
    extends Object
    implements LayoutManager2
```

Lớp này có **SpringLayout()** constructor để tạo một SpringLayout mới.

5.4.3.8.1 Lớp này kế thừa các phương thức từ các lớp sau:

- `java.lang.Object`

5.4.3.8.2 Lớp `SpringLayout` bao gồm các trường sau:

- **static String BASELINE:** Xác định baseline của một thành phần.
- **static String EAST:** Xác định cạnh phải của hình chữ nhật bao của một thành phần.
- **static String HEIGHT:** Xác định chiều cao của hình chữ nhật bao của một thành phần.
- **static String HORIZONTAL_CENTER:** Xác định sự căn chỉnh ngang của hình chữ nhật bao của một thành phần.
- **static String NORTH:** Xác định cạnh trên của hình chữ nhật bao của một thành phần.
- **static String SOUTH:** Xác định cạnh dưới của hình chữ nhật bao của một thành phần.
- **static String VERTICAL_CENTER:** Xác định sự căn chỉnh dọc của hình chữ nhật bao của một thành phần.
- **static String WEST:** Xác định cạnh trái của hình chữ nhật bao của một thành phần.
- **static String WIDTH:** Xác định độ rộng của hình chữ nhật bao của một thành phần.

5.4.3.8.3 Ví dụ lớp `SpringLayout`

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import javax.swing.WindowConstants;

public class SpringLayoutExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
```

```
private JPanel controlPanel;

public SpringLayoutExam1() {
    prepareGUI();
}

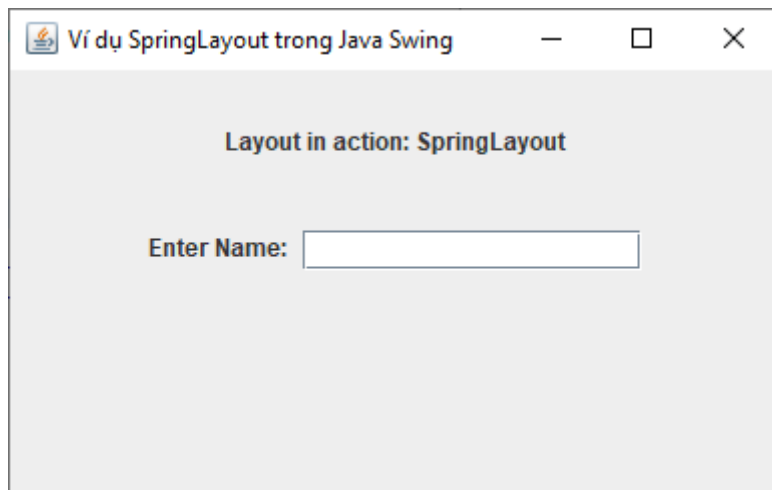
public static void main(String[] args) {
    SpringLayoutExam1 demo = new SpringLayoutExam1();
    demo.showSpringLayoutDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame();
    mainFrame.setSize(400, 250);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
    mainFrame.setTitle("Ví dụ SpringLayout trong Java Swing");
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showSpringLayoutDemo() {
    headerLabel.setText("Layout in action: SpringLayout");
    SpringLayout layout = new SpringLayout();
    JPanel panel = new JPanel();
    panel.setLayout(layout);
    JLabel label = new JLabel("Enter Name: ");
    JTextField textField = new JTextField("", 15);
    panel.add(label);
    panel.add(textField);
    layout.putConstraint(SpringLayout.WEST, label, 5,
        SpringLayout.WEST, controlPanel);
```

```
        layout.putConstraint(SpringLayout.NORTH, label, 5,  
SpringLayout.NORTH, controlPanel);  
  
        layout.putConstraint(SpringLayout.WEST, textField, 5,  
SpringLayout.EAST, label);  
  
        layout.putConstraint(SpringLayout.NORTH, textField, 5,  
SpringLayout.NORTH, controlPanel);  
  
        layout.putConstraint(SpringLayout.EAST, panel, 5,  
SpringLayout.EAST, textField);  
  
        layout.putConstraint(SpringLayout.SOUTH, panel, 5,  
SpringLayout.SOUTH, textField);  
  
        controlPanel.add(panel);  
        mainFrame.setVisible(true);  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.5 Xử lý sự kiện trong Java Swing

Nội dung chính

- Sự kiện (Event) là gì?
- Lớp EventObject trong Java Swing
- Các lớp Event trong Java Swing
 - Lớp AWTEvent
 - Lớp ActionEvent
 - Lớp InputEvent
 - Lớp KeyEvent
 - Lớp MouseEvent
 - Lớp WindowEvent
 - Lớp AdjustmentEvent
 - Lớp ComponentEvent
 - Lớp ContainerEvent
 - Lớp MouseMotionEvent
 - Lớp PaintEvent

5.5.1 Sự kiện (Event) là gì?

Event là một sự thay đổi trong trạng thái của đối tượng, chẳng hạn như sự kiện mô tả sự thay đổi trong trạng thái của source. Các sự kiện được tạo ra là do tương tác của người dùng với các thành phần UI. Ví dụ như việc nhấn vào một nút button, di chuyển chuột, nhập ký tự thông qua bàn phím, ... Các sự kiện có thể được phân chia thành hai loại sau:

Foreground Event: Những sự kiện này yêu cầu tương tác trực tiếp của người dùng. Chúng được tạo ra do tương tác của người dùng với các thành phần đồ họa trong Graphical User Interface. Ví dụ như nhấn vào một nút, di chuyển chuột, chọn một item từ list, ...

Background Event: Các sự kiện này yêu cầu tương tác của người dùng cuối cùng, ví dụ như tín hiệu ngắt hệ điều hành, hardware hoặc software failure ...

Xử lý sự kiện (Event Handling) là gì?

Xử lý sự kiện (Event Handling) là một kỹ thuật kiểm soát sự kiện và quyết định những gì cần thực hiện nếu một sự kiện xảy ra. Kỹ thuật này có code, mà được biết như là Event Handler, được thực thi khi một sự kiện xảy ra. Java sử dụng Delegation Event Model để xử lý các sự kiện. Model này định nghĩa kỹ thuật chuẩn để tạo và xử lý các sự kiện. Model này bao gồm hai thành phần quan trọng sau:

Source: đây là một đối tượng mà trên đó sự kiện xuất hiện. Source chịu trách nhiệm cung cấp thông tin về sự kiện đã xảy ra tới bộ xử lý Handler của nó.

Listener: Còn được biết như là Event Handler. Listener chịu trách nhiệm tạo phản hồi tới một sự kiện. Theo quan điểm của Java, Listener cũng là một đối

tượng. Listener đợi tới khi nó nhận một sự kiện. Khi một sự kiện đã được nhận, Listener xử lý sự kiện đó và sau đó trả về kết quả.

5.5.2 Lớp EventObject trong Java Swing

Đây là lớp gốc (root class) từ đó tất cả đối tượng về trạng thái sự kiện sẽ được kế thừa. Tất cả sự kiện được xây dựng với một tham chiếu tới đối tượng đó, là source. Lớp này được định nghĩa trong java.util package. Cú pháp khai báo của lớp java.util.EventObject như sau:

```
public class EventObject
    extends Object
        implements Serializable
```

Lớp EventObject có một trường là **protected Object source**. Đây là đối tượng mà trên đó sự kiện được xảy ra.

Lớp EventObject có một constructor là **EventObject(Object source)** dùng để xây dựng một nguyên mẫu sự kiện.

Các phương thức của lớp EventObject:

- **Object getSource()**: Đối tượng mà trên đó sự kiện xảy ra.
- **String toString()**: Trả về một biểu diễn chuỗi của EventObject này.

Lớp này kế thừa các phương thức từ lớp **java.lang.Object**.

5.5.3 Các lớp Event trong Java Swing

Ngoài lớp EventObject trên, phần dưới đây giới thiệu cho bạn một số Event Class được sử dụng phổ biến khác.

5.5.4 Lớp AWTEvent

Đây là lớp sự kiện gốc (root class) cho tất cả sự kiện AWTEvent. Lớp này và các lớp con của nó thay thế lớp ban đầu java.awt.Event. Lớp này được định nghĩa trong java.awt package. Lớp AWTEvent có phương thức getID() được sử dụng để xác định kiểu của sự kiện. Cú pháp khai báo của lớp java.awt.AWTEvent là:

```
public class AWTEvent
    extends EventObject
```

5.5.5 Lớp ActionEvent

Lớp ActionEvent được định nghĩa trong java.awt.event package. ActionEvent được tạo ra khi một nút được nhấn hoặc một item của một danh sách được nhấn đúp. Cú pháp khai báo cho lớp ActionEvent như sau:

```
public class ActionEvent
    extends AWTEvent
```


5.5.6 Lớp InputEvent

Lớp InputEvent là lớp sự kiện gốc (root class) cho tất cả sự kiện liên quan tới đầu vào (cấp độ thành phần). Các sự kiện liên quan tới đầu vào (input event) được phân phối bởi Listener trước khi chúng được xử lý một cách thông thường bởi source, nơi chúng sinh ra. Điều này cho phép các Listener và các lớp thành phần con có thể "consume" sự kiện để mà source sẽ không xử lý chúng theo phương thức mặc định. Cú pháp khai báo của lớp **java.awt.event.InputEvent** là:

```
public abstract class InputEvent
    extends ComponentEvent
```

5.5.7 Lớp KeyEvent

Sự kiện liên quan tới phím (Key Event) được tạo ra khi bạn nhập ký tự. Có ba kiểu key event, mà được biểu diễn bởi các hằng nguyên, chúng là:

- KEY_PRESSED
- KEY_RELEASED
- KEY_TYPED

Cú pháp khai báo của lớp **java.awt.event.KeyEvent** như sau:

```
public class KeyEvent
    extends InputEvent
```

5.5.8 Lớp MouseEvent

Sự kiện này chỉ một hoạt động liên quan tới chuột xảy ra trong một thành phần. Sự kiện tầm thấp này được tạo bởi một đối tượng Component cho các sự kiện liên quan tới chuột và di chuyển chuột, chẳng hạn như một nút chuột được nhấn, được nhả ra, được click (nhấn và nhả ra), di chuyển chuột, kéo chuột, ...

Cú pháp khai báo cho lớp **java.awt.event.MouseEvent**:

```
public class MouseEvent
    extends InputEvent
```

5.5.9 Lớp WindowEvent

Đối tượng của lớp này biểu diễn thay đổi về trạng thái của một cửa sổ. Sự kiện tầm thấp này được tạo bởi một đối tượng Window khi nó được mở, đóng, kích hoạt, ... hoặc khi trọng tâm focus được chuyển vào trong hoặc ngoài Window. Cú pháp khai báo của lớp **java.awt.event.WindowEvent** là:

```
1 public class WindowEvent
2     extends ComponentEvent
```

5.5.10 Lớp AdjustmentEvent

Lớp `AdjustmentEvent` biểu diễn các sự kiện liên quan tới sự căn chỉnh được tạo bởi các đối tượng có thể căn chỉnh (Adjustable object). Cú pháp khai báo của lớp `java.awt.event.AdjustmentEvent` là như sau:

```
public class AdjustmentEvent
    extends AWTEvent
```

5.5.11 Lớp `ComponentEvent`

Lớp `ComponentEvent` biểu diễn rằng một thành phần bị di chuyển, thay đổi kích cỡ, hoặc thay đổi tính nhìn thấy. Cú pháp khai báo của lớp `java.awt.event.ComponentEvent` là:

```
public class ComponentEvent
    extends AWTEvent
```

5.5.12 Lớp `ContainerEvent`

Lớp `ContainerEvent` biểu thị rằng các nội dung của một container đã thay đổi bởi vì một thành phần đã được thêm vào hoặc bị gỡ bỏ. Cú pháp khai báo của lớp `java.awt.event.ContainerEvent` là:

```
public class ContainerEvent
    extends ComponentEvent
```

5.5.13 Lớp `MouseEvent`

Lớp `MouseEvent` chỉ một hành động liên quan tới chuột (mouse action) đã xảy ra trong một thành phần. Sự kiện tầm thấp này được tạo bởi một đối tượng `Component` khi chuột bị kéo hoặc di chuyển. Cú pháp khai báo cho lớp `java.awt.event.MouseEvent` là:

```
public class MouseEvent
    extends InputEvent
```

5.5.14 Lớp `PaintEvent`

Lớp `PaintEvent` được sử dụng để bảo đảm rằng các lời gọi phương thức `paint/update` được xếp thứ tự theo các sự kiện khác được phân phối từ hàng sự kiện (event queue). Cú pháp khai báo cho lớp `java.awt.event.PaintEvent` là:

```
public class PaintEvent
    extends ComponentEvent
```

5.5.15 Event Listener trong Java Swing

Event Listener biểu diễn các giao diện chịu trách nhiệm xử lý các sự kiện. Java cung cấp các lớp Event Listener đa dạng, nhưng trong chương này chúng ta chỉ tập trung vào các lớp mà thường xuyên được sử dụng. Mỗi phương thức của

một Event Listener có một tham số đơn là một đối tượng mà là lớp con của lớp EventObject.

EventListener Interface là một marker interface mà mỗi listener phải kế thừa. Lớp này được định nghĩa trong java.util package.

5.5.15.1 Cú pháp để khai báo java.util.EventListener Interface là:

```
public interface EventListener
```

5.5.15.2 Các Event Listener Interface trong Java Swing

Bảng dưới liệt kê một số Event Listener được sử dụng phổ biến:

STT	Lớp & Mô tả
1	<u>ActionListener Interface</u> Interface này được sử dụng để nhận action event
2	<u>ComponentListener Interface</u> Interface này được sử dụng để nhận component event
3	<u>ItemListener Interface</u> Interface này được sử dụng để nhận item event
4	<u>KeyListener Interface</u> Interface này được sử dụng để nhận key event
5	<u>MouseListener Interface</u> Interface này được sử dụng để nhận mouse event
6	<u>WindowListener Interface</u> Interface này được sử dụng để nhận window event
7	<u>AdjustmentListener Interface</u> Interface này được sử dụng để nhận adjusmtent event
8	<u>ContainerListener Interface</u> Interface này được sử dụng để nhận container event
9	<u>MouseMotionListener Interface</u>

	Interface này được sử dụng để nhận mouse motion event
10	FocusListener Interface Interface này được sử dụng để nhận focus event

5.5.16 Event Adapter trong Java Swing

Adapter là các lớp abstract để nhận các sự kiện đa dạng. Các phương thức trong các lớp này là trống. Với các lớp này, việc tạo các đối tượng Listener trở nên thuận tiện hơn. Bảng dưới đây liệt kê một số adapter được sử dụng phổ biến trong khi nghe các GUI event trong Java Swing.

STT	Adapter & Mô tả
1	Lớp FocusAdapter Một lớp abstract để nhận các focus event.
2	Lớp KeyAdapter Một lớp abstract để nhận các key event
3	Lớp MouseAdapter Một lớp abstract để nhận các mouse event
4	Lớp MouseMotionAdapter Một lớp abstract để nhận các mouse motion event
5	Lớp WindowAdapter Một lớp abstract để nhận các window event

5.5.17 ActionListener trong Java Swing

Lớp xử lý ActionEvent nên triển khai interface ActionListener. Đối tượng của lớp đó phải được đăng ký với một thành phần. Đối tượng có thể được đăng ký bởi sử dụng phương thức addActionListener(). Khi action event xảy ra, phương thức actionPerformed() của đối tượng đó được gọi.

Cú pháp khai báo cho java.awt.event.ActionListener interface là:

```
public interface ActionListener
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp **java.awt.EventListener**.

Phương thức của ActionListener trong Java Swing:

- **void actionPerformed(ActionEvent e):** Được triệu hồi khi một action xuất hiện.
-

Ví dụ ActionListener

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class ActionListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public ActionListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        ActionListenerExam1 demo = new ActionListenerExam1();
        demo.showActionListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
```

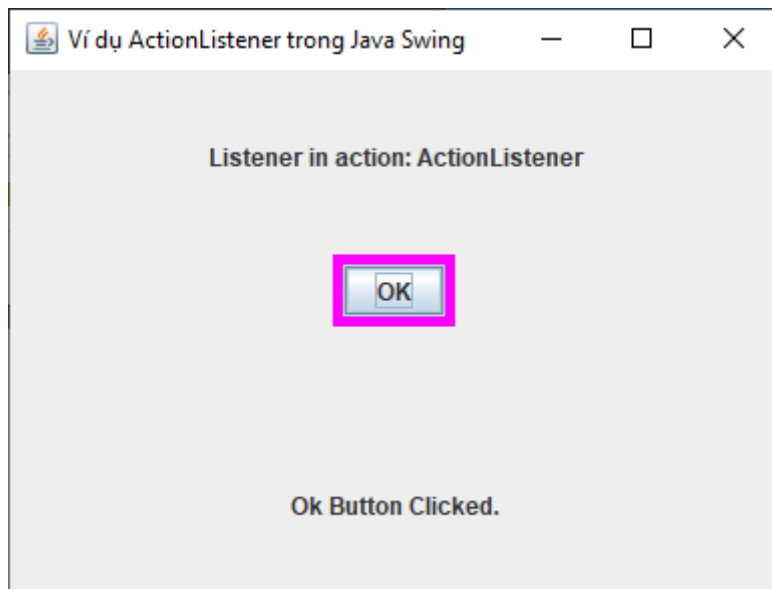
```
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ ActionListener trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_C
LOSE);
    }

    private void showActionListenerDemo() {
        headerLabel.setText("Listener in action: ActionListener");
        JPanel panel = new JPanel();
        panel.setBackground(Color.magenta);

        JButton okButton = new JButton("OK");
        okButton.addActionListener(new CustomActionListener());
        panel.add(okButton);
        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }

    class CustomActionListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            statusLabel.setText("Ok Button Clicked.");
        }
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.18 ComponentListener trong Java Swing

Lớp xử lý ComponentEvent nên triển khai interface EventListener. Đối tượng của lớp đó phải được đăng ký với một thành phần. Đối tượng có thể được đăng ký bởi sử dụng phương thức addComponentListener(). Cú pháp khai báo cho java.awt.event.ComponentListener interface như sau:

```
public interface ComponentListener
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.18.1 Các phương thức của ComponentListener Interface trong Java Swing

void componentHidden(ComponentEvent e): Được gọi khi thành phần trở nên không nhìn thấy.

void componentMoved(ComponentEvent e): Được gọi khi vị trí của thành phần thay đổi.

void componentResized(ComponentEvent e): Được gọi khi kích cỡ của thành phần thay đổi.

void componentShown(ComponentEvent e): Được gọi khi thành phần trở nên nhìn thấy.

5.5.18.2 Ví dụ ComponentListener

```
package vn.plpsoft.swing;
```

```
import java.awt.Color;
```

```
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ComponentEvent;
import java.awt.event.ComponentListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class ComponentListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public ComponentListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        ComponentListenerExam1 demo = new ComponentListenerExam1();
        demo.showComponentListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ ActionListener trong Java Swing");
    }
}
```



```
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_C
LOSE);
    }

    private void showComponentListenerDemo() {
        headerLabel.setText("Listener: ComponentListener");

        JPanel panel = new JPanel();
        panel.setBackground(Color.magenta);
        JLabel msglabel = new JLabel("Vi du ComponentListener trong
Java Swing.", JLabel.CENTER);
        panel.add(msglabel);

        msglabel.addComponentListener(new CustomComponentListener());
        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }

    class CustomComponentListener implements ComponentListener {

        public void componentResized(ComponentEvent e) {
            statusLabel.setText(statusLabel.getText()
+ e.getComponent().getClass().getSimpleName() + " resized. ");
        }

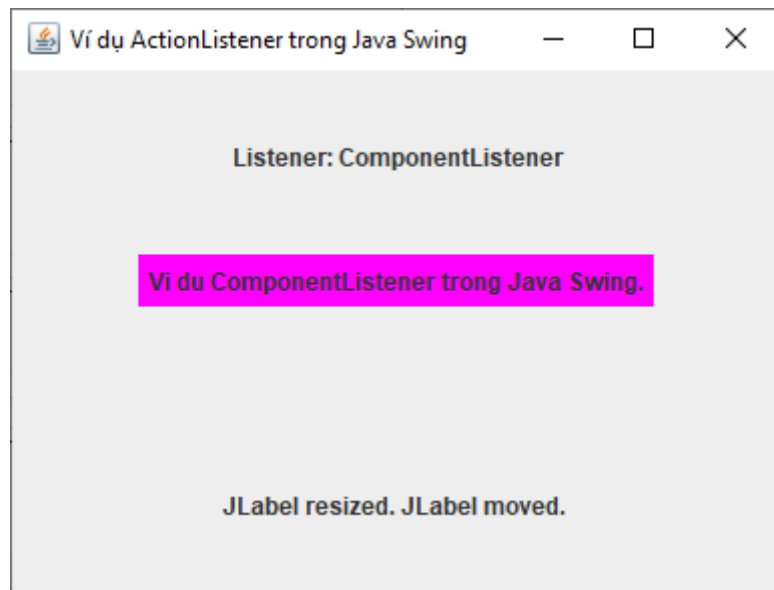
        public void componentMoved(ComponentEvent e) {
            statusLabel.setText(statusLabel.getText()
+ e.getComponent().getClass().getSimpleName() + " moved. ");
        }

        public void componentShown(ComponentEvent e) {
            statusLabel.setText(statusLabel.getText()
+ e.getComponent().getClass().getSimpleName() + " shown. ");
        }

        public void componentHidden(ComponentEvent e) {
            statusLabel.setText(statusLabel.getText()
+ e.getComponent().getClass().getSimpleName() + " hidden. ");
        }
    }
}
```

```
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.19 ItemListener trong Java Swing

Lớp xử lý ItemEvent nên triển khai interface ItemListener. Đối tượng của lớp đó phải được đăng ký với một thành phần. Đối tượng có thể được đăng ký bởi sử dụng phương thức addItemListener(). Khi một action event xảy ra, phương thức itemStateChanged() của đối tượng đó được triệu hồi. Cú pháp khai báo cho java.awt.event.ItemListener interface như sau:

```
public interface ItemListener
    extends EventListener
```

Interface ItemListener kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.19.1 Phương thức của interface ItemListener trong Java Swing

void itemStateChanged(ItemEvent e): Được triệu hồi khi một item đã được lựa chọn (selected) hoặc bị loại bỏ (deselected) bởi người dùng.

5.5.19.2 Ví dụ ItemListener

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.WindowAdapter;
```

```
import java.awt.event.WindowEvent;

import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class ItemListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public ItemListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        ItemListenerExam1 demo = new ItemListenerExam1();
        demo.showItemListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame();
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
    }
}
```

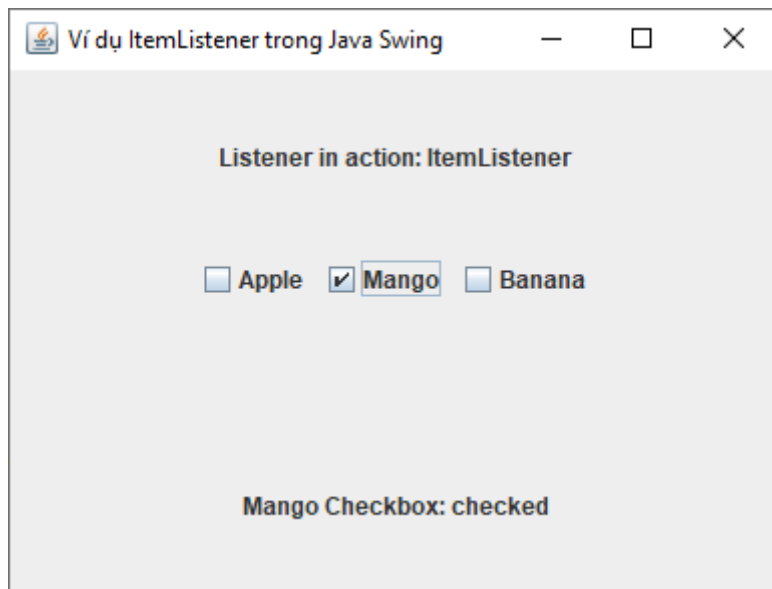
```
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
mainFrame.setTitle("Ví dụ ItemListener trong Java Swing");
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_
CLOSE);
}

private void showItemListenerDemo() {
    headerLabel.setText("Listener in action: ItemListener");
    JCheckBox chkApple = new JCheckBox("Apple");
    JCheckBox chkMango = new JCheckBox("Mango");
    JCheckBox chBanana = new JCheckBox("Banana");

    chkApple.addItemListener(new CustomItemListener());
    chkMango.addItemListener(new CustomItemListener());
    chBanana.addItemListener(new CustomItemListener());
    controlPanel.add(chkApple);
    controlPanel.add(chkMango);
    controlPanel.add(chBanana);
    mainFrame.setVisible(true);
}

class CustomItemListener implements ItemListener {
    public void itemStateChanged(ItemEvent e) {
        statusLabel.setText(((JCheckBox) e.getItem()).getText()
            + " Checkbox: "
            + (e.getStateChange() == 1 ? "checked" :
"unchecked"));
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.20 KeyListener trong Java Swing

Lớp xử lý KeyEvent nên triển khai interface KeyListener. Đối tượng của lớp đó phải được đăng ký với một thành phần. Đối tượng có thể được đăng ký bởi sử dụng phương thức addKeyListener(). Cú pháp khai báo cho java.awt.event.KeyListener interface như sau:

```
public interface KeyListener
    extends EventListener
```

Interface KeyListener kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.20.1 Các phương thức của KeyListener Interface trong Java Swing

1. **void keyPressed(KeyEvent e)**: Được triệu hồi khi một key đã được nhấn.
2. **void keyReleased(KeyEvent e)**: Được triệu hồi khi một key đã được nhả ra.
3. **void keyTyped(KeyEvent e)**: Được triệu hồi khi một key đã được gõ.

5.5.20.2 Ví dụ KeyListener

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.WindowAdapter;
```

```
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

public class KeyListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public KeyListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        KeyListenerExam1 demo = new KeyListenerExam1();
        demo.showKeyListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame();
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
```

```
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ KeyListener trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON
_CLOSE);
    }

    private JTextField textField;

    private void showKeyListenerDemo() {
        headerLabel.setText("Listener in action: KeyListener");
        textField = new JTextField(10);
        textField.addKeyListener(new CustomKeyListener());
        JButton okButton = new JButton("OK");
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                statusLabel.setText("Entered text: " +
textField.getText());
            }
        });
        controlPanel.add(textField);
        controlPanel.add(okButton);
        mainFrame.setVisible(true);
    }

    class CustomKeyListener implements KeyListener {
        public void keyTyped(KeyEvent e) {

        }

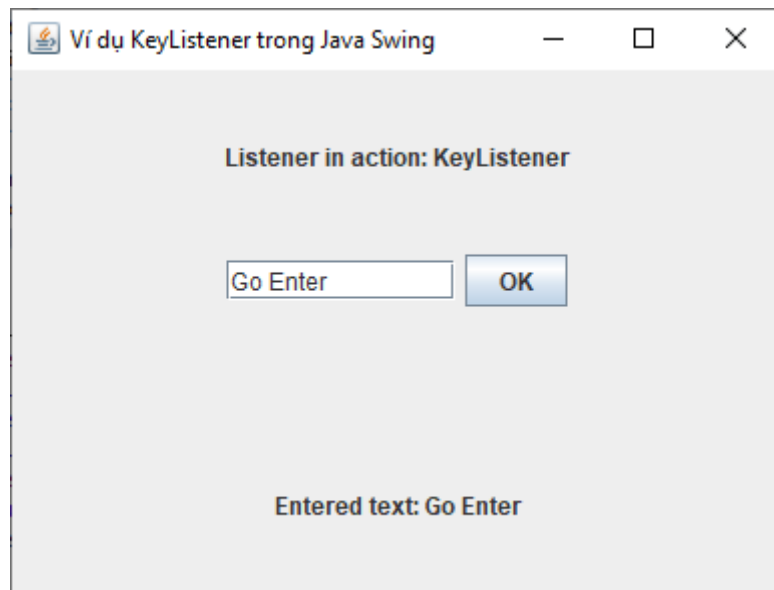
        public void keyPressed(KeyEvent e) {
            if (e.getKeyCode() == KeyEvent.VK_ENTER) {
                statusLabel.setText("Entered text: " +
textField.getText());
            }
        }

        public void keyReleased(KeyEvent e) {

        }
    }
```

```
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.21 MouseListener trong Java Swing

Lớp xử lý MouseEvent nên triển khai interface MouseListener. Đối tượng của lớp đó phải được đăng ký với một thành phần. Đối tượng có thể được đăng ký bởi sử dụng phương thức addMouseListener(). Cú pháp khai báo cho java.awt.event.MouseListener interface như sau:

```
public interface MouseListener
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.21.1 Các phương thức của MouseListener Interface trong Java Swing

1. **void mouseClicked(MouseEvent e):** Được triệu hồi khi nút chuột đã được click (được nhấn và nhả ra) trên một thành phần.
2. **void mouseEntered(MouseEvent e):** Được triệu hồi khi chuột nhập vào một thành phần.
3. **void mouseExited(MouseEvent e):** Được triệu hồi khi chuột thoát ra khỏi một thành phần.
4. **void mousePressed(MouseEvent e):** Được triệu hồi khi một nút chuột đã được nhấn trên một thành phần.
5. **void mouseReleased(MouseEvent e):** Được triệu hồi khi một nút chuột đã được nhả ra trên một thành phần.

5.5.21.2 Ví dụ MouseListener


```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class MouseListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public MouseListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        MouseListenerExam1 demo = new MouseListenerExam1();
        demo.showMouseListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
    }
}
```

```
        }

    });

    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
}

private void showMouseListenerDemo() {
    headerLabel.setText("Listener in action: MouseListener");
    JPanel panel = new JPanel();
    panel.setBackground(Color.magenta);
    panel.setLayout(new FlowLayout());
    panel.addMouseListener(new CustomMouseListener());
    JLabel msglabel = new JLabel("Click vào đây để lấy tọa độ.",
        JLabel.CENTER);
    panel.add(msglabel);
    msglabel.addMouseListener(new CustomMouseListener());
    panel.add(msglabel);
    controlPanel.add(panel);
    mainFrame.setVisible(true);
}

class CustomMouseListener implements MouseListener {
    public void mouseClicked(MouseEvent e) {
        statusLabel.setText("Mouse Clicked: "
            + "(" + e.getX() + ", " + e.getY() + ")");
    }

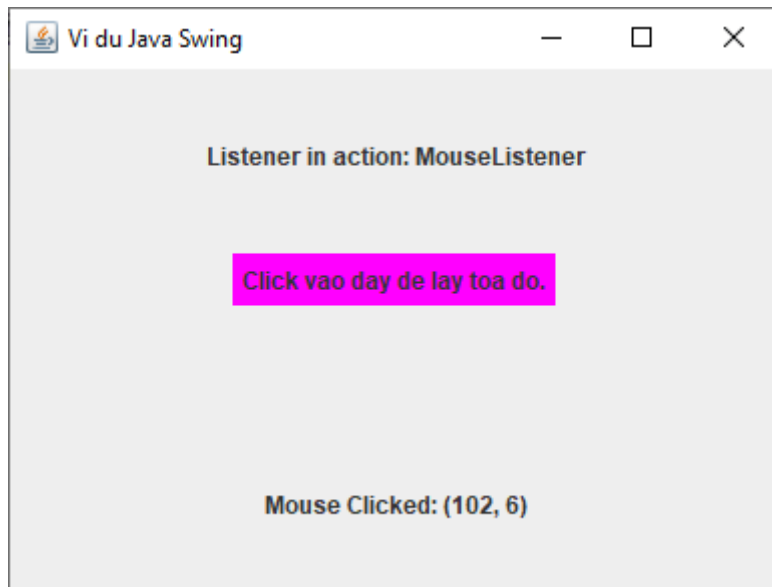
    public void mousePressed(MouseEvent e) {
    }

    public void mouseReleased(MouseEvent e) {
    }

    public void mouseEntered(MouseEvent e) {
    }
}
```

```
        public void mouseExited(MouseEvent e) {  
        }  
    }  
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.22 WindowListener trong Java Swing

Lớp mà xử lý WindowEvent nên triển khai interface EventListener. Đối tượng của lớp đó phải được đăng ký với một thành phần. Đối tượng có thể được đăng ký bởi sử dụng phương thức addWindowListener(). Cú pháp khai báo cho java.awt.event.WindowListener interface như sau:

```
public interface WindowListener  
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.22.1 Các phương thức của MouseListener Interface trong Java Swing

- 1. void windowActivated(WindowEvent e):** Được triệu hồi khi Window được thiết lập là active.
- 2. void windowClosed(WindowEvent e):** Được triệu hồi khi một cửa sổ đã được đóng.
- 3. void windowClosing(WindowEvent e):** Được triệu hồi khi người dùng cố gắng đóng cửa sổ từ system menu của Window.
- 4. void windowDeactivated(WindowEvent e):** Được triệu hồi khi một Window không còn là active nữa.

5. void windowDeiconified(WindowEvent e): Được triệu hồi khi một cửa sổ đã bị thay đổi từ kích cỡ tối thiểu tới trạng thái thường.

6. void windowIconified(WindowEvent e): Được triệu hồi khi một cửa sổ đã bị thay đổi từ trạng thái thường về kích cỡ tối thiểu.

7. void windowOpened(WindowEvent e): Được triệu hồi khi lần đầu tiên một cửa sổ trở nên nhìn thấy.

5.5.22.2 Ví dụ WindowListener

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class WindowEventExam1 {
    private JFrame mainFrame;
    private JFrame aboutFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public WindowEventExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        WindowEventExam1 demo = new WindowEventExam1();
        demo.showWindowListenerDemo();
    }

    private void prepareGUI() {
```

```
mainFrame = new JFrame();
mainFrame.setSize(400, 300);
mainFrame.setLayout(new GridLayout(3, 1));
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setSize(350, 100);
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);
mainFrame.setTitle("Ví dụ WindowListener trong Java Swing");
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CL
OSE);
}
```

```
private void showWindowListenerDemo() {
    headerLabel.setText("Listener in action: WindowListener");
    aboutFrame = new JFrame();
    aboutFrame.setSize(300, 200);
    aboutFrame.setTitle("WindowListener Demo");
    aboutFrame.addWindowListener(new CustomWindowListener());

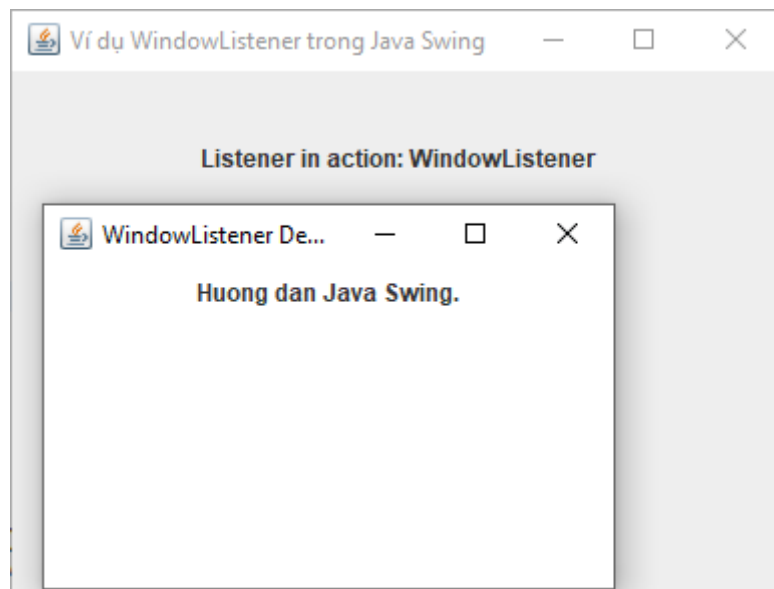
    JPanel panel = new JPanel();
    panel.setBackground(Color.white);
    JLabel msglabel = new JLabel("Huong dan Java Swing.",
        JLabel.CENTER);
    panel.add(msglabel);
    aboutFrame.add(panel);
    aboutFrame.setVisible(true);
}
```

```
class CustomWindowListener implements WindowListener {
    public void windowOpened(WindowEvent e) {
    }

    public void windowClosing(WindowEvent e) {
        aboutFrame.dispose();
    }
}
```

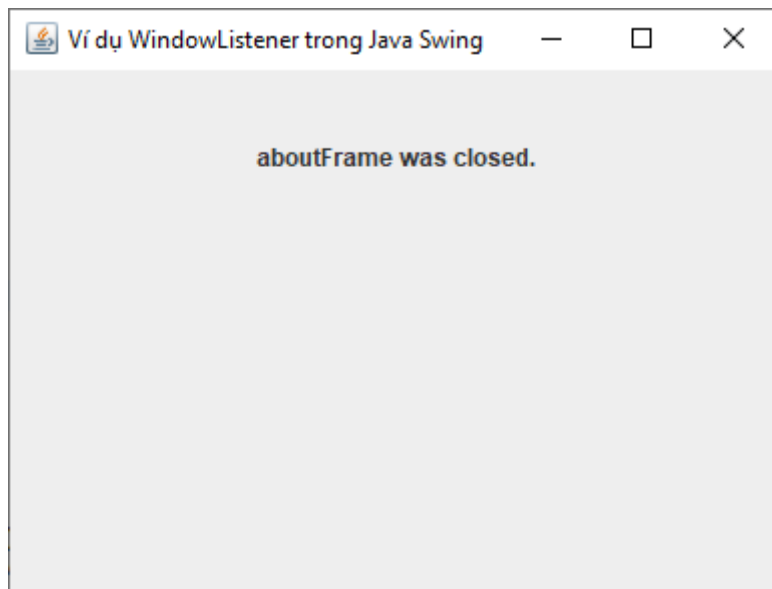
```
        headerLabel.setText("aboutFrame was closed.");  
    }  
  
    public void windowClosed(WindowEvent e) {  
    }  
  
    public void windowIconified(WindowEvent e) {  
    }  
  
    public void windowDeiconified(WindowEvent e) {  
    }  
  
    public void windowActivated(WindowEvent e) {  
    }  
  
    public void windowDeactivated(WindowEvent e) {  
    }  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



Close About Frame:

Chạy chương trình trên cho kết quả như sau:



5.5.23 AdjustmentListener trong Java Swing

AdjustmentListener được sử dụng để nhận các sự kiện liên quan tới căn chỉnh. Lớp mà xử lý adjustment event cần triển khai interface này. Cú pháp để khai báo `java.awt.event.AdjustmentListener` là:

```
public interface AdjustmentListener
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp `java.awt.EventListener`.

5.5.23.1 Các phương thức của AdjustmentListener Interface trong Java Swing

1. void adjustmentValueChanged(AdjustmentEvent e): Được triệu hồi khi giá trị có thể điều chỉnh đã thay đổi.

5.5.23.2 Ví dụ AdjustmentListener

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.AdjustmentEvent;
import java.awt.event.AdjustmentListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
```

```
import javax.swing.JScrollBar;
import javax.swing.WindowConstants;

public class AdjustmentListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public AdjustmentListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        AdjustmentListenerExam1 demo = new
AdjustmentListenerExam1();
        demo.showAdjustmentListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame();
        mainFrame.setSize(450, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
mainFrame.setTitle("Ví dụ AdjustmentListener trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_O
N_CLOSE);
    }

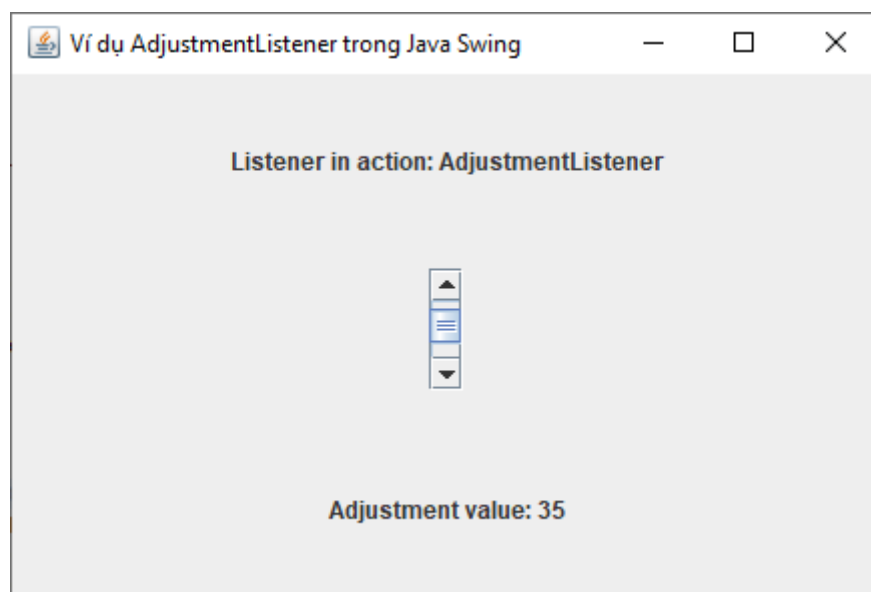
    private void showAdjustmentListenerDemo() {
        headerLabel.setText("Listener in action:
AdjustmentListener");
    }
}
```



```
JPanel panel = new JPanel();
JScrollBar scrollbar = new JScrollBar();
scrollbar.addAdjustmentListener(new
CustomAdjustmentListener());
panel.add(scrollbar);
controlPanel.add(panel);
mainFrame.setVisible(true);
}

class CustomAdjustmentListener implements AdjustmentListener {
    public void adjustmentValueChanged(AdjustmentEvent e) {
        statusLabel.setText("Adjustment value: " +
            Integer.toString(e.getValue()));
    }
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.24 ContainerListener trong Java Swing

ContainerListener Interface được sử dụng để nhận các sự kiện liên quan tới container. Lớp mà xử lý container event cần triển khai interface này. Cú pháp để khai báo `java.awt.event.ContainerListener` là:

```
public interface ContainerListener
extends EventListener
```

Interface này kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.24.1 Các phương thức của **MouseListener** Interface trong **Java Swing**

1. void componentAdded(ContainerEvent e): Được triệu hồi khi một thành phần đã được thêm vào container.

2. void componentRemoved(ContainerEvent e): Được triệu hồi khi một thành phần đã bị xóa khỏi container.

5.5.24.2 Ví dụ **ContainerListener**

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ContainerEvent;
import java.awt.event.ContainerListener;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class ContainerListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public ContainerListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        ContainerListenerExam1 demo = new ContainerListenerExam1();
        demo.showContainerListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame();
        mainFrame.setSize(450, 300);
```

```
mainFrame.setLayout(new GridLayout(3, 1));
headerLabel = new JLabel("", JLabel.CENTER);
statusLabel = new JLabel("", JLabel.CENTER);
statusLabel.setSize(350, 100);
controlPanel = new JPanel();
controlPanel.setLayout(new FlowLayout());
mainFrame.add(headerLabel);
mainFrame.add(controlPanel);
mainFrame.add(statusLabel);
mainFrame.setVisible(true);

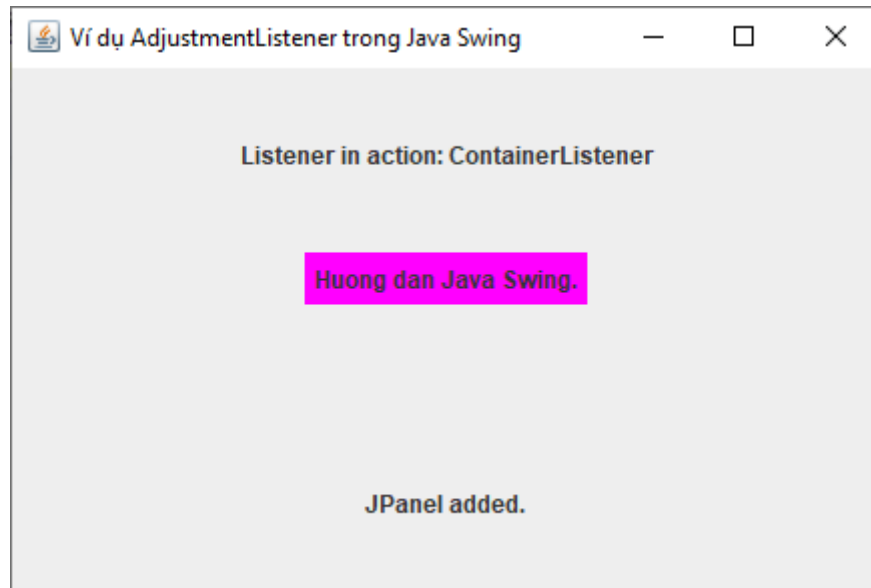
mainFrame.setTitle("Ví dụ AdjustmentListener trong Java Swing");
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showContainerListenerDemo() {
    headerLabel.setText("Listener in action: ContainerListener");
    JPanel panel = new JPanel();
    panel.setBackground(Color.magenta);
    panel.addContainerListener(new CustomContainerListener());
    JLabel msglabel = new JLabel("Huong dan Java Swing.",
                                JLabel.CENTER);
    panel.add(msglabel);
    controlPanel.add(panel);
    mainFrame.setVisible(true);
}

class CustomContainerListener implements ContainerListener {
    public void componentAdded(ContainerEvent e) {
        statusLabel.setText(statusLabel.getText() +
            e.getComponent().getClass().getSimpleName() + " added. ");
    }

    public void componentRemoved(ContainerEvent e) {
        statusLabel.setText(statusLabel.getText() +
            e.getComponent().getClass().getSimpleName() + " removed. ");
    }
}
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.25 MouseMotionListener trong Java Swing

MouseMotionListener Interface được sử dụng để nhận các sự kiện liên quan tới di chuyển chuột trên một thành phần. Lớp mà xử lý các sự kiện này cần triển khai interface này. Cú pháp để khai báo `java.awt.event.MouseMotionListener` interface là:

```
public interface MouseMotionListener
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp `java.awt.EventListener`.

5.5.25.1 Các phương thức của MouseListener Interface trong Java Swing

1. void mouseDragged(MouseEvent e): Được triệu hồi khi một nút chuột được nhấn trên một thành phần và sau đó được kéo (drag).

2. void mouseMoved(MouseEvent e): Được triệu hồi khi con trỏ chuột đã được di chuyển trên một thành phần nhưng không có nút nào được nhấn.

5.5.25.2 Ví dụ MouseMotionListener

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
```

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class MouseMotionListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public MouseMotionListenerExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        MouseMotionListenerExam1 demo = new
        MouseMotionListenerExam1();
        demo.showMouseMotionListenerDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(450, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
```

```
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ MouseMotionListener trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_C
LOSE);
    }

    private void showMouseMotionListenerDemo() {
        headerLabel.setText("Listener in action:
MouseMotionListener");
        JPanel panel = new JPanel();
        panel.setBackground(Color.magenta);
        panel.setLayout(new FlowLayout());
        panel.addMouseMotionListener(new
CustomMouseMotionListener());
        JLabel msglabel = new JLabel("Huong dan Java Swing.",
JLabel.CENTER);
        panel.add(msglabel);
        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }

    class CustomMouseMotionListener implements MouseMotionListener {
        public void mouseDragged(MouseEvent e) {
            statusLabel.setText("Mouse Dragged: "
+ "(" + e.getX() + ", " + e.getY() + ")");
        }

        public void mouseMoved(MouseEvent e) {
            statusLabel.setText("Mouse Moved: "
+ "(" + e.getX() + ", " + e.getY() + ")");
        }
    }
}
```

Chạy chương trình trên cho kết quả như sau:

5.5.26 FocusListener trong Java Swing

FocusListener Interface được sử dụng để nhận sự kiện liên quan tới keyboard focus. Lớp mà triển khai focus event cần triển khai interface này. Cú pháp để khai báo java.awt.event.FocusListener Interface là:

```
public interface FocusListener
    extends EventListener
```

Interface này kế thừa các phương thức từ lớp **java.awt.EventListener**.

5.5.26.1 Các phương thức của MouseListener Interface trong Java Swing

1. void focusGained(FocusEvent e): Được triệu hồi khi một thành phần nhận keyboard focus.

2. void focusLost(FocusEvent e): Được triệu hồi khi một thành phần mất keyboard focus.

5.5.26.2 Ví dụ FocusListener

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class FocusListenerExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public FocusListenerExam1() {
        prepareGUI();
    }
}
```

```
public static void main(String[] args) {
    FocusListenerExam1 demo = new FocusListenerExam1();
    demo.showFocusListenerDemo();
}

private void prepareGUI() {
    mainFrame = new JFrame("Vi du Java Swing");
    mainFrame.setSize(500, 300);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);

    mainFrame.setTitle("Ví dụ FocusListener trong Java Swing");
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}

private void showFocusListenerDemo() {
    headerLabel.setText("Listener in action: FocusListener");
    JButton okButton = new JButton("OK");
    JButton cancelButton = new JButton("Cancel");

    okButton.addFocusListener(new CustomFocusListener());
    cancelButton.addFocusListener(new CustomFocusListener());

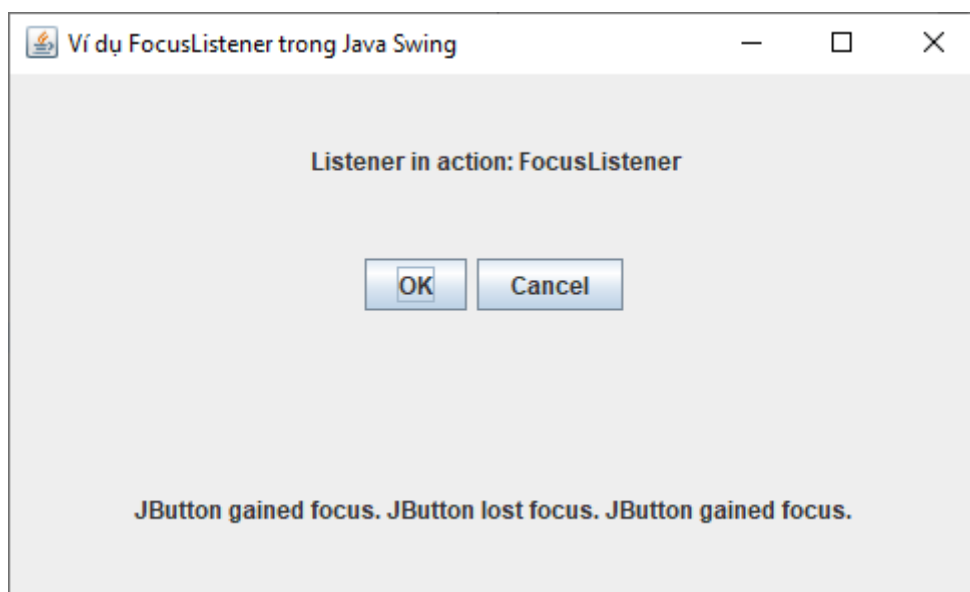
    controlPanel.add(okButton);
    controlPanel.add(cancelButton);
    mainFrame.setVisible(true);
}

class CustomFocusListener implements FocusListener {
    public void focusGained(FocusEvent e) {
        statusLabel.setText(statusLabel.getText()
            + e.getComponent().getClass().getSimpleName())
    }
}
```



```
        + " gained focus. ");  
    }  
  
    public void focusLost(FocusEvent e) {  
        statusLabel.setText(statusLabel.getText()  
            + e.getComponent().getClass().getSimpleName()  
            + " lost focus. ");  
    }  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.27 Lớp FocusAdapter trong Java Swing

Lớp FocusAdapter là một lớp abstract để nhận các sự kiện keyboard event. Tất cả phương thức của lớp này là trống. Với lớp này, việc tạo các đối tượng Listener trở nên khá thuận tiện. Cú pháp khai báo cho lớp `java.awt.event.FocusAdapter` là:

```
public abstract class FocusAdapter  
    extends Object  
        implements FocusListener
```

Lớp này kế thừa các phương thức từ lớp `java.lang.Object`.

Lớp FocusAdapter có một constructor là: **FocusAdapter()**.

5.5.27.1 Phương thức của lớp FocusAdapter trong Java Swing

void focusGained(FocusEvent e): Được triệu hồi khi một thành phần nhận keyboard focus.

5.5.27.2 Ví dụ FocusAdapter

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class FocusAdapterExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public FocusAdapterExam1 () {
        prepareGUI ();
    }

    public static void main(String[] args) {
        FocusAdapterExam1 demo = new FocusAdapterExam1 ();
        demo.showFocusAdapterDemo ();
    }

    private void prepareGUI () {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(500, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
    }
}
```

```
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);

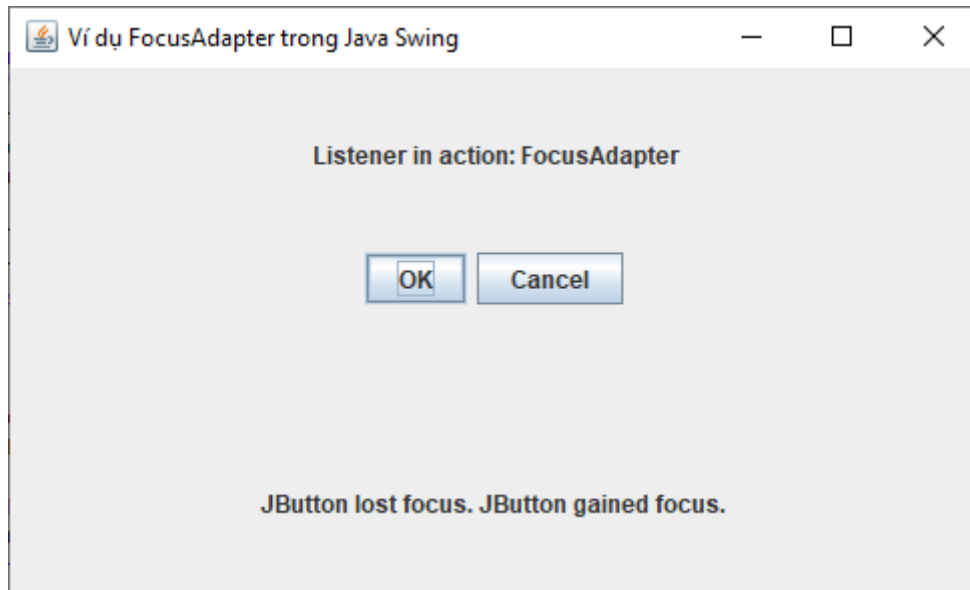
        mainFrame.setTitle("Ví dụ FocusAdapter trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    private void showFocusAdapterDemo() {
        headerLabel.setText("Listener in action: FocusAdapter");
        JButton okButton = new JButton("OK");
        JButton cancelButton = new JButton("Cancel");
        okButton.addFocusListener(new FocusAdapter() {
            public void focusGained(FocusEvent e) {
                statusLabel.setText(
                    statusLabel.getText()
                    +
                    e.getComponent().getClass().getSimpleName()
                    + " gained focus. ");
            }
        });

        cancelButton.addFocusListener(new FocusAdapter() {
            public void focusLost(FocusEvent e) {
                statusLabel
                    .setText(statusLabel.getText()
                    +
                    e.getComponent().getClass().getSimpleName()
                    + " lost focus. ");
            }
        });

        controlPanel.add(okButton);
        controlPanel.add(cancelButton);
        mainFrame.setVisible(true);
    }
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.28 Lớp KeyAdapter trong Java Swing

Lớp KeyAdapter là một lớp abstract để nhận các keyboard event. Tất cả phương thức của lớp này là trống. Với lớp này, việc tạo các đối tượng Listener trở nên khá thuận tiện. Cú pháp khai báo cho lớp java.awt.event.KeyAdapter là:

```
public abstract class KeyAdapter
    extends Object
        implements KeyListener
```

Lớp này kế thừa các phương thức từ lớp **java.lang.Object**.

Lớp KeyAdapter có một constructor là: **KeyAdapter()**.

5.5.28.1 Phương thức của lớp KeyAdapter trong Java Swing

1. **void keyPressed(KeyEvent e):** Được triệu hồi khi một key đã được nhấn.
2. **void keyReleased(KeyEvent e):** Được triệu hồi khi một key đã được nhả ra.
3. **void keyTyped(KeyEvent e):** Được triệu hồi khi một key đã được gõ.

5.5.28.2 Ví dụ KeyAdapter

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

```
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

public class KeyAdapterExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

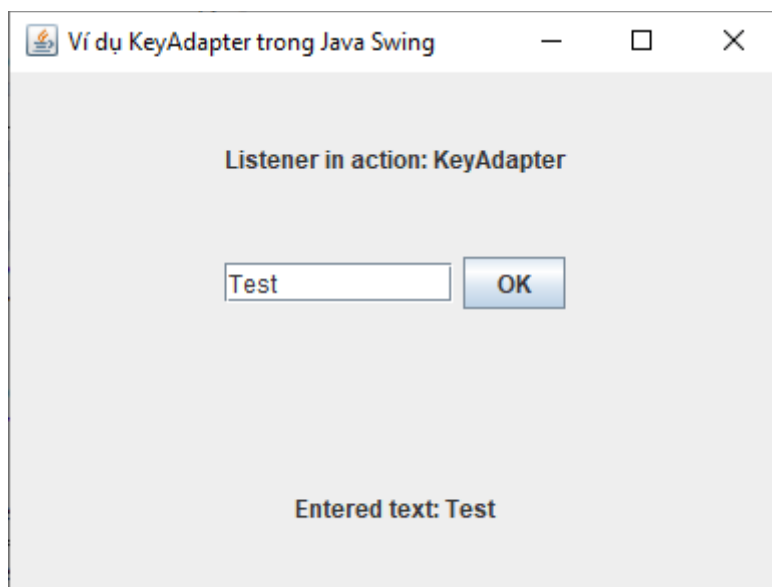
    public KeyAdapterExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        KeyAdapterExam1 demo = new KeyAdapterExam1();
        demo.showKeyAdapterDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(400, 300);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ KeyAdapter trong Java Swing");
    }
}
```

```
mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
}  
  
private void showKeyAdapterDemo() {  
    headerLabel.setText("Listener in action: KeyAdapter");  
    final JTextField textField = new JTextField(10);  
    textField.addKeyListener(new KeyAdapter() {  
        public void keyPressed(KeyEvent e) {  
            if (e.getKeyCode() == KeyEvent.VK_ENTER) {  
                statusLabel.setText("Entered text: " +  
textField.getText());  
            }  
        }  
    });  
    JButton okButton = new JButton("OK");  
    okButton.addActionListener(new ActionListener() {  
        public void actionPerformed(ActionEvent e) {  
            statusLabel.setText("Entered text: " +  
textField.getText());  
        }  
    });  
    controlPanel.add(textField);  
    controlPanel.add(okButton);  
    mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.29 Lớp MouseAdapter trong Java Swing

Lớp MouseAdapter là một lớp abstract để nhận các keyboard event. Tất cả phương thức của lớp này là trống. Với lớp này, việc tạo các đối tượng Listener trở nên khá thuận tiện. Cú pháp khai báo cho lớp java.awt.event.MouseAdapter là:

```
public abstract class MouseAdapter
    extends Object
        implements MouseListener, MouseWheelListener, MouseMotionListener
```

Lớp này kế thừa các phương thức từ lớp **java.lang.Object**.

Lớp MouseAdapter có một constructor là: **MouseListener()**.

5.5.29.1 Phương thức của lớp MouseAdapter trong Java Swing

1. **void mouseClicked(MouseEvent e)**: Được triệu hồi khi nút chuột đã được click (được nhấn và nhả ra) trên một thành phần.
 2. **void mouseDragged(MouseEvent e)**: Được triệu hồi khi nút chuột đã được nhấn trên một thành phần và sau đó được kéo (drag).
 3. **void mouseEntered(MouseEvent e)**: Được triệu hồi khi chuột nhập vào một thành phần.
 4. **void mouseExited(MouseEvent e)**: Được triệu hồi khi chuột thoát khỏi một thành phần.
 5. **void mouseMoved(MouseEvent e)**: Được triệu hồi khi con trỏ chuột đã được di chuyển trên một thành phần nhưng không có nút nào được nhấn.
 6. **void mousePressed(MouseEvent e)**: Được triệu hồi khi nút chuột đã được nhấn trên một thành phần.
 7. **void mouseReleased(MouseEvent e)**: Được triệu hồi khi nút chuột đã được giải phóng trên một thành phần.
 8. **void mouseWheelMoved(MouseWheelEvent e)**: Được triệu hồi khi bánh xe chuột được quay.
-

5.5.29.2 Ví dụ MouseAdapter

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class MouseAdapterExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public MouseAdapterExam1() {
        prepareGUI();
    }

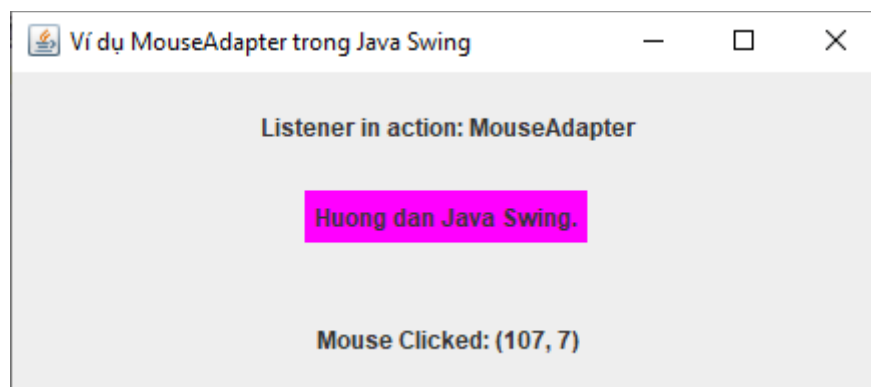
    public static void main(String[] args) {
        MouseAdapterExam1 demo = new MouseAdapterExam1();
        demo.showMouseAdapterDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Vi du Java Swing");
        mainFrame.setSize(450, 200);
        mainFrame.setLayout(new GridLayout(3, 1));
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);
        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
        mainFrame.setTitle("Ví dụ MouseAdapter trong Java Swing");
        mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }
}
```



```
private void showMouseAdapterDemo() {  
    headerLabel.setText("Listener in action:  
MouseAdapter");  
    JPanel panel = new JPanel();  
    panel.setBackground(Color.magenta);  
    panel.setLayout(new FlowLayout());  
    panel.addMouseListener(new MouseAdapter() {  
        public void mouseClicked(MouseEvent e) {  
            statusLabel.setText("Mouse Clicked: "  
                + "(" + e.getX() + ", " + e.getY() + ")");  
        }  
    });  
    JLabel msglabel = new JLabel("Huong dan Java Swing.",  
JLabel.CENTER);  
  
    msglabel.addMouseListener(new MouseAdapter() {  
        public void mouseClicked(MouseEvent e) {  
            statusLabel.setText("Mouse Clicked: "  
                + "(" + e.getX() + ", " + e.getY() + ")");  
        }  
    });  
    panel.add(msglabel);  
    controlPanel.add(panel);  
    mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.30 Lớp MouseMotionAdapter trong Java Swing

Lớp MouseMotionAdapter là một lớp abstract để nhận các keyboard event. Tất cả phương thức của lớp này là trống. Với lớp này, việc tạo các đối tượng

Listener trở nên khá thuận tiện. Cú pháp khai báo cho lớp `java.awt.event.MouseMotionAdapter` là:

```
public abstract class MouseMotionAdapter
    extends Object
    implements MouseMotionListener
```

Lớp này kế thừa các phương thức từ lớp **`java.lang.Object`**.

Lớp `MouseMotionAdapter` có một constructor là: **`MouseMotionAdapter()`**.

5.5.30.1 Phương thức của lớp `MouseMotionAdapter` trong Java Swing

1. `void mouseDragged(MouseEvent e)`: Được triệu hồi khi một nút chuột được nhấn trên một thành phần và sau đó được kéo (drag).

2. `void mouseMoved(MouseEvent e)`: Được triệu hồi khi con trỏ chuột đã được di chuyển trên một thành phần nhưng không có nút nào được nhấn.

5.5.30.2 Ví dụ `MouseMotionAdapter`

```
package vn.plpsoft.swing;

import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionAdapter;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class MouseMotionAdapterExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public MouseMotionAdapterExam1 () {
        prepareGUI ();
    }
}
```

```
public static void main(String[] args) {
    MouseMotionAdapterExam1 demo = new MouseMotionAdapterExam1();
    demo.showMouseMotionAdapterDemo();
}

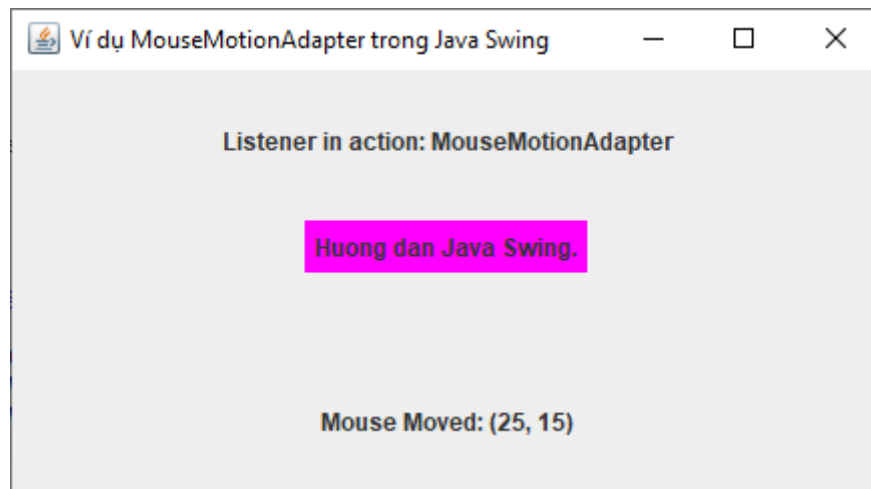
private void prepareGUI() {
    mainFrame = new JFrame("Vi du Java Swing");
    mainFrame.setSize(450, 250);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);

    mainFrame.setTitle("Ví dụ MouseMotionAdapter trong Java Swing");
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_C
LOSE);
}

private void showMouseMotionAdapterDemo() {
    headerLabel.setText("Listener in action:
MouseMotionAdapter");
    JPanel panel = new JPanel();
    panel.setBackground(Color.magenta);
    panel.setLayout(new FlowLayout());
    panel.addMouseMotionListener(new MouseMotionAdapter() {
        public void mouseMoved(MouseEvent e) {
            statusLabel.setText("Mouse Moved: "
                + "(" + e.getX() + ", " + e.getY() + ")");
        }
    });
    JLabel msglabel = new JLabel("Huong dan Java Swing.",
JLabel.CENTER);
    panel.add(msglabel);
    controlPanel.add(panel);
}
```

```
mainFrame.setVisible(true);  
}  
}
```

Chạy chương trình trên cho kết quả như sau:



5.5.31 Lớp WindowAdapter trong Java Swing

Lớp WindowAdapter là một lớp abstract để nhận các keyboard event. Tất cả phương thức của lớp này là trống. Với lớp này, việc tạo các đối tượng Listener trở nên khá thuận tiện. Cú pháp khai báo cho lớp `java.awt.event.WindowAdapter` là:

```
public abstract class WindowAdapter extends Object  
implements WindowListener, WindowStateListener, WindowFocusListener
```

Lớp này kế thừa các phương thức từ lớp `java.lang.Object`.

Lớp WindowAdapter có một constructor là: **WindowAdapter()**.

5.5.31.1 Phương thức của lớp WindowAdapter trong Java Swing

1. **void windowActivated(WindowEvent e)**: Được triệu hồi khi một cửa sổ là activated.
2. **void windowClosed(WindowEvent e)**: Được triệu hồi khi một cửa sổ đã được đóng.
3. **void windowClosing(WindowEvent e)**: Được triệu hồi khi một cửa sổ là trong tiến trình đang được đóng.
4. **void windowDeactivated(WindowEvent e)**: Được triệu hồi khi một cửa sổ là de-activated.
5. **void windowDeiconified(WindowEvent e)**: Được triệu hồi khi một cửa sổ là de-iconified.

6. void windowGainedFocus(WindowEvent e): Được triệu hồi khi một cửa sổ được thiết lập thành focused window, nghĩa là Window, hoặc một trong các thành phần con của nó, sẽ nhận các keyboard event.

7. void windowIconified(WindowEvent e): Được triệu hồi khi một cửa sổ là iconified.

8. void windowLostFocus(WindowEvent e): Được triệu hồi khi một cửa sổ không còn là focused window nữa, nghĩa là các keyboard event không còn được phân phối tới Window hoặc bất cứ thành phần con nào của nó.

9. void windowOpened(WindowEvent e): Được triệu hồi khi một cửa sổ đã được mở.

10. void windowStateChanged(WindowEvent e): Được triệu hồi khi trạng thái một cửa sổ đã thay đổi.

5.5.31.2 Ví dụ WindowAdapter

```
package vn.plpsoft.swing;

import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.WindowConstants;

public class WindowAdapterExam1 {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;

    public WindowAdapterExam1() {
        prepareGUI();
    }

    public static void main(String[] args) {
        WindowAdapterExam1 demo = new WindowAdapterExam1();
    }
}
```

```
demo.showWindowAdapterDemo();
}

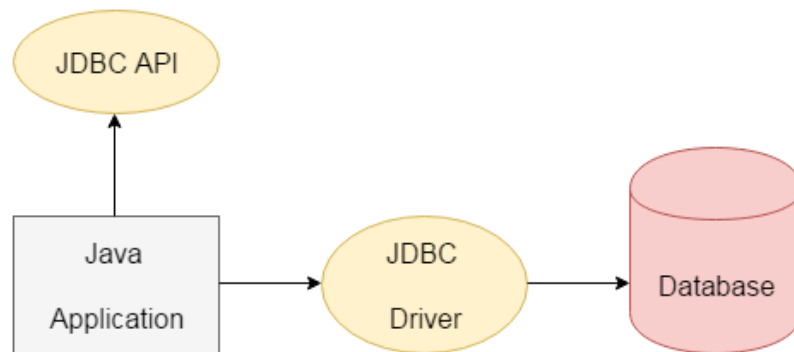
private void prepareGUI() {
    mainFrame = new JFrame("Vi du Java Swing");
    mainFrame.setSize(450, 300);
    mainFrame.setLayout(new GridLayout(3, 1));
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);
    statusLabel.setSize(350, 100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
    mainFrame.setVisible(true);
    mainFrame.setTitle("Ví dụ WindowAdapter trong Java Swing");
    mainFrame.setDefaultCloseOperation(WindowConstants.EXIT_ON_C
LOSE);
}

private void showWindowAdapterDemo() {
    headerLabel.setText("Listener in action: WindowAdapter");
    JButton okButton = new JButton("OK");
    final JFrame aboutFrame = new JFrame();
    aboutFrame.setSize(300, 200);
    ;
    aboutFrame.setTitle("WindowAdapter Demo");
    aboutFrame.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent windowEvent) {
            aboutFrame.dispose();
        }
    });
    JLabel msgLabel = new JLabel("Huong dan Java Swing.",
JLabel.CENTER);
    aboutFrame.add(msgLabel);
    aboutFrame.setVisible(true);
}
}
```

CHƯƠNG 6. Lập trình CSDL trong Java

6.1 Giới thiệu về JDBC (Java Database Connectivity)

Java JDBC là một java API được sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu. JDBC API sử dụng trình điều khiển jdbc để kết nối với cơ sở dữ liệu.



6.1.1 Tại sao sử dụng JDBC

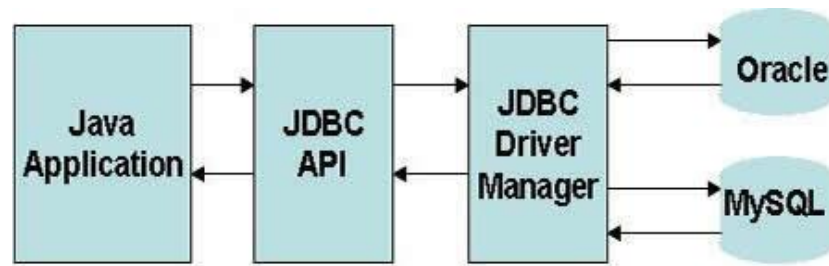
Trước JDBC, ODBC API là một database API được sử dụng để kết nối và thực hiện truy vấn với cơ sở dữ liệu. Nhưng, ODBC API sử dụng trình điều khiển ODBC được viết bằng ngôn ngữ C (tức là nền tảng phụ thuộc và không an toàn). Đó là lý do tại sao Java đã định nghĩa API của chính nó (JDBC API) sử dụng trình điều khiển JDBC (viết bằng ngôn ngữ Java).

6.1.2 API là gì?

API (Application programming interface - Giao diện lập trình ứng dụng) là một tài liệu có chứa mô tả về tất cả các tính năng của một sản phẩm hoặc phần mềm. Nó đại diện cho các lớp và các giao diện (interface) mà các chương trình phần mềm có thể làm theo để giao tiếp với nhau. Một API có thể được tạo cho các ứng dụng, thư viện, hệ điều hành, ...

Bạn có biết?

- Làm thế nào để kết nối cơ sở dữ liệu MySQL trong java?
- Làm thế nào để kết nối cơ sở dữ liệu SQLServer trong java?
- Làm thế nào để kết nối cơ sở dữ liệu Oracle trong java?
- Sự khác nhau giữa giao diện Statement và PreparedStatement?
- Làm thế nào để in tổng số các bảng và view của một CSDL bằng việc sử dụng JDBC?
- Làm thế nào để lưu trữ và lấy ra các file ảnh trong CSDL Oracle?
- Làm thế nào để lưu trữ và lấy ra các file dữ liệu trong CSDL Oracle?



6.1.3 JDBC là gì?

JDBC (Java Database Connectivity) là một API tiêu chuẩn dùng để tương tác với các loại cơ sở dữ liệu quan hệ. **JDBC** có một tập hợp các class và các Interface dùng cho ứng dụng **Java** có thể nói chuyện với các cơ sở dữ liệu.



Các thành phần của JDBC Api về cơ bản bao gồm:

a. DriverManager:

Là một class, nó dùng để quản lý danh sách các **Driver** (database drivers).

b. Driver:

Là một Interface, nó dùng để liên kết các liên lạc với cơ sở dữ liệu, điều khiển các liên lạc với database. Một khi Driver được tải lên, lập trình viên không cần phải gọi nó một cách cụ thể.

c. Connection :

Là một Interface với tất cả các method cho việc liên lạc với database. Nó mô tả nội dung liên lạc. tất cả các thông tin liên lạc với cơ sở dữ liệu là thông qua chỉ có đối tượng **Connection**.

d. Statement :

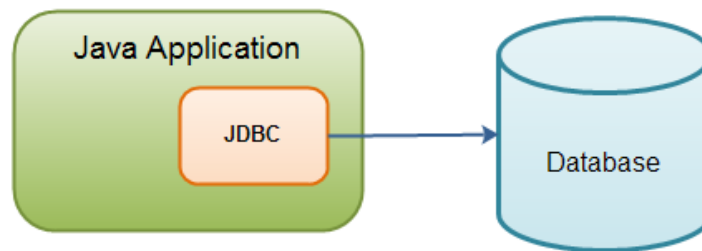
Là một Interface, gói gọn một câu lệnh SQL gửi tới cơ sở dữ liệu được phân tích, tổng hợp, lập kế hoạch và thực hiện.

e. ResultSet:

ResultSet đại diện cho tập hợp các bản ghi lấy do thực hiện truy vấn.

6.1.4 Java kết nối với database dựa trên nguyên tắc nào?

Java sử dụng **JDBC** để làm việc với các cơ sở dữ liệu.

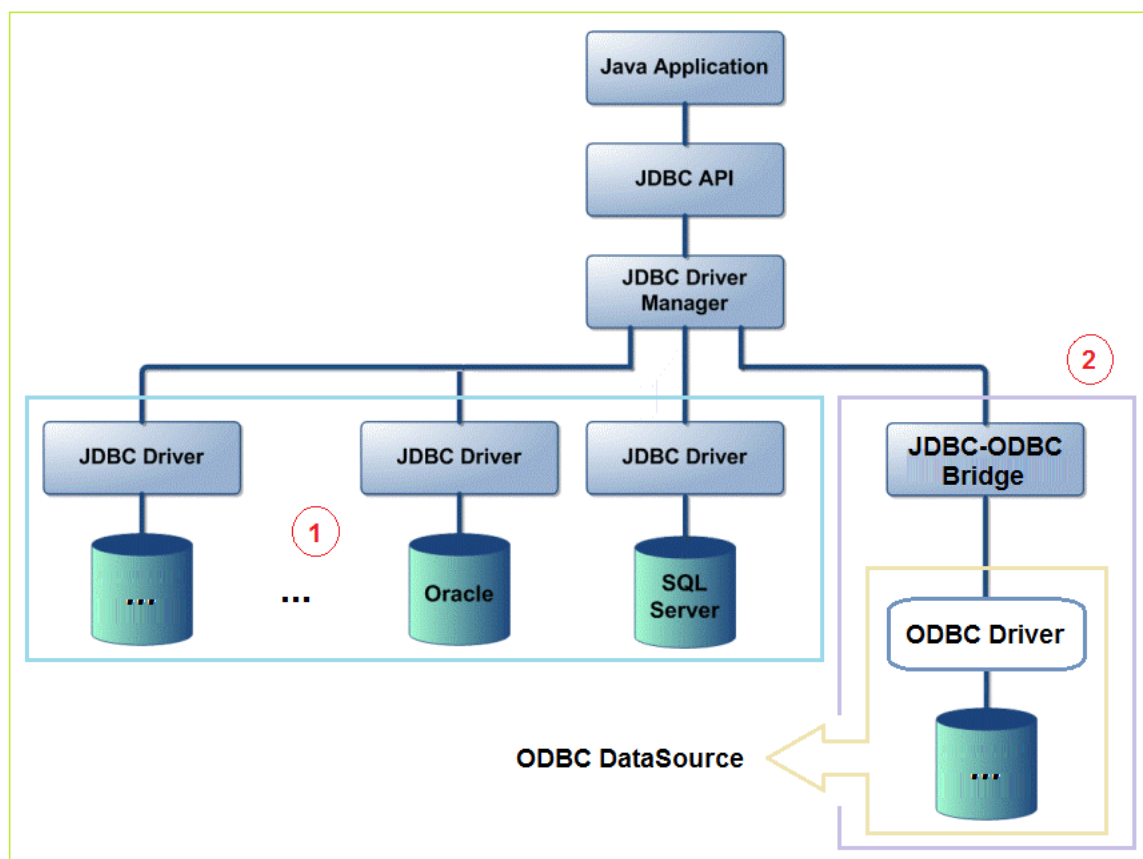


Ví dụ bạn làm việc với cơ sở dữ liệu Oracle từ Java bạn cần phải có Driver (Đó là class điều khiển việc kết nối với loại cơ sở dữ liệu bạn muốn). Trong **JDBC API** chúng ta có **java.sql.Driver**, nó chỉ là một interface, và nó có sẵn trong **JDK**. Như vậy bạn phải download thư viện **Driver** ứng với loại Database mà bạn mong muốn.

Chẳng hạn với **Oracle** thì class thi hành Interface *java.sql.Driver* đó là: **oracle.jdbc.driver.OracleDriver**

java.sql.DriverManager là một class trong **JDBC API**. Nó làm nhiệm vụ quản lý các Driver.

Bạn hãy xem hình minh họa dưới đây:



Chúng ta có 2 cách để làm việc với một loại cơ sở dữ liệu cụ thể nào đó.

Cách 1: Bạn hãy cung cấp thư viện Driver điều khiển loại cơ sở dữ liệu đó, đây là cách trực tiếp. Nếu bạn dùng DB oracle (hoặc DB khác) bạn phải download thư viện dành cho loại DB này.

Cách 2: Khai báo một "ODBC DataSource", và sử dụng cầu nối **JDBC-ODBC** để kết nối với "ODBC DataSource" kia. Cầu nối **JDBC-ODBC** là thứ có sẵn trong **JDBC API**.



ODBC DataSource là gì?

ODBC - Open Database Connectivity: Nó chính là một bộ thư viện mở, có khả năng kết nối với hầu hết các loại cơ sở dữ liệu khác nhau, và nó miễn phí. Được cung cấp bởi **Microsoft**.

ODBC DataSource: Trên hệ điều hành Window bạn có thể khai báo một kết nối ODBC tới một loại DB nào đó. Và như vậy chúng ta có một nguồn dữ liệu (Data Source).

Trong **JDBC API**, đã xây dựng sẵn một cầu nối **JDBC-ODBC** để **JDBC** có thể nói chuyện được với **ODBC Data Source**.

Về tốc độ, cách 1 sẽ nhanh hơn cách 2, vì cách 2 phải sử dụng tới cầu nối.

6.1.5 Download một số các driver quan trọng

Trong trường hợp nếu bạn không muốn sử dụng **JDBC-ODBC**, bạn có thể sử dụng cách trực tiếp kết nối vào Database, trong trường hợp đó cần phải download Driver ứng với mỗi loại DB này. Tại đây tôi hướng dẫn download một loại Driver cho các Database thông dụng:

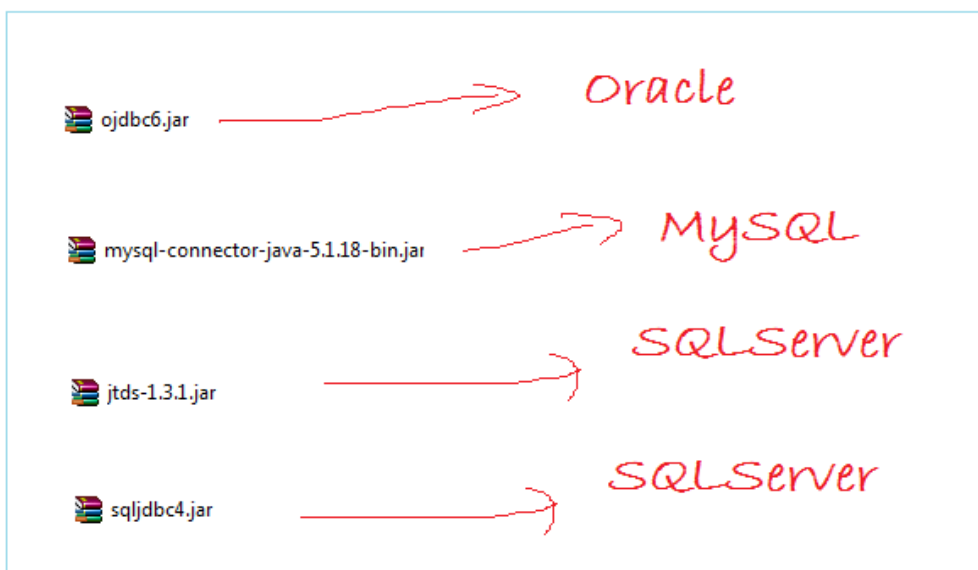
Oracle

MySQL

SQLServer

....

Kết quả chúng ta có một vài file:



6.2 Kết nối dữ liệu với MySQL, SQL Server, Oracle,...

6.2.1 Kết nối Java với MySQL

Để kết nối Java với **MySQL** bằng **JDBC**, giả sử bạn đã tạo bảng trong MySQL, bạn cần làm theo 4 bước sau:

1. Tải file **mysql-connector-java-x.y.zz.zip** về máy tại <https://dev.mysql.com/downloads/connector/j/>, giải nén ra được file **mysql-connector-java-x.y.zz-bin.jar**.
2. Add thư viện JDBC Driver **mysql-connector-java-x.y.zz-bin.jar** vào project.
3. Gọi phương thức `Class.forName("com.mysql.jdbc.Driver")`.
4. Gọi phương thức `DriverManager.getConnection()` để kết nối đến cơ sở dữ liệu MySQL.

Chi tiết về việc kết nối ứng dụng Java với cơ sở dữ liệu MySQL bằng JDBC được thể hiện trong ví dụ dưới đây.

6.2.2 Ví dụ về kết nối Java với MySQL

Tạo bảng 'student' trong cơ sở dữ liệu có tên 'testdb' trong MySQL với câu lệnh như sau:

```
CREATE TABLE student (  
    id      INT      NOT NULL,  
    name VARCHAR (32)      NOT NULL,  
    address VARCHAR (32) NOT NULL,  
    PRIMARY KEY (id)  
);
```

Insert vài dòng dữ liệu cho bảng 'student'

```
INSERT INTO student(id, name, address) VALUES (1, "Công", "Hanoi");  
INSERT INTO student(id, name, address) VALUES (2, "Dung", "Vinhphuc");  
INSERT INTO student(id, name, address) VALUES (3, "Ngôn", "Danang");  
INSERT INTO student(id, name, address) VALUES (4, "Hạnh", "Hanoi")
```

Sau khi thực thi các câu lệnh trên chúng ta có được dữ liệu như sau:

				id	name	address
<input type="checkbox"/>		Sửa		Chép		Xóa bỏ
				1	Công	Hanoi
<input type="checkbox"/>		Sửa		Chép		Xóa bỏ
				2	Dung	Vinhphuc
<input type="checkbox"/>		Sửa		Chép		Xóa bỏ
				3	Ngôn	Danang
<input type="checkbox"/>		Sửa		Chép		Xóa bỏ
				4	Hạnh	Hanoi

Tạo chương trình để kết nối và hiển thị dữ liệu của bảng 'student' ra màn hình như sau:

File: **ConnectMysqlExample.java**

```
package vn.plpsoft.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConnectMysqlExample {
    private static String DB_URL =
        "jdbc:mysql://localhost:3306/testdb";
    private static String USER_NAME = "root";
    private static String PASSWORD = "1234567890";
    /**
     * main
     *
     * @author plpsoft.vn
     * @param args
     */
    public static void main(String args[]) {
        try {
            // connect to database 'testdb'
            Connection conn = getConnection(DB_URL, USER_NAME, PASSWORD);
            // create statement
            Statement stmt = conn.createStatement();
            // get data from table 'student'
            ResultSet rs = stmt.executeQuery("select * from student");
            // show data
            while (rs.next()) {
                System.out.println(rs.getInt(1) + " " + rs.getString(2))
            }
        }
    }
}
```

```
        + " " + rs.getString(3));
    }
    // close connection
    conn.close();
} catch (Exception ex) {
    ex.printStackTrace();
}
}
/**
 * create connection
 *
 * @author plpsoft.vn
 * @param dbURL: database's url
 * @param userName: username is used to login
 * @param password: password is used to login
 * @return connection
 */
public static Connection getConnection(String dbURL, String
userName, String password) {
    Connection conn = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection(dbURL, userName, password);
        System.out.println("connect successfully!");
    } catch (Exception ex) {
        System.out.println("connect failure!");
        ex.printStackTrace();
    }
    return conn;
}
}
```

Kết quả:

```
connect successfully!
```

```
1  Công   Hanoi
2  Dung   Vinhphuc
3  Ngôn   Danang
4  Hạnh   Hanoi
```

Tham khảo bài học: [Cài đặt MySQL](#)

6.2.3 Kết nối Java với SQL Server

Để kết nối Java với SQL Server bằng JDBC, giả sử bạn đã tạo bảng trong SQL Server, bạn cần làm theo 4 bước sau:

1. Download Microsoft JDBC Driver tại [đây](#), giải nén ra ta được:
 - file **sqljdbc41.jar** trong thư mục Microsoft JDBC Driver 6.0 for SQL Server\sqljdbc_6.0\enu\jre7, file driver này được sử dụng với jdk/jre 7
 - file **sqljdbc42.jar** trong thư mục Microsoft JDBC Driver 6.0 for SQL Server\sqljdbc_6.0\enu\jre8, file driver này được sử dụng với jdk/jre 8.Một bước **quan trọng** không kém đó là copy file **sqljdbc_auth.dll** của thư mục Microsoft JDBC Driver 6.0 for SQL Server\sqljdbc_6.0\enu\auth\xx (với xx là version của window của bạn, 32 hoặc 64) vào thư mục C:\Windows\System32.
2. Add thư viện sqljdbc4x.jar vào project tương ứng với phiên bản JDK bạn đang sử dụng.
3. Định nghĩa URL của database và username + password để truy cập vào database đó.
4. Đăng ký JDBC driver cho SQL Server và thiết lập kết nối.

Chi tiết về việc kết nối ứng dụng Java với cơ sở dữ liệu SQL Server bằng JDBC được thể hiện trong ví dụ dưới đây.

6.2.4 Ví dụ về kết nối Java với SQL Server

Tạo bảng 'student' trong cơ sở dữ liệu có tên 'testdb' trong SQL Server với câu lệnh như sau:

```
CREATE TABLE student (  
    id INT NOT NULL,  
    name NVARCHAR (32) NOT NULL,  
    address NVARCHAR (32) NOT NULL,  
    PRIMARY KEY (id)  
);
```

Insert vài dòng dữ liệu cho bảng 'student'

```
INSERT INTO student(id, name, address) VALUES (1, N'Công', 'Hanoi');  
INSERT INTO student(id, name, address) VALUES (2, N'Dung', 'Vinhphuc');  
INSERT INTO student(id, name, address) VALUES (3, N'Ngôn', 'Danang');  
INSERT INTO student(id, name, address) VALUES (4, N'Hạnh', 'Hanoi');
```

Sau khi thực thi các câu lệnh trên chúng ta có được dữ liệu như sau:

	id	name	address
1	1	Công	Hanoi
2	2	Dung	Vinhphuc
3	3	Ngôn	Danang
4	4	Hạnh	Hanoi

Tạo chương trình để kết nối và hiển thị dữ liệu của bảng 'student' ra màn hình như sau:

File: ConnectSQLServerExample.java

```
package vn.plpsoft.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConnectSQLServerExample {
    private static String DB_URL =
        "jdbc:sqlserver://localhost:1433;"
        + "databaseName=testdb;"
        + "integratedSecurity=true";
    private static String USER_NAME = "sa";
    private static String PASSWORD = "1234567890";
    /**
     * main
     *
     * @author plpsoft.vn
     * @param args
     */
    public static void main(String args[]) {
        try {
            // connect to database 'testdb'
            Connection conn = getConnection(DB_URL, USER_NAME, PASSWORD);
            // create statement
            Statement stmt = conn.createStatement();
            // get data from table 'student'
            ResultSet rs = stmt.executeQuery("select * from student");
            // show data
            while (rs.next()) {
                System.out.println(rs.getInt(1) + "  " +
```

```
rs.getString(2)
                + " " + rs.getString(3));
    }
    // close connection
    conn.close();
} catch (Exception ex) {
    ex.printStackTrace();
}
}
/**
 * create connection
 *
 * @author plpsoft.vn
 * @param dbURL: database's url
 * @param userName: username is used to login
 * @param password: password is used to login
 * @return connection
 */
public static Connection getConnection(String dbURL, String
userName,String password) {
    Connection conn = null;
    try{
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
        conn = DriverManager.getConnection(dbURL, userName, password);
        System.out.println("connect successfully!");
    } catch (Exception ex) {
        System.out.println("connect failure!");
        ex.printStackTrace();
    }
    return conn;
}
}
```

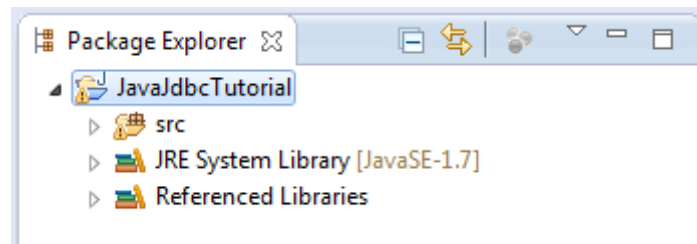
Kết quả:

```
connect successfully!
1  Công   Hanoi
2  Dung   Vinhphuc
3  Ngôn   Danang
4  Hạnh   Hanoi
```


6.2.5 Sample Project

1. Tạo project để bắt đầu ví dụ với JDBC

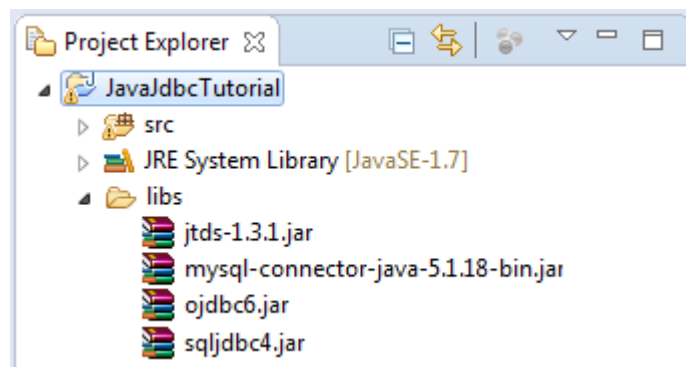
Tạo mới project **JavaJdbcTutorial**:



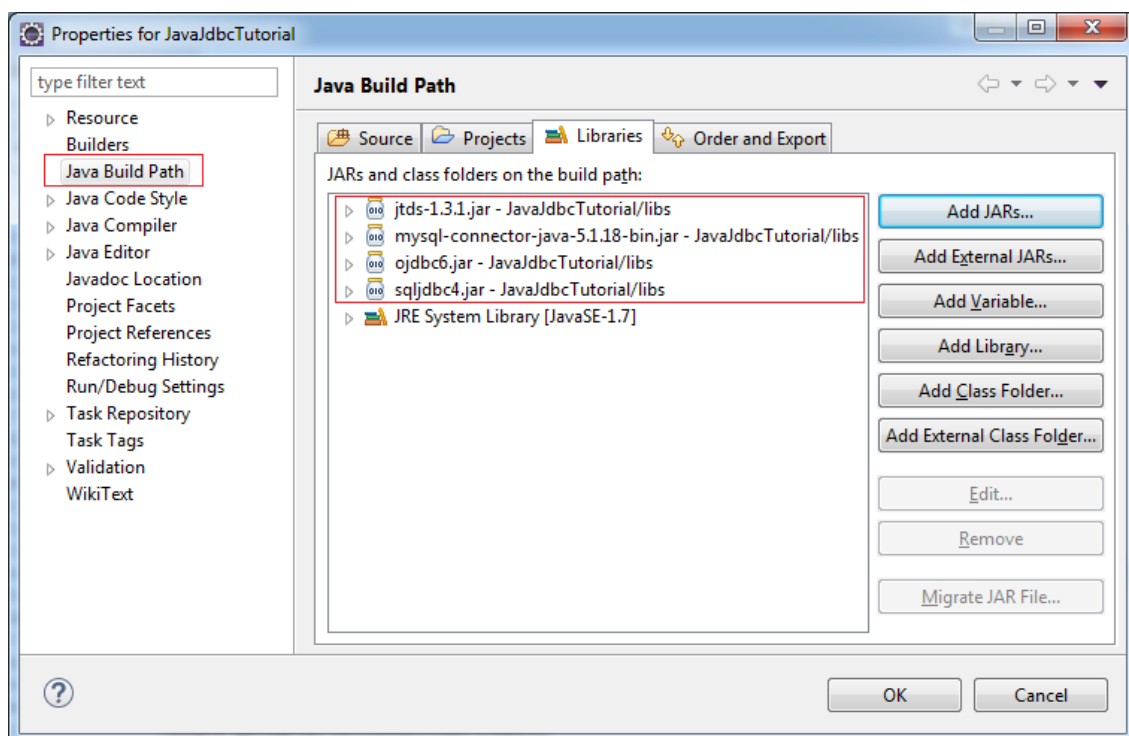
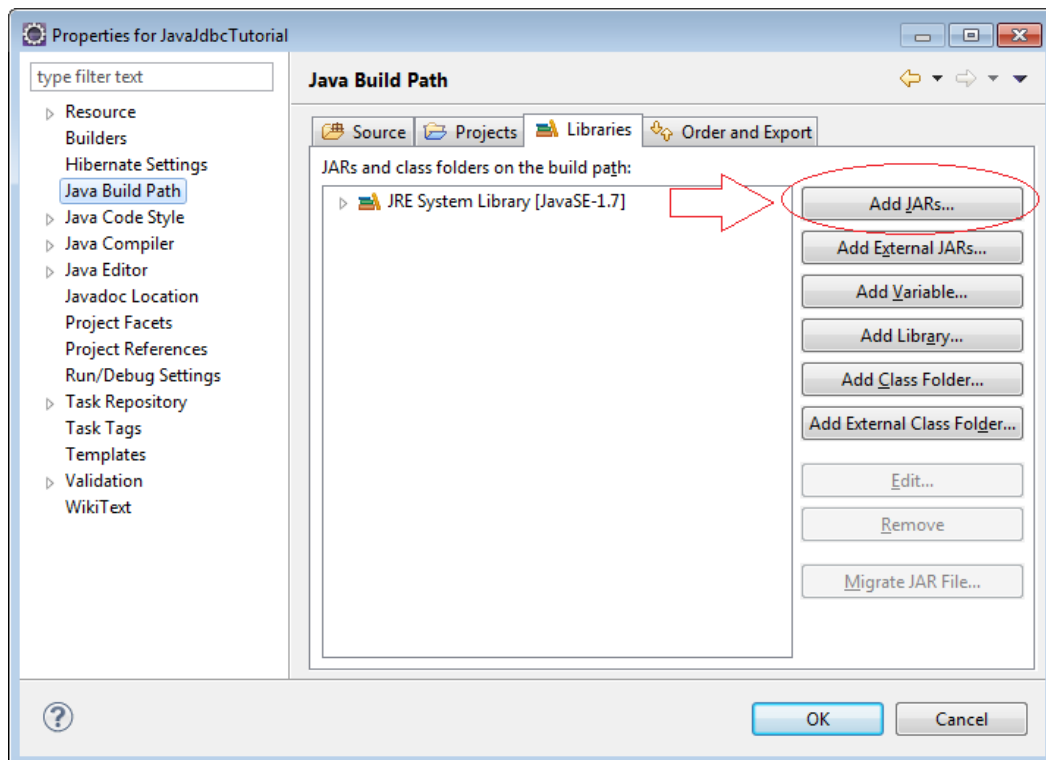
Tạo thư mục **libs** trên project và copy các thư viện kết nối trực tiếp các loại database **Oracle**, **MySQL**, **SQLServer** mà bạn vừa download được ở trên vào. Bạn có thể copy hết hoặc một trong các thư viện đó, theo loại DB mà bạn sử dụng.

Chú ý: Bạn chỉ cần download một Driver ứng với loại Database mà bạn quen thuộc. Cơ sở dữ liệu dùng làm ví dụ trong tài liệu này bạn có thể lấy tại:

Cơ sở dữ liệu mẫu



Nhấn phải vào Project chọn Properties:



2. Connection to Database (Oracle, MySQL, SQL Server)

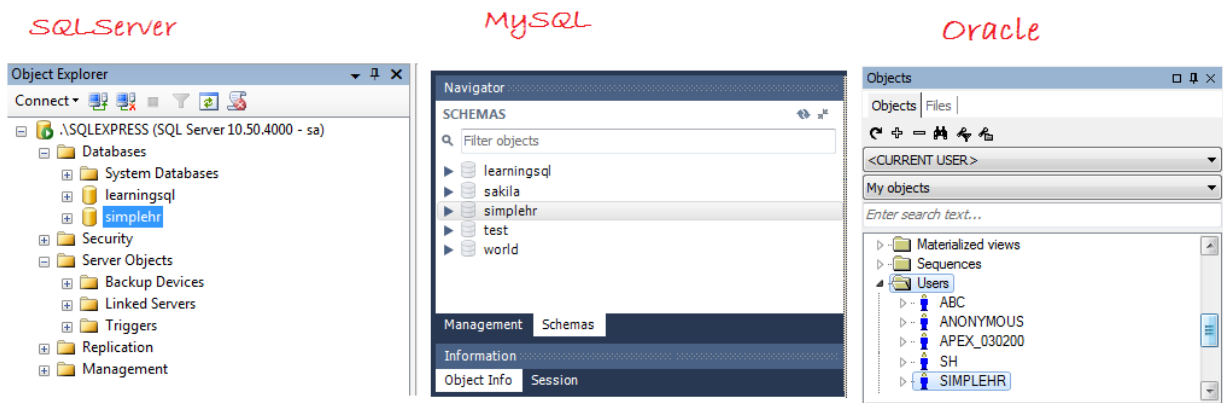
Trong project này sẽ hướng dẫn các kết nối vào cả 3 loại database:

MySQL

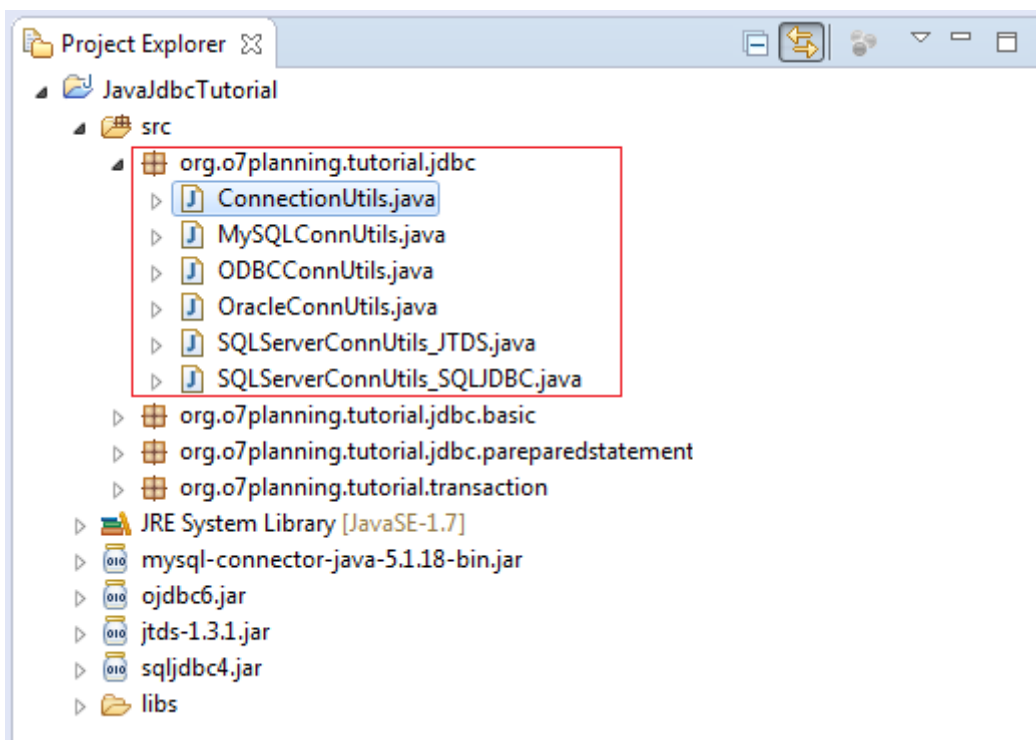
SQLServer

Oracle

Trong khi thực hành, bạn chỉ cần làm việc với một loại **DB** nào mà bạn quen thuộc.



Chúng ta tạo class **ConnectionUtils** để lấy ra đối tượng **Connection** kết nối với Database.



ConnectionUtils.java

```
package org.plpsoft.tutorial.jdbc;

import java.sql.Connection;
import java.sql.SQLException;

public class ConnectionUtils {

    public static Connection getMyConnection() throws SQLException,
        ClassNotFoundException {
```

```
// Sử dụng Oracle.
// Bạn có thể thay thế bởi Database nào đó.
return OracleConnUtils.getOracleConnection();
}

//
// Test Connection ...
//
public static void main(String[] args) throws SQLException,
    ClassNotFoundException {

    System.out.println("Get connection ... ");

    // Lấy ra đối tượng Connection kết nối vào database.
    Connection conn = ConnectionUtils.getMyConnection();

    System.out.println("Get connection " + conn);

    System.out.println("Done!");
}
}
```

OracleConnUtils.java

```
package org.plpsoft.tutorial.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class OracleConnUtils {

    // Kết nối vào ORACLE.
    public static Connection getOracleConnection() throws
        SQLException,
        ClassNotFoundException {
        String hostName = "localhost";
        String sid = "db11g";
        String userName = "simplehr";
        String password = "simplehr";
```

```
        return getOracleConnection(hostName, sid, userName,
password);
    }

    public static Connection getOracleConnection(String hostName,
String sid,
        String userName, String password) throws
ClassNotFoundException,
        SQLException {

        // Khai báo class Driver cho DB Oracle
        // Việc này cần thiết với Java 5
        // Java6 tự động tìm kiếm Driver thích hợp.
        // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.
        Class.forName("oracle.jdbc.driver.OracleDriver");

        // Cấu trúc URL Connection dành cho Oracle
        // Ví dụ: jdbc:oracle:thin:@localhost:1521:db11g
        String connectionURL = "jdbc:oracle:thin:@" + hostName +
":1521:" + sid;

        Connection conn =
        DriverManager.getConnection(connectionURL, userName,
            password);
        return conn;
    }
}
```

MySQLConnUtils.java

```
package org.plpsoft.tutorial.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class MySQLConnUtils {
    // Kết nối vào MySQL.
    public static Connection getMySQLConnection() throws
SQLException,
        ClassNotFoundException {
```

```
String hostName = "localhost";

String dbName = "simplehr";
String userName = "root";
String password = "1234";

return getMySQLConnection(hostName, dbName, userName, password);
}

public static Connection getMySQLConnection(String hostName,
String dbName,
String userName, String password) throws SQLException,
ClassNotFoundException {
// Khai báo class Driver cho DB MySQL
// Việc này cần thiết với Java 5
// Java6 tự động tìm kiếm Driver thích hợp.
// Nếu bạn dùng Java6, thì ko cần dòng này cũng được.
Class.forName("com.mysql.jdbc.Driver");

// Cấu trúc URL Connection dành cho Oracle
// Ví dụ: jdbc:mysql://localhost:3306/simplehr
String connectionURL = "jdbc:mysql://" + hostName +
":3306/" + dbName;

Connection conn =
DriverManager.getConnection(connectionURL, userName,
password);
return conn;
}
}
```

SQLServerConnUtils_JTDS.java

```
package org.plpsoft.tutorial.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class SQLServerConnUtils_JTDS {
```

```
// Kết nối vào SQLServer.
// (Sử dụng thư viện điều khiển JTDS)
public static Connection getSQLServerConnection()
    throws SQLException, ClassNotFoundException {
    String hostName = "localhost";
    String sqlInstanceName = "SQLEXPRESS";
    String database = "simplehr";
    String userName = "sa";
    String password = "1234";

    return getSQLServerConnection(hostName, sqlInstanceName,
        database,
            userName, password);
}

// Trường hợp sử dụng SQLServer.
// Và thư viện JTDS.
public static Connection getSQLServerConnection(String
    hostName, String sqlInstanceName, String database, String
    userName, String password) throws ClassNotFoundException,
    SQLException {
    // Khai báo class Driver cho DB SQLServer
    // Việc này cần thiết với Java 5
    // Java6 tự động tìm kiếm Driver thích hợp.
    // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.
    Class.forName("net.sourceforge.jtds.jdbc.Driver");

    // Cấu trúc URL Connection dành cho SQLServer
    // Ví dụ:
    //
    jdbc:jtds:sqlserver://localhost:1433/simplehr;instance=SQLEXPRESS
    String connectionURL = "jdbc:jtds:sqlserver://" + hostName
    + ":1433/"
        + database + ";instance=" + sqlInstanceName;

    Connection conn =
    DriverManager.getConnection(connectionURL, userName,
        password);
    return conn;
}
```

```
}
```

```
}
```

SQLServerConnUtils_SQLJDBC.java

```
package org.plpsoft.tutorial.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class SQLServerConnUtils_SQLJDBC {

    // Kết nối vào SQLServer.
    // (Sử dụng thư viện điều khiển SQLJDBC)
    public static Connection getSQLServerConnection()
        throws SQLException, ClassNotFoundException {
        String hostName = "localhost";
        String sqlInstanceName = "SQLEXPRESS";
        String database = "simplehr";
        String userName = "sa";
        String password = "1234";

        return getSQLServerConnection(hostName, sqlInstanceName,
            database, userName, password);
    }

    // Trường hợp sử dụng SQLServer.
    // Và thư viện SQLJDBC.
    public static Connection getSQLServerConnection(String
        hostName,
        String sqlInstanceName, String database, String
        userName,
        String password) throws ClassNotFoundException,
        SQLException {
        // Khai báo class Driver cho DB SQLServer
        // Việc này cần thiết với Java 5
        // Java6 tự động tìm kiếm Driver thích hợp.
        // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.
        Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDr
```



```
iver");

    // Cấu trúc URL Connection dành cho SQLServer
    // Ví dụ:
    //
    jdbc:sqlserver://ServerIp:1433/SQLEXPRESS;databaseName=simplehr

    String connectionURL = "jdbc:sqlserver://" + hostName +
        ":1433" + ";instance=" + sqlInstanceName + ";databaseName=" +
        database;

    Connection conn =
        DriverManager.getConnection(connectionURL, userName,
            password);
    return conn;
}

}
```

ODBCConnUtils.java

```
package org.plpsoft.tutorial.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ODBCConnUtils {

    // Lấy ra kết nối vào ODBC Data Source có tên "simplehr-ds".
    public static Connection getJdbcOdbcConnection() throws
        SQLException,
            ClassNotFoundException {
        String odbcDataSourceName = "simplehr-ds";
        String userName = "simplehr";
        String password = "simplehr";
        return getJdbcOdbcConnection(odbcDataSourceName, userName,
            password);
    }

    public static Connection getJdbcOdbcConnection(String
        odbcDataSourceName,
```

```
String userName, String password) throws SQLException,
ClassNotFoundException {

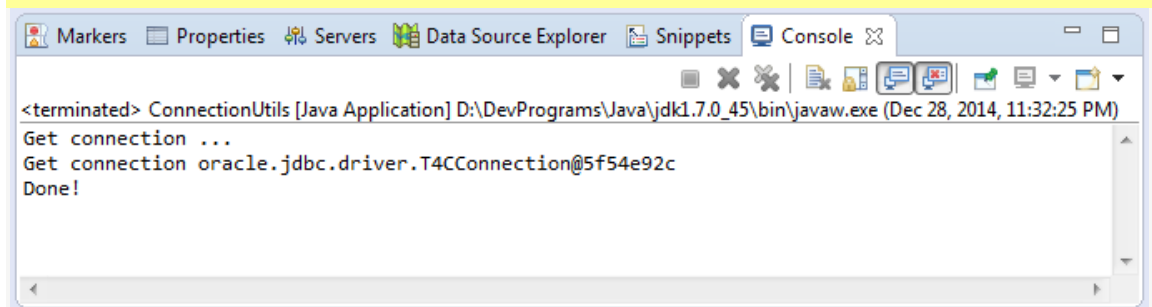
    // Khai báo class Driver (Cầu nối Jdbc-Odbc)
    // Việc này cần thiết với Java 5
    // Java6 tự động tìm kiếm Driver thích hợp.
    // Nếu bạn dùng Java6, thì ko cần dòng này cũng được.
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

    // Cấu trúc URL Connection dành cho JDBC-ODBC
    String connectionURL = "jdbc:odbc:" + odbcDataSourceName;

    Connection conn =
        DriverManager.getConnection(connectionURL, userName,
                                   password);
    return conn;
}
}
```

Bạn có thể thay đổi Class **ConnectionUtils** để sử dụng kết nối tới một Database nào đó quen thuộc. Và chạy class này để test kết nối.

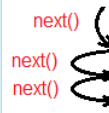
*Chú ý: Nếu bạn sử dụng **MySQL** hoặc **SQL Server** mặc định 2 Database này chặn không cho phép kết nối vào nó từ một IP khác. Bạn cần cấu hình để cho phép điều này. Bạn có thể xem hướng dẫn trong tài liệu cài đặt và cấu hình **MySQL**, **SQL Server** trên **plpsoft**.*



3- Sử dụng JDBC API truy vấn dữ liệu

Đây là hình ảnh dữ liệu trong bảng **Employee**. Chúng ta sẽ xem cách **Java** lấy ra dữ liệu thế nào thông qua một ví dụ:

```
Select Emp_Id, Emp_No, Emp_Name from Employee
```



	1	2	3
	EMP_ID	EMP_NO	EMP_NAME
1	7839	E7839	KING
2	7566	E7566	JONES
3	7902	E7902	FORD
4	7369	E7369	SMITH
5	7698	E7698	BLAKE
6	7499	E7499	ALLEN
7	7521	E7521	WARD
8	7654	E7654	MARTIN
9	7782	E7782	CLARK
10	7788	E7788	SCOTT
11	7844	E7844	TURNER
12	7876	E7876	ADAMS
13	7900	E7900	ADAMS
14	7934	E7934	MILLER

```
Connection connection = ....;
```

```
Statement statement =  
    connection.createStatement();
```

```
ResultSet rs = statement.executeQuery(sql);
```

```
while (rs.next()) {  
    int empId = rs.getInt(1);  
    String empNo = rs.getString(2);  
    String empName = rs.getString("Emp_Name");  
}
```

ResultSet là một đối tượng Java, nó được trả về khi bạn truy vấn (query) dữ liệu. Sử dụng **ResultSet.next()** để di chuyển con trỏ tới các bản ghi tiếp theo (Di chuyển dòng). Tại một bản ghi nào đó bạn sử dụng các method **ResultSet.get Xxx()** để lấy ra các giá trị tại các cột. Các cột được đánh với thứ tự 1,2,3,...

**** ResultSet ****

```
public String getString(int columnIndex) throws SQLException;  
public boolean getBoolean(int columnIndex) throws SQLException;  
public int getInt(int columnIndex) throws SQLException;  
public double getDouble(int columnIndex) throws SQLException;  
...  
public String getString(String columnLabel) throws  
    SQLException;  
public boolean getBoolean(String columnLabel) throws  
    SQLException;  
public int getInt(String columnLabel) throws SQLException;  
public double getDouble(String columnLabel) throws  
    SQLException;  
....
```

Ví dụ minh họa:

QueryDataExample.java

```
package org.plpsoft.tutorial.jdbc.basic;  
  
import java.sql.Connection;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
import org.plpsoft.tutorial.jdbc.ConnectionUtils;
```

```
public class QueryDataExample {

    public static void main(String[] args) throws
        ClassNotFoundException,
            SQLException {

        // Lấy ra đối tượng Connection kết nối vào DB.
        Connection connection = ConnectionUtils.getMyConnection();

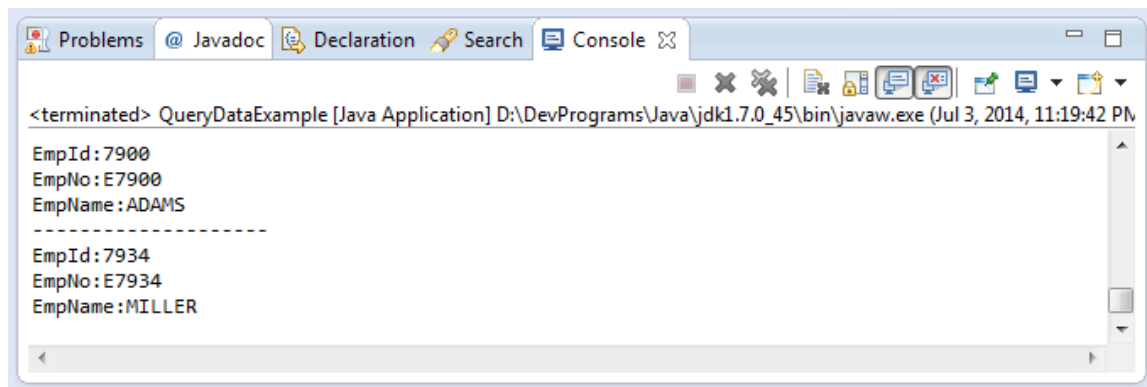
        // Tạo đối tượng Statement.
        Statement statement = connection.createStatement();

        String sql = "Select Emp_Id, Emp_No, Emp_Name from Employee";

        // Thực thi câu lệnh SQL trả về đối tượng ResultSet.
        ResultSet rs = statement.executeQuery(sql);

        // Duyệt trên kết quả trả về.
        while (rs.next()) {
            // Di chuyển con trỏ xuống bản ghi kế tiếp.
            int empId = rs.getInt(1);
            String empNo = rs.getString(2);
            String empName = rs.getString("Emp_Name");
            System.out.println("-----");
            System.out.println("EmpId:" + empId);
            System.out.println("EmpNo:" + empNo);
            System.out.println("EmpName:" + empName);
        }
        // Đóng kết nối
        connection.close();
    }
}
```

Kết quả chạy ví dụ:



4- Các kiểu ResultSet

Bạn đã làm quen với **ResultSet** với các ví dụ phía trên. Mặc định các **ResultSet** khi duyệt dữ liệu chỉ có thể chạy từ trên xuống dưới, từ trái sang phải. Điều đó có nghĩa là với các **ResultSet** mặc định bạn không thể gọi:

- **ResultSet.previous()** : Lùi lại một bản ghi.
- Trên cùng một bản ghi không thể gọi **ResultSet.getXxx(4)** rồi mới gọi **ResultSet.getXxx(2)**.

Việc cố tình gọi sẽ bị một **Exception**.

```
public Statement createStatement(int resultSetType, int
resultSetConcurrency)
    throws SQLException;

// Ví dụ:
Statement statement = connection.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_READ_ONLY);

// ResultSet có thể cuộn (tiến lùi, sang trái sang phải).
ResultSet rs = statement.executeQuery(sql);
```

Ý nghĩa
- ResultSet chỉ cho phép duyệt từ trên xuống dưới, từ trái sang phải. Đây là kiểu mặc định của các ResultSet .
- ResultSet cho phép cuộn tiến lùi, sang trái, sang phải, nhưng không nhạy với các sự thay đổi dữ liệu dưới DB. Nghĩa là trong quá trình duyệt qua một bản ghi và lúc nào đó duyệt lại bản ghi đó, nó không lấy các dữ liệu mới nhất của bản ghi mà có thể bị ai đó thay đổi.
- ResultSet cho phép cuộn tiến lùi, sang trái, sang phải, và nhạy cảm với sự thay đổi dữ liệu.

resultSetConcurrency	Ý nghĩa
CONCUR_READ_ONLY	- Khi duyệt dữ liệu với các ResultSet kiểu này bạn chỉ có thể đọc dữ liệu.
CONCUR_UPDATABLE	- Khi duyệt dữ liệu với các ResultSet kiểu này bạn chỉ có thể thay đổi dữ liệu tại nơi con trỏ đứng, ví dụ update giá trị cột nào đó.

ScrollableResultSetExample.java

```
package org.plpsoft.tutorial.jdbc.basic;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.plpsoft.tutorial.jdbc.ConnectionUtils;

public class ScrollableResultSetExample {

    public static void main(String[] args) throws
        ClassNotFoundException,
            SQLException {

        // Lấy ra đối tượng Connection kết nối tới DB.
        Connection connection = ConnectionUtils.getMyConnection();

        // Tạo một đối tượng Statement
        // Có thể cuộn dữ liệu, nhưng không nhẩy với các thay đổi dưới DB.
        // Con trỏ chỉ có khả năng đọc, không có khả năng update dữ liệu
        // trong quá trình duyệt.
        Statement statement = connection.createStatement(
            ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);

        String sql = "Select Emp_Id, Emp_No, Emp_Name from Employee";

        // Thực thi câu lệnh SQL trả về đối tượng ResultSet.
        ResultSet rs = statement.executeQuery(sql);
```

```
// Nhảy con trỏ tới cuối
boolean last = rs.last();

System.out.println("last : " + last);

if(last) {
    // Ghi ra thông tin bản ghi cuối.
    System.out.println("EmpId:" + rs.getInt(1));
    System.out.println("EmpNo:" + rs.getString(2));
    System.out.println("EmpName:" + rs.getString(3));
}

System.out.println("-----");

// Nhảy con trỏ lùi lại lần 1
boolean previous = rs.previous();
System.out.println("Previous 1: " + previous);

// Nhảy lùi con trỏ lần 2
previous = rs.previous();
System.out.println("Previous 2: " + previous);

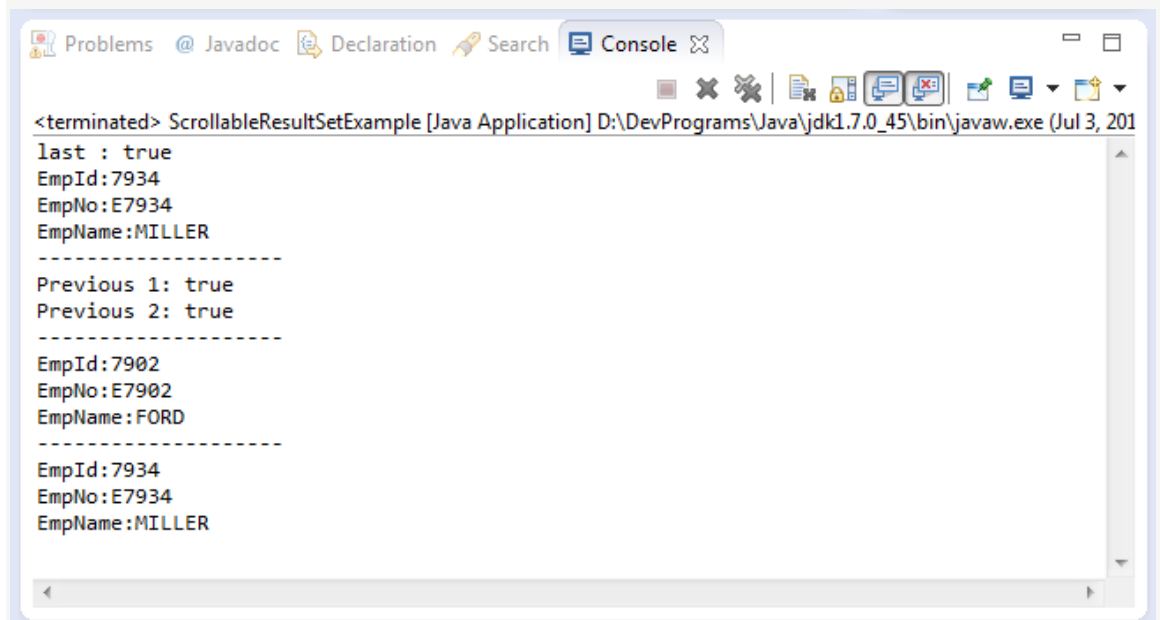
// Duyệt trên kết quả trả về.
while (rs.next()) {
    // Lấy dữ liệu cột 2
    String empNo = rs.getString(2);
    // Rồi mới lấy dữ liệu cột 1.
    int empId = rs.getInt(1);

    String empName = rs.getString("Emp_Name");

    System.out.println("-----");
    System.out.println("EmpId:" + empId);
    System.out.println("EmpNo:" + empNo);
    System.out.println("EmpName:" + empName);
}
```

```
// Đóng kết nối  
connection.close();  
  
}  
  
}
```

Kết quả chạy ví dụ:



5- Ví dụ Insert dữ liệu

InsertDataExample.java

```
package org.plpsoft.tutorial.jdbc.basic;  
  
import java.sql.Connection;  
import java.sql.SQLException;  
import java.sql.Statement;  
  
import org.plpsoft.tutorial.jdbc.ConnectionUtils;  
  
public class InsertDataExample {  
  
    public static void main(String[] args) throws  
        ClassNotFoundException,  
        SQLException {  
  
        // Lấy ra kết nối tới cơ sở dữ liệu.  
        Connection connection = ConnectionUtils.getMyConnection();
```



```
Statement statement = connection.createStatement();

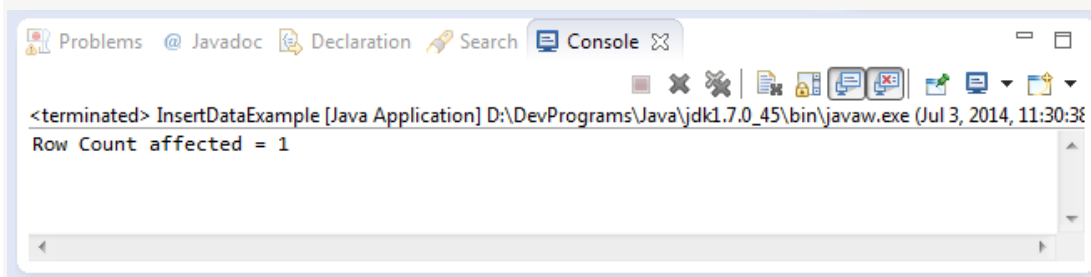
String sql = "Insert into Salary_Grade(Grade, High_Salary, Low_Salary)"
            + " values (2, 20000, 10000) ";

// Thực thi câu lệnh.
// executeUpdate(String) sử dụng cho các loại
//lệnh Insert,Update,Delete.

int rowCount = statement.executeUpdate(sql);

// In ra số dòng được trèn vào bởi câu lệnh trên.
System.out.println("Row Count affected = " + rowCount);
}
}
```

Kết quả chạy ví dụ:



6- PreparedStatement

PreparedStatement là một Interface con của **Statement**.

PreparedStatement sử dụng để chuẩn bị trước các câu lệnh **SQL**, và tái sử dụng nhiều lần, giúp cho chương trình thực hiện nhanh hơn.

PreparedStatementExample.java

```
package org.plpsoft.tutorial.jdbc.pareparedstatement;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import org.plpsoft.tutorial.jdbc.ConnectionUtils;

public class PreparedStatementExample {
```

```
public static void main(String[] args) throws
ClassNotFoundException,
    SQLException {
    // Lấy ra kết nối tới cơ sở dữ liệu.
    Connection connection = ConnectionUtils.getMyConnection();

    // Tạo một câu SQL có 2 tham số (?)
    String sql = "Select emp.Emp_Id, emp.Emp_No, emp.Emp_Name,
emp.Dept_Id from Employee emp "
        + " where emp.Emp_Name like ? and emp.Dept_Id = ? ";

    // Tạo một đối tượng PreparedStatement.
    PreparedStatement pstmt = connection.prepareStatement(sql);

    // Sét đặt giá trị tham số thứ nhất (Dấu ? thứ nhất)
    pstmt.setString(1, "%S");
    // Sét đặt giá trị tham số thứ hai (Dấu ? thứ hai)
    pstmt.setInt(2, 20);

    ResultSet rs = pstmt.executeQuery();

    while (rs.next()) {
        System.out.println(" ---- ");
        System.out.println("EmpId : " + rs.getInt("Emp_Id"));
        System.out.println("EmpNo : " + rs.getString(2));
        System.out.println("EmpName : " +
rs.getString("Emp_Name"));
    }

    System.out.println();
    System.out.println("Set other parameters ..");

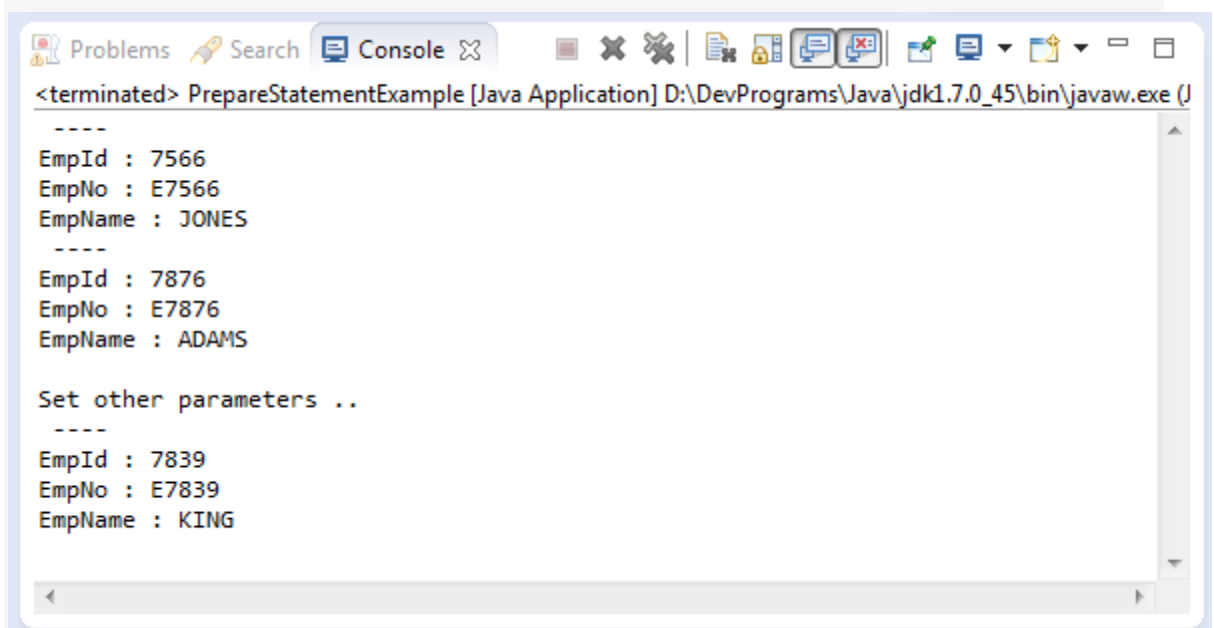
    // Tái sử dụng PreparedStatement.
    // Sét đặt các tham số khác.
    pstmt.setString(1, "KI%");
    pstmt.setInt(2, 10);

    // Thực thi câu lệnh truy vấn.
```

```
rs = pstmt.executeQuery();

while (rs.next()) {
    System.out.println(" ---- ");
    System.out.println("EmpId : " + rs.getInt("Emp_Id"));
    System.out.println("EmpNo : " + rs.getString(2));
    System.out.println("EmpName : " +
rs.getString("Emp_Name"));
}
}
```

Kết quả chạy ví dụ:



```
<terminated> PrepareStatementExample [Java Application] D:\DevPrograms\Java\jdk1.7.0_45\bin\javaw.exe (J
----
EmpId : 7566
EmpNo : E7566
EmpName : JONES
----
EmpId : 7876
EmpNo : E7876
EmpName : ADAMS

Set other parameters ..
----
EmpId : 7839
EmpNo : E7839
EmpName : KING
```

7- CallableStatement

CallableStatement được xây dựng để gọi một thủ tục (procedure) hoặc hàm (function) của **SQL**.

```
// Câu lệnh gọi thủ tục SQL trên Java
String sql = "{call procedure_name(?,?,?)}";

// Câu lệnh gọi một hàm SQL trên Java
String sql = "{? = call function_name(?,?,?)}";
```

Để làm ví dụ với **CallableStatement** chúng ta cần một hàm hoặc một thủ tục trong DB. Với **Oracle**, **MySQL** hoặc **SQLServer** bạn có thể tạo nhanh một thủ tục như dưới đây:

- **ORACLE**

Get_Employee_Info

```
-- Thủ tục lấy ra thông tin của một nhân viên,  
-- Truyền vào tham số p_Emp_ID (Integer)  
-- Có 4 tham số đầu ra v_Emp_No, v_First_Name, v_Last_Name, v_Hire_Date
```

```
Create Or Replace Procedure  
Get_Employee_Info(p_Emp_Id Integer,v_Emp_No Out Varchar2,v_First_Name  
Out Varchar2,v_Last_Name Out Varchar2,v_Hire_Date Out Date) Is  
Begin  
    v_Emp_No := 'E' || p_Emp_Id;  
    --  
    v_First_Name := 'Michael';  
    v_Last_Name := 'Smith';  
    v_Hire_Date := Sysdate;  
End Get_Employee_Info;
```

- **MySQL**

Get_Employee_Info

```
-- Thủ tục lấy ra thông tin của một nhân viên,  
-- Truyền vào tham số p_Emp_ID (Integer)  
-- Có 4 tham số đầu ra v_Emp_No, v_First_Name, v_Last_Name, v_Hire_Date
```

```
CREATE PROCEDURE get_Employee_Info(p_Emp_ID Integer,  
                                   out v_Emp_No Varchar(50) ,  
                                   out v_First_Name Varchar(50) ,  
                                   Out v_Last_name Varchar(50) ,  
                                   Out v_Hire_date Date)  
BEGIN  
    set v_Emp_No = concat( 'E' , Cast(p_Emp_Id as char(15)) );  
    --  
    set v_First_Name = 'Michael';  
    set v_Last_Name = 'Smith';  
    set v_Hire_date = curdate();  
END
```

- **SQL Server**

Get_Employee_Info

```
-- Thủ tục lấy ra thông tin của một nhân viên,
```

```
-- Truyền vào tham số p_Emp_ID (Integer)
-- Có 4 tham số đầu ra v_Emp_No, v_First_Name, v_Last_Name, v_Hire_Date
```

```
CREATE PROCEDURE Get_Employee_Info
    @p_Emp_Id      Integer,
    @v_Emp_No      Varchar(50)    OUTPUT,
    @v_First_Name  Varchar(50)    OUTPUT,
    @v_Last_Name   Varchar(50)    OUTPUT,
    @v_Hire_Date   Date           OUTPUT
AS
BEGIN
    set @v_Emp_No = 'E' + CAST( @p_Emp_Id as varchar) ;
    --
    set @v_First_Name = 'Michael';
    set @v_Last_Name  = 'Smith';
    set @v_Hire_date  = getdate();
END
```

CallableStatementExample.java

```
package org.plpsoft.tutorial.jdbc.callablestatement;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.Date;
import java.sql.SQLException;

import org.plpsoft.tutorial.jdbc.ConnectionUtils;

public class CallableStatementExample {

    public static void main(String[] args) throws
        ClassNotFoundException,
            SQLException {
        // Lấy ra kết nối tới cơ sở dữ liệu.
        Connection connection = ConnectionUtils.getMyConnection();

        // Câu lệnh gọi thủ tục (***)
        String sql = "{call get_Employee_Info(?,?,?,?,?)}";
```

```
// Tạo một đối tượng CallableStatement.
CallableStatement cstm = connection.prepareCall(sql);

// Truyền tham số vào hàm (p_Emp_ID)
// (Là dấu chấm hỏi thứ 1 trên câu lệnh sql ***)
cstm.setInt(1, 10);

// Đăng ký nhận giá trị trả về tại dấu hỏi thứ 2
// (v_Emp_No)
cstm.registerOutParameter(2,
java.sql.Types.VARCHAR);

// Đăng ký nhận giá trị trả về tại dấu hỏi thứ 3
// (v_First_Name)
cstm.registerOutParameter(3, java.sql.Types.VARCHAR);

// Đăng ký nhận giá trị trả về tại dấu hỏi thứ 4
// (v_Last_Name)
cstm.registerOutParameter(4, java.sql.Types.VARCHAR);

// Đăng ký nhận giá trị trả về tại dấu hỏi thứ 5
// (v_Hire_Date)
cstm.registerOutParameter(5, java.sql.Types.DATE);

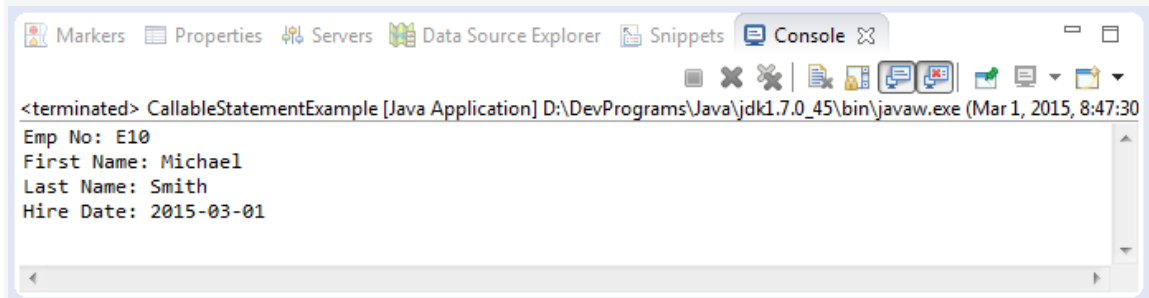
// Thực thi câu lệnh
cstm.executeUpdate();

String empNo = cstm.getString(2);
String firstName = cstm.getString(3);
String lastName = cstm.getString(4);
Date hireDate = cstm.getDate(5);

System.out.println("Emp No: " + empNo);
System.out.println("First Name: " + firstName);
System.out.println("Last Name: " + lastName);
System.out.println("Hire Date: " + hireDate);

}
}
```

Kết quả chạy ví dụ:



8- Điều khiển giao dịch (Transaction)

Giao dịch (Transaction) là một khái niệm quan trọng trong **SQL**.

Ví dụ người A chuyển một khoản tiền 1000\$ vào tài khoản người B như vậy trong Database diễn ra 2 quá trình:

- Trừ số dư tài khoản của người A đi 1000\$
- Thêm vào số dư tài khoản của người B 1000\$.

Và giao dịch được gọi là thành công nếu cả 2 bước kia thành công. Ngược lại chỉ cần 1 trong hai bước hỏng là coi như giao dịch không thành công, phải rollback lại trạng thái ban đầu.

TransactionExample.java

```
package org.plpsoft.tutorial.transaction;

import java.sql.Connection;
import java.sql.SQLException;

import org.plpsoft.tutorial.jdbc.ConnectionUtils;

public class TransactionExample {

    private static void doJob1(Connection conn) {
        // Làm gì đó tại đây.
        // Insert update dữ liệu.
    }

    private static void doJob2(Connection conn) {
        // Làm gì đó tại đây.
        // Insert update dữ liệu.
    }

    public static void main(String[] args) throws
```

```
ClassNotFoundException,  
    SQLException {  
    // Lấy ra kết nối tới cơ sở dữ liệu.  
    Connection connection = ConnectionUtils.getMyConnection();  
  
    // Sét đặt chế độ tự động Commit thành false  
    // Để tự quản lý việc commit trên chương trình.  
    connection.setAutoCommit(false);  
  
    try {  
        // Làm một việc gì đó liên quan tới DB.  
        doJob1(connection);  
        // Làm nhiệm vụ thứ 2  
        doJob2(connection);  
  
        // Gọi method commit dữ liệu xuống DB.  
        connection.commit();  
  
    }  
    // Có vấn đề gì đó lỗi xảy ra.  
    catch (Exception e) {  
        e.printStackTrace();  
        // Rollback dữ liệu  
        connection.rollback();  
    }  
  
    // Đóng Connection.  
    connection.close();  
}  
  
}
```

9- Thực thi một lô lệnh (Batch)

BatchExample.java

```
package org.plpsoft.tutorial.transaction;  
  
import java.sql.Connection;  
import java.sql.SQLException;  
import java.sql.Statement;
```



```
import org.plpsoft.tutorial.jdbc.ConnectionUtils;

public class BatchExample {

    public static void main(String[] args) throws SQLException,
        ClassNotFoundException {

        Connection conn = ConnectionUtils.getMyConnection();

        try{
            // Create statement object
            Statement stmt = conn.createStatement();

            // Set auto-commit to false
            conn.setAutoCommit(false);

            // Create SQL statement
            // Tạo câu lệnh Insert dữ liệu vào bảng Employee
            String sql1 = "Update Employee emp set emp.Salary =
emp.Salary + 100 "
                + " where emp.Dept_Id = 10 ";
            // Add above SQL statement in the batch.
            // Thêm câu lệnh SQL trên vào lô
            stmt.addBatch(sql1);

            // Create one more SQL statement
            String sql2 = "Update Employee emp set emp.Salary =
emp.Salary + 20 "
                + " where emp.Dept_Id = 20 ";
            // Add above SQL statement in the batch.
            // Thêm vào lô
            stmt.addBatch(sql2);

            // Create one more SQL statement
            String sql3 = "Update Employee emp set emp.Salary =
emp.Salary + 30 "
                + " where emp.Dept_Id = 30 ";
            // Add above SQL statement in the batch.
```

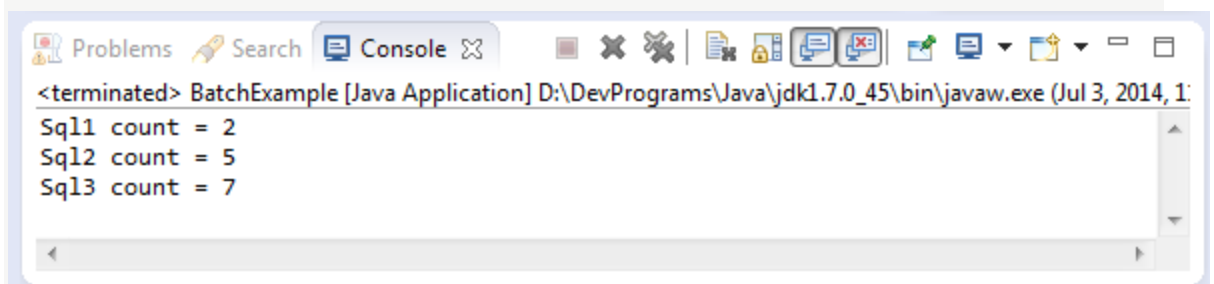
```
// Thêm vào lô
stmt.addBatch(sql3);

// Create an int[] to hold returned values
int[] counts = stmt.executeBatch();

System.out.println("Sql1 count = " + counts[0]);
System.out.println("Sql2 count = " + counts[1]);
System.out.println("Sql3 count = " + counts[2]);

// Explicitly commit statements to apply changes
conn.commit();
} catch (Exception e) {
    e.printStackTrace();
    conn.rollback();
}
}
```

Kết quả chạy ví dụ:



BatchExample2.java

```
package org.plpsoft.tutorial.transaction;

import java.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.UUID;

import org.plpsoft.tutorial.jdbc.ConnectionUtils;

public class BatchExample2 {
```

```
public static void main(String[] args) throws
ClassNotFoundException,
    SQLException {

    Connection conn = ConnectionUtils.getMyConnection();

    try{
        String sql = "Insert into Timekeeper(Timekeeper_Id,
Date_Time, In_Out, Emp_Id) "
            + " values (?, ?, ?, ?) ";
        // Create statement object
        PreparedStatement stmt = conn.prepareStatement(sql);

        // Set auto-commit to false
        conn.setAutoCommit(false);

        // Sét đặt các tham số.
        stmt.setString(1, UUID.randomUUID().toString());
        stmt.setDate(2, new Date(System.currentTimeMillis()));
        stmt.setString(3, "I");
        stmt.setInt(4, 7839);
        // Thêm vào lô.
        stmt.addBatch();

        // Sét đặt các giá trị tham số khác
        stmt.setString(1, UUID.randomUUID().toString());
        stmt.setDate(2, new Date(System.currentTimeMillis()));
        stmt.setString(3, "I");
        stmt.setInt(4, 7566);
        // Thêm vào lô.
        stmt.addBatch();

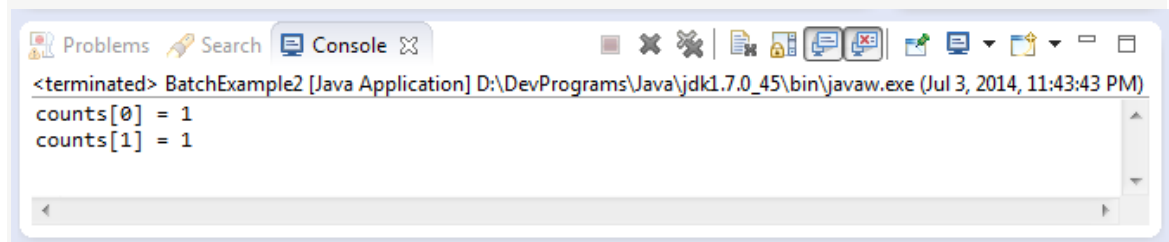
        // Create an int[] to hold returned values
        int[] counts = stmt.executeBatch();

        System.out.println("counts[0] = " + counts[0]);
        System.out.println("counts[1] = " + counts[1]);

        // Explicitly commit statements to apply changes
```

```
        conn.commit();  
    } catch (Exception e) {  
        e.printStackTrace();  
        conn.rollback();  
    }  
}  
}
```

Kết quả chạy ví dụ:



6.3 Connection trong java

Một **Connection trong java** là phiên làm việc giữa ứng dụng java và cơ sở dữ liệu. Đối tượng Connection được sử dụng để tạo Statement, PreparedStatement, và DatabaseMetaData. Giao diện Connection cung cấp nhiều phương thức quản lý transaction như commit(), rollback(), ...

Theo mặc định, connection thực hiện commit sự thay đổi vào database sau việc thực thi các truy vấn.

Các phương thức phổ biến của Connection interface

Dưới đây là các phương thức phổ biến của giao diện Connection trong java:

Phương thức	Mô tả
1) public Statement createStatement():	Tạo một đối tượng Statement được sử dụng để thực thi các câu truy vấn SQL.
2) public Statement createStatement(int resultSetType, int resultSetConcurrency):	Tạo ra một đối tượng Statement sẽ tạo các đối tượng ResultSet với kiểu đã cho và concurrency.
3) public void setAutoCommit(boolean status):	Được sử dụng để thiết lập trạng thái commit. Mặc định là true.
4) public void commit():	Lưu các thay đổi được thực hiện từ khi commit/rollback trước đó.
5) public void rollback():	Loại bỏ tất cả các thay đổi được thực hiện kể từ lần commit/rollback trước đó.
6) public void close():	Đóng kết nối và giải phóng tài nguyên JDBC ngay lập tức.

6.4 Statement trong java

Statement interface trong java cung cấp các phương thức để thực thi các câu lệnh truy vấn với cơ sở dữ liệu SQL. Statement interface là một nhà máy của ResultSet, tức là nó cung cấp phương thức để tạo ra đối tượng ResultSet.

Các phương thức phổ biến của Statement interface

Dưới đây là các phương thức phổ biến của Statement interface:

Phương thức	Mô tả
1) public ResultSet executeQuery(String sql):	được sử dụng để thực hiện truy vấn SELECT. Nó trả về đối tượng của ResultSet.
2) public int executeUpdate(String sql):	được sử dụng để thực thi câu truy vấn được chỉ định, nó có thể là create, drop, insert, update, delete, ...
3) boolean execute(String sql):	được sử dụng để thực thi các câu truy vấn trả về nhiều kết quả.
4) int[] executeBatch():	được sử dụng để thực thi một tập các lệnh.

Ví dụ sử dụng Statement trong java

Dưới đây là ví dụ INSERT và SELECT sử dụng Statement trong java với MySQL:

```
package vn.plpsoft.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ConnectMysqlExample {
    private static String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    private static String USER_NAME = "root";
    private static String PASSWORD = "1234567890";

    /**
     * main
     *
     * @author plpsoft.vn
     * @param args
     */
    public static void main(String args[]) {
        try {
```

```
// connect to database 'testdb'
Connection conn = getConnection(DB_URL, USER_NAME, PASSWORD);
// create statement
Statement stmt = conn.createStatement();
// insert 'student'
stmt.executeUpdate("INSERT INTO student(id, name, address) "
    + "VALUES (5, 'Vinh', 'Hanoi')");
// get data from table 'student'
ResultSet rs = stmt.executeQuery("SELECT * FROM student");
// show data
while (rs.next()) {
    System.out.println(rs.getInt(1) + " " + rs.getString(2)
        + " " + rs.getString(3));
}
// close connection
conn.close();
} catch (Exception ex) {
    ex.printStackTrace();
}
}

/**
 * create connection
 *
 * @author plpsoft.vn
 * @param dbURL: database's url
 * @param userName: username is used to login
 * @param password: password is used to login
 * @return connection
 */
public static Connection getConnection(String dbURL, String userName,
    String password) {
    Connection conn = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        conn = DriverManager.getConnection(dbURL, userName,
password);
        System.out.println("connect successfully!");
    }
```

```
        } catch (Exception ex) {  
            System.out.println("connect failure!");  
            ex.printStackTrace();  
        }  
        return conn;  
    }  
}
```

Kết quả:

```
connect successfully!  
1  Công   Hanoi  
2  Dung   Vinhphuc  
3  Ngôn   Danang  
4  Hạnh   Hanoi  
5  Vinh   Hanoi
```

6.5 ResultSet trong java

Đối tượng của ResultSet duy trì một con trỏ trỏ đến một hàng của một bảng. Ban đầu, con trỏ trỏ đến hàng đầu tiên.

Theo mặc định, đối tượng **ResultSet** chỉ có thể di chuyển về phía trước và nó không thể cập nhật được.

Nhưng chúng ta có thể làm cho đối tượng này di chuyển hướng chuyển tiếp và ngược lại bằng cách truyền một trong hai kiểu **TYPE_SCROLL_INSENSITIVE** hoặc **TYPE_SCROLL_SENSITIVE** trong phương thức **createStatement(int, int)** cũng như chúng ta có thể làm cho đối tượng này có thể cập nhật được bằng cách:

```
Statement stmt =  
conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,  
                      ResultSet.CONCUR_UPDATABLE);
```

Các phương thức phổ biến của giao diện ResultSet trong java

Dưới đây là các phương thức phổ biến của giao diện **ResultSet** trong java:

Phương thức	Mô tả
1) public boolean next():	được sử dụng để di chuyển con trỏ đến một hàng tiếp theo từ vị trí hiện tại.

2) public boolean previous():	được sử dụng để di chuyển con trỏ đến một hàng trước đó từ vị trí hiện tại.
3) public boolean first():	được sử dụng để di chuyển con trỏ đến hàng đầu tiên trong đối tượng thiết lập kết quả.
4) public boolean last():	được sử dụng để di chuyển con trỏ đến hàng cuối cùng trong đối tượng thiết lập kết quả.
5) public boolean absolute(int row):	được sử dụng để di chuyển con trỏ đến số hàng được chỉ định trong đối tượng ResultSet.
6) public boolean relative(int row):	được sử dụng để di chuyển con trỏ đến số hàng tương đối trong đối tượng ResultSet, nó có thể là dương hoặc âm.
7) public int getInt(int columnIndex):	được sử dụng để trả về dữ liệu của chỉ mục cột được chỉ định của hàng hiện tại như int.
8) public int getInt(String columnName):	được sử dụng để trả lại dữ liệu của tên cột được chỉ định của hàng hiện tại như int.
9) public String getString(int columnIndex):	được sử dụng để trả về dữ liệu của chỉ mục cột được chỉ định của dòng hiện tại như String.
10) public String getString(String columnName):	được sử dụng để trả lại dữ liệu của tên cột được chỉ định của hàng hiện tại như String.

Ví dụ về Scrollable ResultSet trong java

Ví dụ dưới đây in ra màn hình row thứ 3 của kết quả truy vấn:

```
package vn.plpsoft.jdbc;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class ResultSetExample {
    private static String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    private static String USER_NAME = "root";
```

```
private static String PASSWORD = "1234567890";

public static void main(String[] args) {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection(DB_URL,
            USER_NAME, PASSWORD);
        // create statement
        Statement stmt =
            conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                ResultSet.CONCUR_UPDATABLE);
        // get data from table 'student'
        ResultSet rs = stmt.executeQuery("SELECT * FROM student");
        // getting the record of 3rd row
        rs.absolute(3);
        System.out.println(rs.getInt(1) + " " + rs.getString(2)
            + " " + rs.getString(3));
    } catch (Exception ex) {
        System.out.println("connect failure!");
        ex.printStackTrace();
    }
}
```

Kết quả:

```
3  Ngôn  Danang
```

6.6 PreparedStatement trong java

Giao diện **PreparedStatement trong java** là một sub-interface của Statement. Nó được sử dụng để thực hiện truy vấn tham số.

```
String sql = "INSERT INTO student VALUES(?, ?, ?)";
```

Như bạn thấy, chúng ta truyền tham số (?) cho các giá trị. Giá trị của nó sẽ được cài đặt bằng cách gọi các phương thức setter của PreparedStatement.

Tại sao sử dụng PreparedStatement?

Cải thiện hiệu suất : Hiệu suất của ứng dụng sẽ nhanh hơn nếu bạn sử dụng giao diện PreparedStatement vì truy vấn được biên dịch chỉ một lần.

Làm thế nào để có được đối tượng PreparedStatement.

Phương thức `prepareStatement()` của giao diện `Connection` được sử dụng để trả về đối tượng `PreparedStatement`. Cú pháp:

```
public PreparedStatement prepareStatement(String  
query) throws SQLException;
```

Các phương thức của PreparedStatement interface

Dưới đây là các phương thức quan trọng của `PreparedStatement` interface trong java:

Phương thức	Mô tả
<code>public void setInt(int paramIndex, int value)</code>	đặt giá trị số nguyên cho chỉ số tham số đã cho.
<code>public void setString(int paramIndex, String value)</code>	đặt giá trị String cho chỉ số tham số đã cho.
<code>public void setFloat(int paramIndex, float value)</code>	đặt giá trị float vào chỉ số tham số đã cho.
<code>public void setDouble(int paramIndex, double value)</code>	đặt giá trị double vào chỉ số tham số đã cho.
<code>public int executeUpdate()</code>	thực hiện truy vấn. Nó được sử dụng để create, drop, insert, update, delete, vv.
<code>public ResultSet executeQuery()</code>	thực hiện truy vấn chọn. Nó trả về một thể hiện của <code>ResultSet</code> .

Ví dụ về PreparedStatement trong java

Tạo bảng 'student' trong cơ sở dữ liệu có tên 'testdb' trong MySQL với câu lệnh như sau:

```
CREATE TABLE student (  
    id    INT        NOT NULL,  
    name VARCHAR (32)    NOT NULL,  
    address  VARCHAR (32) NOT NULL,  
    PRIMARY KEY (id)
```

```
);
```

Tạo chương trình insert và select student:

```
package vn.plpsoft.jdbc;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
public class PreparedStatementExample {
```

```
    private static String DB_URL = "jdbc:mysql://localhost:3306/testdb";
```

```
    private static String USER_NAME = "root";
```

```
    private static String PASSWORD = "1234567890";
```

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

```
        String sqlInsert = "INSERT INTO student VALUES(?, ?, ?)";
```

```
        String selectAll = "SELECT * FROM student";
```

```
        try {
```

```
            // connect to database
```

```
            Class.forName("com.mysql.jdbc.Driver");
```

```
            Connection conn = DriverManager.getConnection(DB_URL,  
                USER_NAME, PASSWORD);
```

```
            // crate statement to insert student
```

```
            PreparedStatement stmt = conn.prepareStatement(sqlInsert);
```

```
            stmt.setInt(1, 1);
```

```
            stmt.setString(2, "Vinh");
```

```
            stmt.setString(3, "Hanoi");
```

```
            stmt.execute();
```

```
            // select all student
```

```
            stmt = conn.prepareStatement(selectAll);
```

```
            // get data from table 'student'
```

```
            ResultSet rs = stmt.executeQuery();
```

```
            // show data
```

```
            while (rs.next()) {
```

```
                System.out.println(rs.getInt(1) + " " + rs.getString(2)
```

```
        + " " + rs.getString(3));  
    }  
    stmt.close();  
    conn.close();  
} catch (SQLException ex) {  
    ex.printStackTrace();  
} catch (ClassNotFoundException e) {  
    e.printStackTrace();  
}  
}  
}
```

Kết quả:

```
1  Vinh  Hanoi
```

6.7 ResultSetMetaData trong java

Giao diện **ResultSetMetaData** trong java được sử dụng để lấy ra các metadata từ đối tượng ResultSet.

Siêu dữ liệu (metadata) là các thông tin của dữ liệu. Ví dụ, một metadata của file là ngày tháng tạo file, dung lượng của file, ... Metadata của một bảng trong CSDL là các thông tin về bảng như tên bảng, tên cột, kiểu giá trị của cột, ...

Làm thế nào để có được đối tượng ResultSetMetaData

Phương thức `getMetaData()` của giao diện ResultSet trả về đối tượng ResultSetMetaData. Cú pháp:

```
ResultSetMetaData getMetaData() throws SQLException;
```

Các phương thức phổ biến của giao diện ResultSetMetaData

Dưới đây là các phương thức phổ biến của giao diện ResultSetMetaData trong java:

Method	Description
<code>public int getColumnCount()throws SQLException</code>	nó trả về tổng số các cột trong đối tượng ResultSet.
<code>public String getColumnName(int index)throws SQLException</code>	nó trả về tên cột của chỉ số cột được chỉ định.

<code>public String getColumnTypeName(int index) throws SQLException</code>	nó trả về tên cột cho chỉ số đã chỉ định.
<code>public String getTableName(int index) throws SQLException</code>	nó trả về tên bảng cho chỉ số cột được chỉ định.

Ví dụ về ResultSetMetaData interface

Tạo bảng 'student' trong cơ sở dữ liệu có tên 'testdb' trong MySQL với câu lệnh như sau

```
CREATE TABLE student (  
    id    INT        NOT NULL,  
    name VARCHAR (32)    NOT NULL,  
    address  VARCHAR (32) NOT NULL,  
    PRIMARY KEY (id)  
);
```

Chương trình sau lấy ra một vài metadata của bảng 'student':

```
package vn.plpsoft.jdbc;  
  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.ResultSetMetaData;  
import java.sql.SQLException;  
  
public class ResultSetMetaDataExample {  
    private static String DB_URL = "jdbc:mysql://localhost:3306/testdb";  
    private static String USER_NAME = "root";  
    private static String PASSWORD = "1234567890";  
  
    public static void main(String[] args) {  
        String selectAll = "SELECT * FROM student";  
        try {  
            // connect to database  
            Class.forName("com.mysql.jdbc.Driver");  
            Connection conn = DriverManager.getConnection(DB_URL,  
                USER_NAME, PASSWORD);
```

```
// crate statement
PreparedStatement stmt = conn.prepareStatement(selectAll);
// get data from table 'student'
ResultSet rs = stmt.executeQuery();
// get metadata from rs
ResultSetMetaData rsmd = rs.getMetaData();

// show metadata
System.out.println("Tong so column cua bang 'student': "
    + rsmd.getColumnCount());
System.out.println("Ten column thu 2: "
    + rsmd.getColumnName(2));
System.out.println("Column type cua column thu 2: "
    + rsmd.getColumnTypeName(2));

stmt.close();
conn.close();
} catch (SQLException ex) {
    ex.printStackTrace();
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
}
```

Kết quả:

```
Tong so column cua bang 'student': 3
Ten column thu 2: name
Column type cua column thu 2: VARCHAR
```

6.8 DatabaseMetaData trong java

Giao diện **DatabaseMetaData** trong java cung cấp các phương thức để lấy metadata của cơ sở dữ liệu như tên sản phẩm cơ sở dữ liệu, phiên bản sản phẩm cơ sở dữ liệu, tên driver, tên của tổng số bảng, tên của tổng số các view,...

Làm thế nào để có được đối tượng DatabaseMetaData

Phương thức `getMetaData()` của Giao diện `Connection` trả về đối tượng `DatabaseMetaData`. Cú pháp:

```
1 public DatabaseMetaData getMetaData() throws SQLException;
```

Ví dụ về DatabaseMetaData trong java

Ví dụ 1:

```
package vn.plpsoft.jdbc;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;

public class DatabaseMetaDataExample {
    private static String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    private static String USER_NAME = "root";
    private static String PASSWORD = "1234567890";

    public static void main(String[] args) {
        try {
            // connect to database
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn = DriverManager.getConnection(DB_URL,
                USER_NAME, PASSWORD);
            // get DatabaseMetaData object
            DatabaseMetaData dbmd = conn.getMetaData();

            // show metadata of database
            System.out.println("Driver Name: " + dbmd.getDriverName());
            System.out.println("Driver Version: " +
                dbmd.getDriverVersion());
            System.out.println("UserName: " + dbmd.getUserName());
            System.out.println("Database Product Name: "
                + dbmd.getDatabaseProductName());
            System.out.println("Database Product Version: "
                + dbmd.getDatabaseProductVersion());

            conn.close();
        } catch (Exception ex) {
            System.out.println("connect failure!");
        }
    }
}
```



```
        ex.printStackTrace();
    }
}
```

Kết quả:

```
Driver Name: MySQL-AB JDBC Driver
Driver Version: mysql-connector-java-5.1.6 ( Revision: ${svn.Revision}
)
UserName: root@localhost
Database Product Name: MySQL
Database Product Version: 5.5.5-10.1.21-MariaDB
```

Ví dụ 2: in ra màn hình tên các bảng của database:

```
package vn.plpsoft.jdbc;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.ResultSet;

public class DatabaseMetaDataExample2 {
    private static String DB_URL = "jdbc:mysql://localhost:3306/testdb";
    private static String USER_NAME = "root";
    private static String PASSWORD = "1234567890";

    public static void main(String[] args) {
        try {
            // connect to database
            Class.forName("com.mysql.jdbc.Driver");
            Connection conn = DriverManager.getConnection(DB_URL,
                USER_NAME, PASSWORD);
            // get DatabaseMetaData object
            DatabaseMetaData dbmd = conn.getMetaData();

            // show list table database
            String table[] = { "TABLE" };
            ResultSet rs = dbmd.getTables(null, null, null, table);
            while (rs.next()) {
                System.out.println(rs.getString(3));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```
    }

    conn.close();
} catch (Exception ex) {
    System.out.println("connect failure!");
    ex.printStackTrace();
}
}
```

Kết quả:

```
sessions
student
```

Chương 7: Project quản lý sinh viên (Giáo trình thực hành)