BÀI 3: LẬP TRÌNH GIAO DIỆN VỚI SWING CƠ BẢN

Học xong bài này người học sẽ:

- Hiểu cấu trúc và nắm vững kỹ năng lập trình tổ chức một giao diện người dùng đồ họa trong Ngôn ngữ lập trình Java với các lớp thuộc 2 gói awt và swing.
- Thiết kế các thành phần là vật chưa các thành phần khác: JFrame, JWindow, JPanel
- Thiết kế và quản lý bố cục giáo diện với các lớp Layout: BoxLayout, BorderLayout, FlowLayout, GridLayout, GridBagLayout, …
- Lập trình tùy biến các đối tượng GUI trong thư viện Swing: Jlabel, JtextField, JPasswordField, JTextArea, JButton.
- Hiểu cơ chế xử lý sự kiện tương tác người dùng và nắm vũng kỹ năng lập trình phát triển các đối tượng xử lý sự kiện với các lớp: Event, Listener, Adapter.
- Kỹ năng tự tìm hiểu và sử dụng các GUI component có sẵn trong thư viện.
- Thiết kế giao diện người dùng trên công cụ trực quan NetBeans IDE.

3.1. GIỚI THIỆU VỀ LẬP TRÌNH GIAO DIỆN

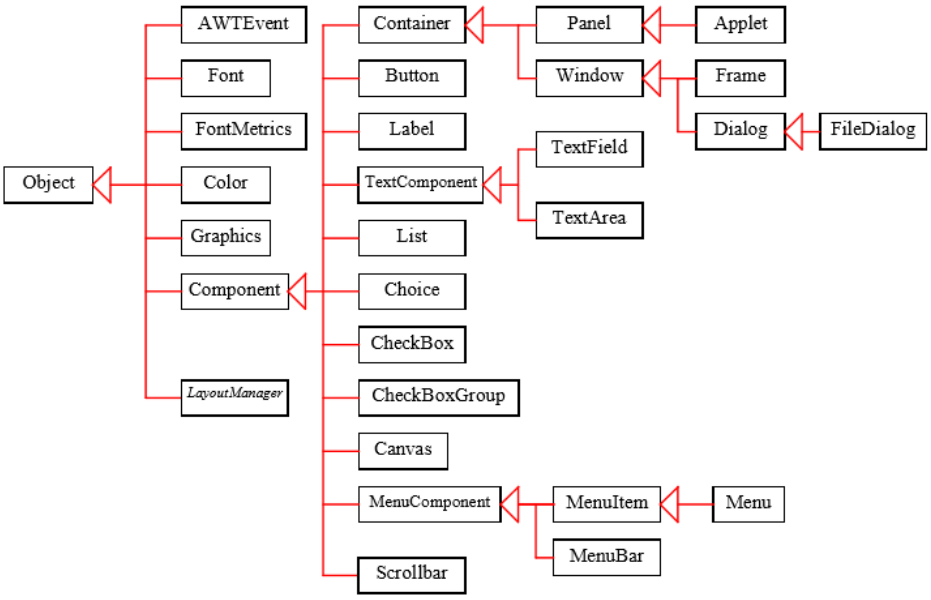
3.1.1. Giới thiệu giao diện người dùng đồ họa

Các ứng dụng phần mềm được trình bày trên nhiều màn hình giao diện đồ họa đẹp mắt. Ngôn ngữ lập trình Java cung cấp các đối tượng đồ họa để lập trình giao diện người dùng đồ họa (Graphical User Interface - GUI).

GUI có thể chứa nhiều điều khiển như textbox, label, listbox ... Một thành phần (component) GUI là một đối tượng trực quan. Người dùng tương tác với đối tượng này thông qua con trỏ chuột hay bàn phím. Ta cần sử dụng các lớp trong gói java.awt hoặc javax.swing

3.1.2. Các lớp thư viện gói AWT

AWT (Abstract Windows Toolkit) là thư viện API cung cấp các đối tượng GUI, là các lớp thuộc gói java.awt. Gói AWT chứa các lớp, giao diện và các gói con.

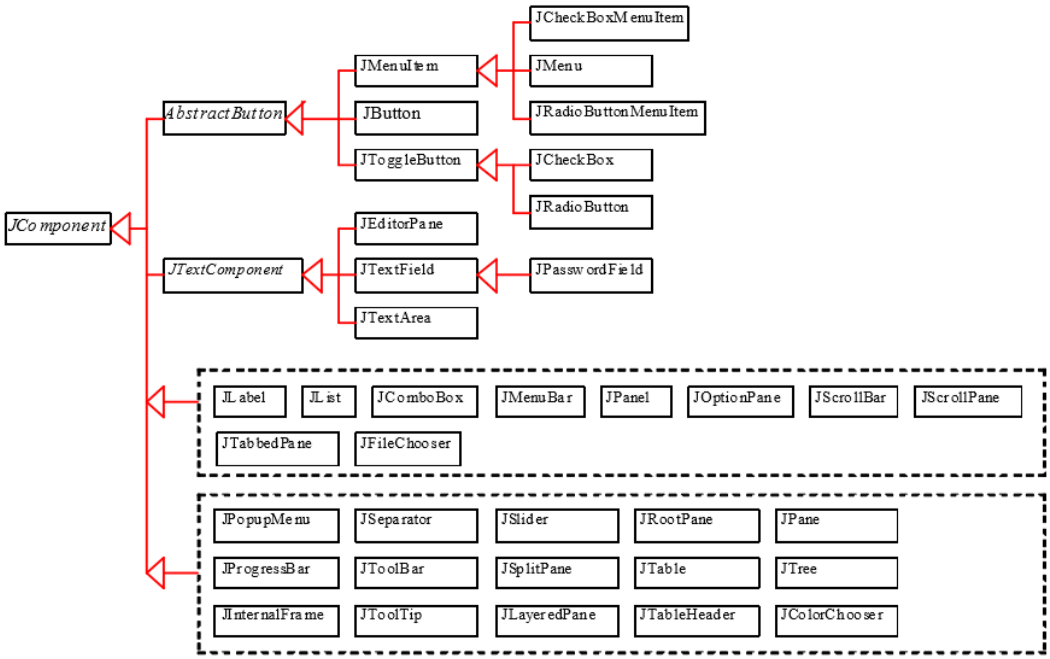


Hình 3.1: Sơ đồ phân cấp trong AWT

3.1.3. Các lớp thư viện gói Swing

Java Foundation Classes (JFC) được được giới thiệu trong phiên bản 2.0 (Java Development Kit – JDK 2.0), là một framework hỗ trợ lập trình giao diện đồ hoạ (Graphical Interface) với thư viện Swing.

Swing là một sự mở rộng AWT, một framework được thiết kế theo mô hình MVC (Model View Controller), hỗ trợ công nghệ gọi là "Pluggable-Look-And-Feel" cho phép các thành phần giao diện nhất quán độc lập với môi trường (cross-platform GUI), có thể được hiển thị trên bất ký hệ điều hành nào như Windows, Mac OS, Linux, … Swing thuộc package javax.swing.



Hình 3.2: Sơ đồ phân cấp trong Swing

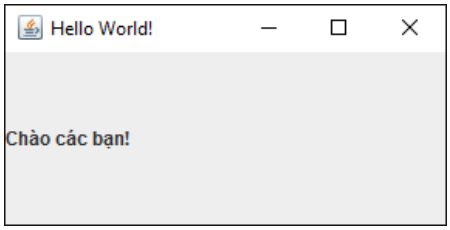
Một số điểm khác nhau giữa AWT và Swing:

Java AWT	Java Swing
Các thành **phụ thuộc nền tảng**	Các thành phần **độc lập nền tảng**
Các thành phần **nặng**	Các thành phần **gọn nhẹ**
Không hỗ trợ pluggable L&F **(Look and Feel)**	Hỗ trợ pluggable L&F
Cung cấp **ít thành phần hơn**	Cung cấp các thành phần mạnh mẽ như table, list, scrollpanes, colorchooser, ...
Không theo sau MVC, model biểu diễn dữ liệu, view biểu diễn sự trình bày và controller xử lý hoạt động	**Theo sau** MVC

Bảng 3.1: Điểm khác nhau giữa AWT và Swing

Ví dụ: Chương trình sau tạo một Cửa sổ với "Hello World" trong thanh tiêu đề

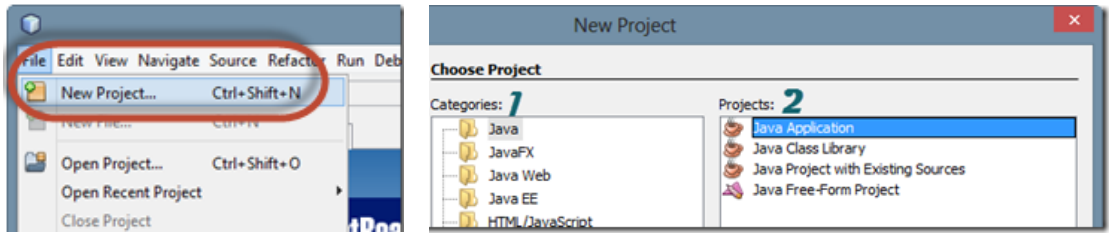
```
import javax.swing.*;
public class HelloApp {
   public static void main(String[] args){
      JFrame myFrame = new JFrame("Hello World!");
      myFrame.setSize(300, 150); //kích thước JFrame
      myFrame.setVisible(true);
      JLabel lbName=new JLabel();
      lbName.setText("Chào các bạn!");
      myFrame.add(lbName);// Thêm jLabel vài JFrame
   }
}
```



Hình 3.3: Kết quả chương trình

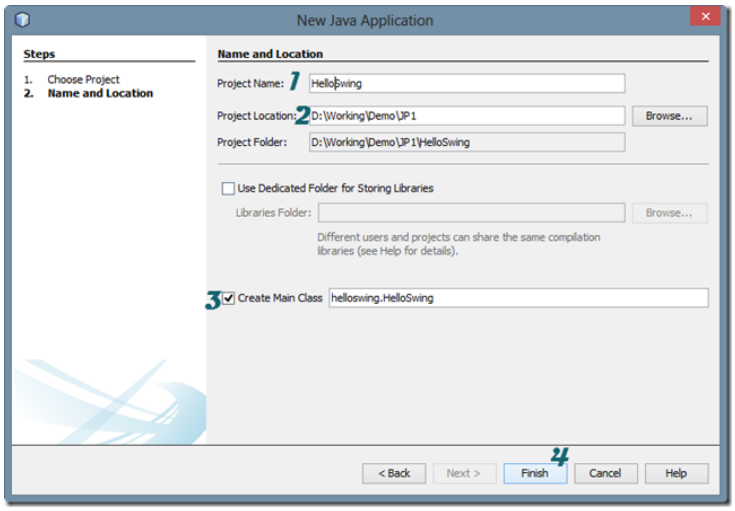
Tạo ứng dụng Swing dùng NetBeans:

- Khởi động Netbean, tạo 1 project cho ứng dụng Java: chọn menu ***File-New Project…***



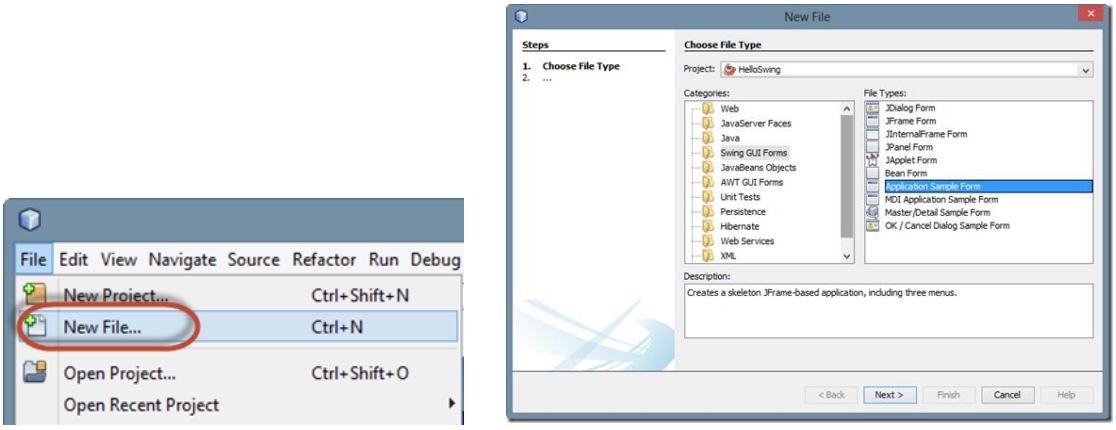
Hình 3.4: Tạo mới Project

- Trong hộp thoại ***New Project*** , mục Categories (1) chọn Java, mục Projects (2) chọn Java Application, Xong chọn nút ***Next*** để chuyển sang màn hình kế tiếp
- Trong màn hình New Java Application
    - Mục Project Name (1): Đặt tên cho Project
    - Mục Project Location (2): Chọn nơi lưu Project
    - Mục Create Main Class (3): Bỏ chọn (không có dấu check)
    - Chọn Finish (4) để hoàn thành việc tạo ra Project cho ứng dụng Java



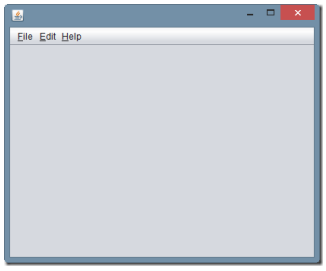
Hình 3.5: Đặt tên và chọn vị trí lưu Project

- Tạo Form Swing có sẵn hệ thống Menu: Chọn ***File-New File…***



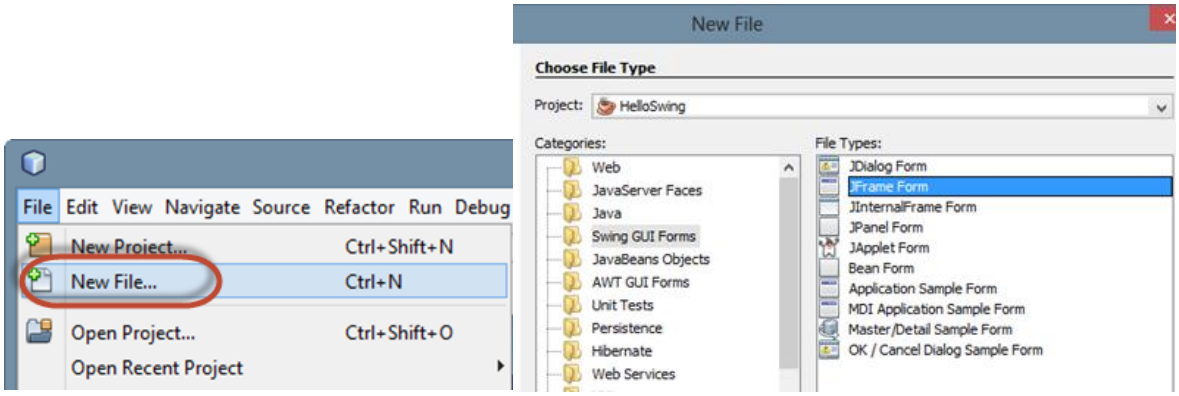
Hình 3.6: Tạo mới File kiểu Form mẫu

- Trong màn hình New File, mục Categories chọn ***Swing GUI Forms***, mục File Types chọn***Application Sample Form***, xong chọn nút ***Next***
- Mỗi cửa sổ Swing trong Java là một class thừa kế từ lớp JFrame, trong màn ***New Application Sample Form***
    - Mục Class Name: Đặt tên cho Form
    - Chọn nút Finish để hoàn thành việc tạo Form
- Run Project để xem kết quả:



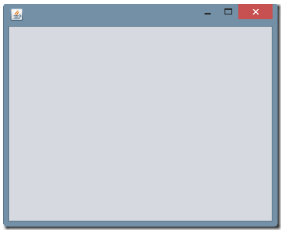
Hình 3.7: Kết quả Form theo mẫu

- Tạo Form trống: chọn ***File-New File…***



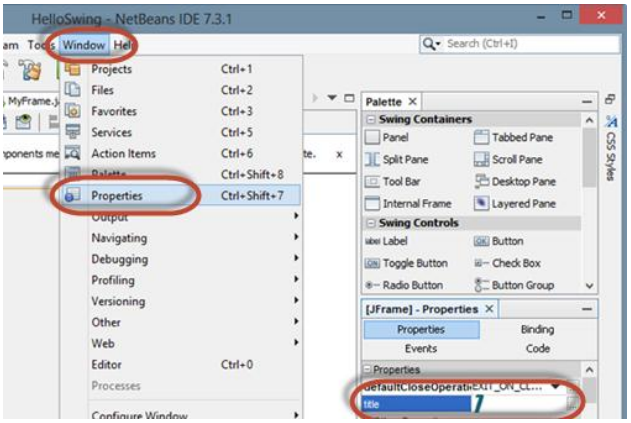
Hình 3.8: Tạo mới File kiểu Form trống

- Trong màn New File: Mục Categories chọn ***Swing GUI Forms***, mục File Types chọn ***JFrame Form***, chọn nút ***Next*** :
- Trong màn ***New Application Sample Form***
  - Mục Class Name: đặt tên cho Form
  - Xong chọn nút Finish để hoàn thành việc tạo Form
- Run Project để xem kết quả:



Hình 3.9: Kết quả Form theo mẫu

- Để thay đổi Title (phần tiêu đề) cho Swing Form:
  - Trong cửa sổ Properties (menu ***Window-Properties***)
  - Trong phần Properties, tìm đến mục title (1): nhập tiêu đề của Form



Hình 3.10: Thay đổi tiêu đề Form

3.2. CẤU TRÚC CHUNG CỦA CÁC GIAO DIỆN NGƯỜI DÙNG

Cấu trúc giao diện người dùng: Tổ chức các đối tượng GUI trên một vật chứa (container) quản lý bởi trình quản lý bố cục (layout manager).

3.2.1. Các lớp Container

Container là thành phần được sử dụng để chứa các thành phần khác:

Loại	Tên	Mô tả
Top Level	JApplet, JDialog, Jframe,…	Các Container cấp cao: Thành phần này luôn xuất hiện trong ứng dụng và được sử dụng để chứa các thành phần khác.
General Purpose	JPanel, JScrollPane, JTabbedPane, JtoolBar,…	Container trung gian phổ biến
Special Purpose	JInternalFrame, JLayeredPane, JRootPane	Container đặc biệt

Bảng 3.2: Các lớp Container

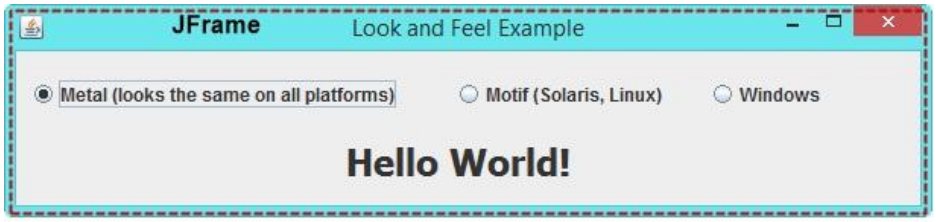
Các lớp container cấp cao: Quản lý gián tiếp các container trung gian cho phép tổ chức giao diện theo chiều sâu Z-oderring. Các container cấp cao phổ biến gồm:

- JFrame là 1 cửa sổ có khung, có tiêu đề và các nút điều khiển dùng hiển thị giao diện ứng dụng Java độc lập
- JDialog là hộp thoại có viền khung và tiêu đề, phải có 1 frame hoặc dialog làm owner, tự động thu nhỏ hoặc hiển thị theo owner.
- JApplet cho phép tạo giao diện GUI cho ứng dụng nhúng.

Các container trung gian: sử dụng trình quản lý layout manager để bố trí và quản lý các đối tượng trên GUI: JPanel, JScrollPane, JTabbedPane….

3.2.1.1. Lớp JFrame

JFrame là top level container và được sử dụng để chứa các thành phần khác như JPanel, JTabbedPane, JToolBar, …



Hình 3.11: Cấu trúc JFrame

Một số thuộc tính thường dùng đối với thành phần Jframe:

Tên phương thức	Mô tả
Title	Tiêu đề màn hình
defaultCloseOperation	Thiết lập xử lý khi chọn nút dấu X ở góc trên phải
iconImage	Thiết lập icon ở góc trên bên trái của màn hình
Resizable	Cho phép điều chỉnh kích thước màn hình hay không

Bảng 3.3: Các thuộc tính thường dùng của JFrame

Lớp JFrame kế thừa từ java.awt.Frame, 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

Lớp JFrame này có các constructor sau:

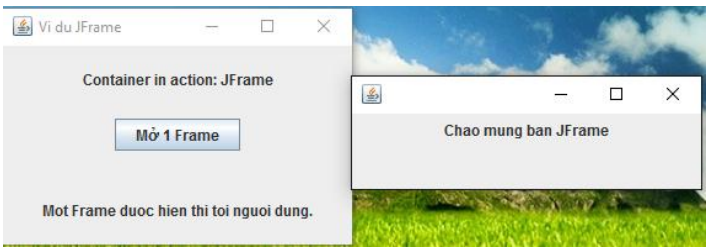
- **JFrame():** Xây dựng một Frame mới, ban đầu là không nhìn thấy (invisible).
- **JFrame(GraphicsConfiguration gc):** Tạo một Frame trong GraphicsConfiguration đã cho của một thiết bị màn hình và một title trống.
- **JFrame(String title):** Tạo một Frame mới, ban đầu là không nhìn thấy (invisible)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

Ví dụ minh họa lớp JFrame:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SwingContainerJFrameDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  private JLabel msglabel;
  public SwingContainerJFrameDemo (){
    prepareGUI();
  }
```

```
public static void main(String[] args){
  SwingContainerJFrameDemo   swingContainerDemo =
                                        new SwingContainerJFrameDemo ();
  swingContainerDemo.showJFrameDemo();
}
private void prepareGUI(){
  mainFrame = new JFrame("Vi du JFrame");
  mainFrame.setSize(300,200);
  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(150,100);
  msglabel = new JLabel("Chao mung JFrame", 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("Mở 1 Frame");
  okButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      statusLabel.setText("Mot Frame duoc hien thi.");
      frame.setVisible(true);
    }
  });
```

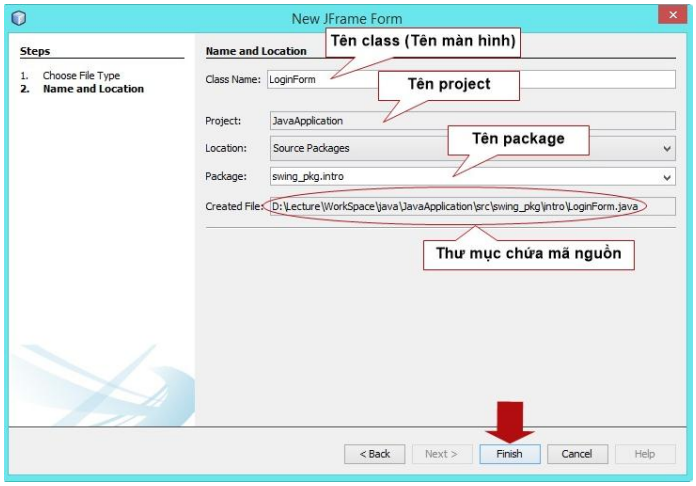
```
        controlPanel.add(okButton);
        mainFrame.setVisible(true);
    }
}
```



Hình 3.12: Kết quả minh họa Jframe

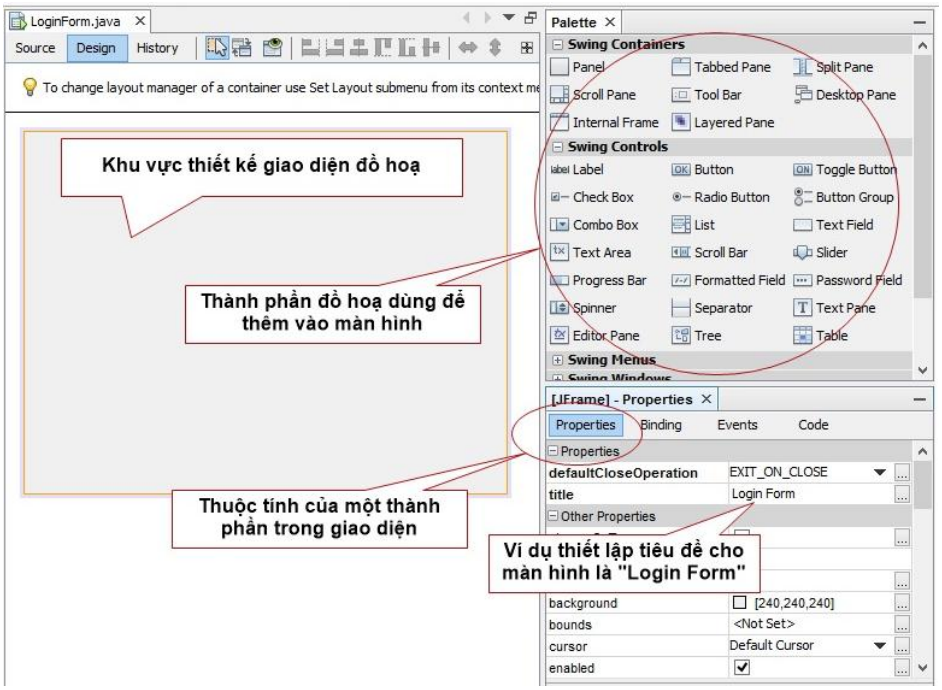
Cách tạo JFrame sử dụng NetBeans:

- Tại cửa sổ Project -> chuột phải -> chọn **New ->** chọn **JFrame Form ->** nhập tên class tại **Class Name ->** chọn **Finish** để kết thúc.



Hình 3.13: Tạo mới Jframe trên NetBean

- Kết quả thu được sau khi nhấn nút "Finish"



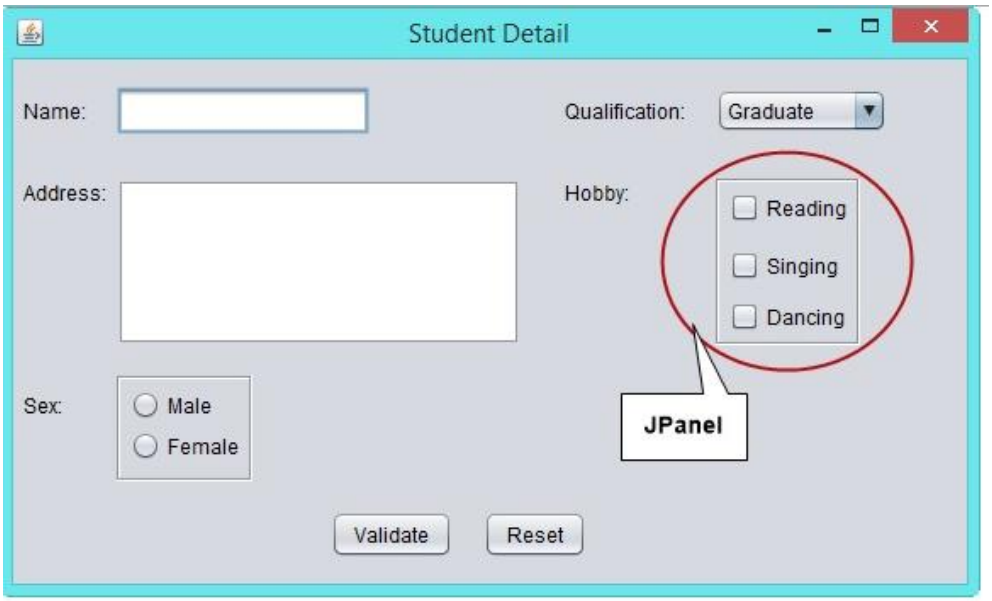
Hình 3.14: Kết quả thiết kế Jframe

3.2.1.2. Lớp JPanel

JPanel vừa là một container vì nó được sử dụng để chứa các thành phần khác, vừa là một thành phần (component) vì được chứa trong một JFrame.

Không giống như JFrame, JPanel không có title, không có các nút điều khiển (minimum button, maximum button, close button) và đặc biệt JPanel không thể sử dụng độc lập.

Đầu tiên chúng ta thêm các thành phần vào JPanel và sau đó thêm JPanel vào top level như JFrame



Hình 3.15: Minh họa Jpanel

Cú pháp khai báo cho lớp javax.swing.JPanel là:

public class JPanel extends JComponent implements Accessible

Các Constructor:

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

Các Phương thức:

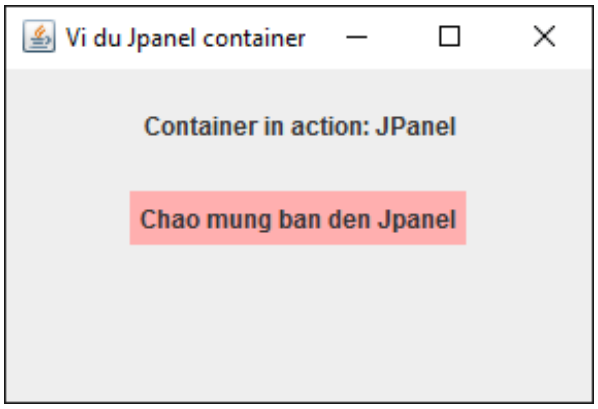
- **AccessibleContext getAccessibleContext():** Lấy AccessibleContext được liên kết với JPanel này.
- **PanelUI getUI():** Trả về đối tượng L&F mà truyền thành phần này

- **String getUIClassID():** Trả về một chuỗi xác định tên của lớp L&F mà truyền thành phần này
- **protected String paramString():** Trả về một biểu diễn chuỗi của JPanel này
- **void setUI(PanelUI ui):** Thiết lập đối tượng L&F mà truyền thành phần này
- **void updateUI():** Phục hồi thuộc tính UI về một giá trị Look và Feel hiện tại.

Ví dụ minh họa lớp Jpanel:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SwingContainerJpanelDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  private JLabel msglabel;
  public SwingContainerJpanelDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    SwingContainerJpanelDemo  swingContainerDemo =
                                                  new SwingContainerJpanelDemo();
    swingContainerDemo.showJPanelDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du Jpanel container");
    mainFrame.setSize(300,200);
    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(150,100);
    msglabel = new JLabel("Chao mung ban den Jpanel", JLabel.CENTER);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
    mainFrame.add(headerLabel);
    mainFrame.add(controlPanel);
    mainFrame.add(statusLabel);
```

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



Hình 3.16: Kết quả minh họa Jpanel Container

3.2.1.3. Lớp JWindow

Lớp JWindow 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

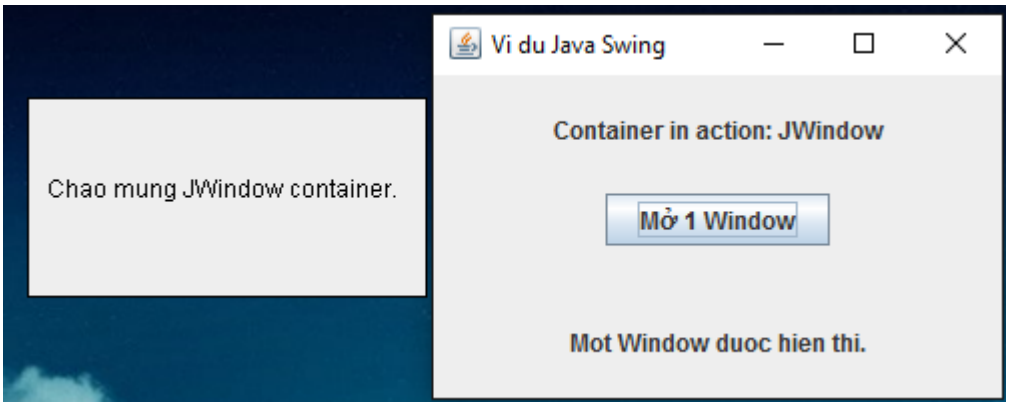
Có các constructor sau:

- **JWindow()**: Tạo một window mà không xác định khung sở hữu nó (owner frame).
- **JWindow(Frame owner)**: Tạo 1 window với owner frame đã cho.
- **JWindow(GraphicsConfiguration gc)**: Tạo 1 window với GraphicsConfiguration đã cho.
- **JWindow(Window owner)**: Tạo 1 window với cửa sổ sở hữu nó đã cho.
- **JWindow(Window owner, GraphicsConfiguration gc)**: Tạo 1 window với cửa sổ sở hữu nó đã cho và GraphicsConfiguration đã cho của một thiết bị màn hìn

Ví dụ minh họa lớp JWindow:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class SwingContainerJWindowDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  private JLabel msglabel;
  public SwingContainerJWindowDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    SwingContainerJWindowDemo  swingContainerDemo =
                                        new SwingContainerJWindowDemo();
    swingContainerDemo.showJWindowDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du Java Swing");
    mainFrame.setSize(300,200);
    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(150,100);
    msglabel = new JLabel("Chao mung JForm Container"
                                                , JLabel.CENTER);
    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 JWindow container.");
    JButton okButton = new JButton("Mở 1 Window");
```

```
        okButton.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            window.setVisible(true);
            statusLabel.setText("Mot Window duoc hien thi.");
          }
        });
        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(200, 100);
          setLocationRelativeTo(parent);
        }
        public void paint(Graphics g)
        {
          super.paint(g);
          g.drawRect(0,0,getSize().width - 1,getSize().height - 1);
          g.drawString(message,10,50);
        }
      }
    }
```



Hình 3.17: Kết quả minh họa JWindow

3.2.2. Trình quản lý Layout

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

public interface LayoutManager

LayoutManager Interface có các phương thức sau:

- **void addLayoutComponent(String name, Component comp):** 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.
- **Dimension preferredLayoutSize(Container parent):** Tính toán các chiều kích cỡ được ưu tiên cho Container đã xác định.
- **void removeLayoutComponent(Component comp):** Xóa thành phần đã cho từ layout.

Giới thiệu về layoutmanager 2 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

LayoutManager 2 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.

Các lớp Layout Manager trong Java Swing:

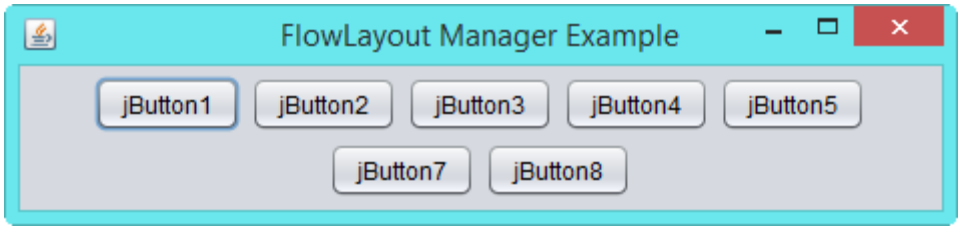
STT	LayoutManager & Mô tả
1	**Lớp BoxLayout**: Sắp xếp các thành phần chiều dọc hoặc ngang
2	**Lớp BorderLayout**: Sắp xếp các thành phần 5 miền: đông, tây, nam, bắc và trung tâm
3	**Lớp CardLayout**: Mỗi thành phần trong Container như là một card. Mỗi card nhìn thấy tại một thời điểm
4	**Lớp FlowLayout:** Layout mặc định. Bố trí các thành phần trong luồng (trong một line, line sau nối tiếp line trước)
5	**Lớp GridLayout**: Quản lý các thành phần trong lưới hình chữ nhật. Một thành phần được hiển thị trong mỗi hình chữ nhật con.
6	**Lớp GridBagLayout**: Quản lý layout linh động. Căn chỉnh các thành phần

	dọc, ngang hoặc baseline mà không yêu cầu các thành phần cùng kích cỡ.
7	**Lớp GroupLayout**: Nhóm các thành phần theo cấu trúc thứ bậc để đặt chúng trong một Container
8	**Lớp SpringLayout**: Đặt vị trí các con của Container liên kết với nó tuân theo một tập hợp các ràng buộc.

Bảng 3.4: Các lớp Layout

3.2.2.1. FlowLayout

FlowLayout cho phép add các control trên cùng một dòng, khi nào hết chỗ chứa sẽ tự động xuống dòng, có thể điều chỉnh hướng xuất hiện của control.



Hình 3.18: Cách trình bày của FlowLayout

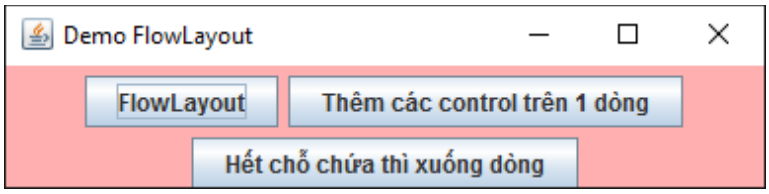
Mặc định khi một JPanel được khởi tạo thì bản thân lớp chứa này sẽ có kiểu Layout là FlowLayout. JPanel giống như thùng đựng đồ vật, từng đồ vật là các control …Ta nên tạo JPanel để add các control vào JPanel để việc quản lý control được dễ dàng hơn.

Chương trình sau minh họa về FlowLayout:

```
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class MyFlowLayout extends JFrame{
   public MyFlowLayout(String title){
      setTitle(title);
      JPanel pnFlow=new JPanel();
      pnFlow.setLayout(new FlowLayout());
      pnFlow.setBackground(Color.PINK);//set màu nên cho JPanel
      JButton btn1=new JButton("FlowLayout");
      JButton btn2=new JButton("Thêm các control trên 1 dòng");
      JButton btn3=new JButton("Hết chỗ chứa thì xuống dòng");
      pnFlow.add(btn1);//ad Jbutton vào JPanel
      pnFlow.add(btn2);
      pnFlow.add(btn3);
      Container con=getContentPane();//lấy lớp chứa của cửa sổ windows
      con.add(pnFlow);// add lớp chứa JPanel vào cửa sổ
   }
```

```
    public static void main(String[] args){
        MyFlowLayout myUI=new MyFlowLayout("Demo FlowLayout");
        myUI.setSize(400, 100);
        // thiết lập sự kiện đóng cửa sổ
        myUI.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        myUI.setLocationRelativeTo(null);//Nằm giữa màn hình
        myUI.setVisible(true);//cho phép cửa sổ hiển thị.
    }
}
```

Kết quả thực thi chương trình:

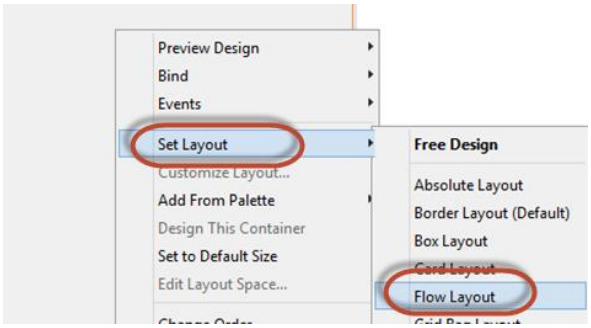


Hình 3.19: Kết quả minh họa cách trình bày của FlowLayout

Sử dụng Flow Layout trong NetBeans:

Sử dụng chế độ thiết kế (Design), chuột phải vào container (ví dụ JFrame) ➔ chọn Set Layout ➔ chọn Flow Layout

Sau khi chọn bố cục là Flow Layout, bộ quản lý bố cục sẽ bố trí các thành phần theo thứ tự từ trái qua phải.



Hình 3.20: Chọn kiểu FlowLayout

Ưu và nhược điểm của Flow Layout Manager:

Flow Layout sử dụng đơn giản, tuy nhiên nếu container bị thay đổi kích cỡ một số thành phần có thể chuyển lên trên hoặc chuyển xuống dưới tuỳ thuộc vào chiều ngang.

3.2.2.2. BoxLayout

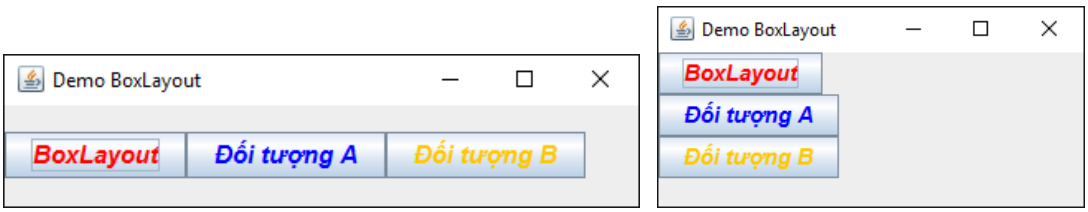
Box Layout tương tự như Flow Layout ngoại trừ bố trí các thành phần theo cột (từ trên xuống dưới hoặc từ dưới lên trên) hoặc theo dòng (từ trái qua phải hoặc từ phải qua trái).

Hình 3.21: Cách trình bày BoxLayout

```
import java.awt.*;
import javax.swing.*;
public class MyBoxLayout extends JFrame{
   public MyBoxLayout(String title){
      super(title);
   }
   public void doShow(){
      setSize(400,100);
      setLocationRelativeTo(null);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      addControl(); //gọi hàm AddControl
      setVisible(true);
   }
   public void addControl(){
      JPanel pnBox=new JPanel();
      pnBox.setLayout(new BoxLayout(pnBox, BoxLayout.X_AXIS));
      JButton btn1=new JButton("BoxLayout");
      btn1.setForeground(Color.RED);
      Font font=new Font("Arial",Font.BOLD | Font.ITALIC,15);
      btn1.setFont(font);
      pnBox.add(btn1);
      JButton btn2=new JButton("Đối tượng A");
      btn2.setForeground(Color.BLUE);
      btn2.setFont(font);
      pnBox.add(btn2);
      JButton btn3=new JButton("Đối tượng B");
      btn3.setForeground(Color.ORANGE);
      btn3.setFont(font);
      pnBox.add(btn3);
      Container con=getContentPane();
      con.add(pnBox);
   }
   public static void main(String[] args) {
```

```
            MyBoxLayout box=new MyBoxLayout("Demo BoxLayout");
            box.doShow();
        }
    }
```

Kết quả:

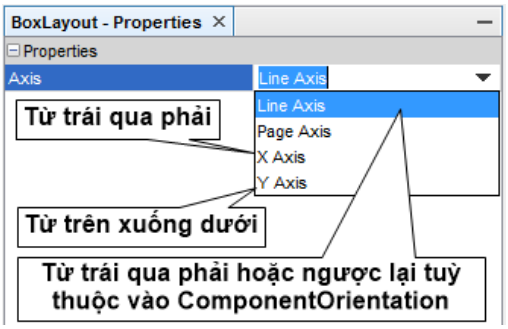


Hình 3.22: Kết quả cách trình bày BoxLayout

Trong đoạn lệnh pnBox.setLayout(new BoxLayout(pnBox, BoxLayout.X_AXIS)); nếu đổi BoxLayout.X_AXIS thành BoxLayout.Y_AXIS thì các control sẽ được hiển thị như sau:

Sử dụng Box Layout trong NetBeans:

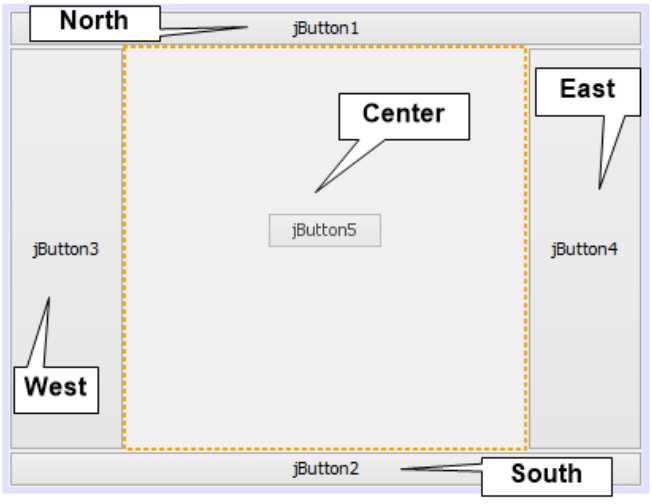
Click phải vào container -> chọn Set Layout -> chọn Box Layout. Lựa chọn cách bố trí



Hình 3.23: Chọn kiểu FlowLayout

3.2.2.3. BorderLayout

Border Layout bố trí các thành phần giao diện theo các biên với các vị trí trên (north), dưới (south), trái (west), phải (east) và ở giữa (center). Hình bên dưới là một ví dụ về Border Layout:



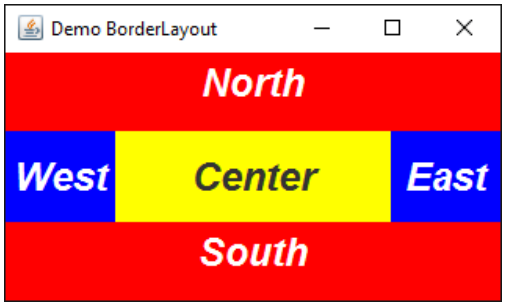
Hình 3.24: Kiểu BorderLayout

Nếu như không có 4 vùng : North, West, South, East. Thì vùng Center sẽ tràn đầy cửa sổ, thông thường khi đưa các control JTable, JTree, ListView, JScrollpane… ta thường đưa vào vùng Center để nó có thể tự co giãn theo kích thước cửa sổ giúp giao diện đẹp hơn

Chương trình sau minh họa về BorderLayout:

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Font;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
public class MyBorderLayout extends JFrame{
   public MyBorderLayout(String title){
      setTitle(title);
   }
   public void doShow(){
      setSize(400,300);
      setLocationRelativeTo(null);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      addControl();
      setVisible(true);
   }
   public void addControl(){
      JPanel pnBorder=new JPanel();
      pnBorder.setLayout(new BorderLayout());
      Font ft=new Font("Arial", Font.BOLD|Font.ITALIC, 25);
      JPanel pnNorth=new JPanel();
      pnNorth.setBackground(Color.RED);
      pnNorth.setPreferredSize(new Dimension(0, 50));
      JLabel lblTitleNorth=new JLabel("North");
      pnNorth.add(lblTitleNorth);
      lblTitleNorth.setForeground(Color.WHITE);
      lblTitleNorth.setFont(ft);
      pnBorder.add(pnNorth,BorderLayout.NORTH);
      JPanel pnSouth=new JPanel();
      pnSouth.setBackground(Color.RED);
      pnSouth.setPreferredSize(pnNorth.getPreferredSize());
      JLabel lblTitleSouth=new JLabel("South");
      pnSouth.add(lblTitleSouth);
      lblTitleSouth.setForeground(Color.WHITE);
```

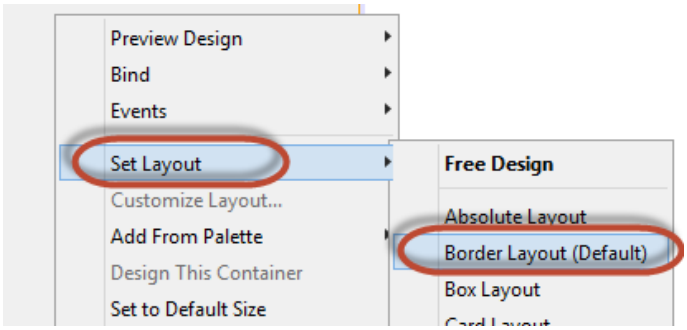
```
        lblTitleSouth.setFont(ft);
        pnBorder.add(pnSouth,BorderLayout.SOUTH);
        JPanel pnWest=new JPanel();
        pnWest.setBackground(Color.BLUE);
        pnWest.setPreferredSize(new Dimension(70, 0));
        JLabel lblTitleWest=new JLabel("West",JLabel.CENTER);
        lblTitleWest.setFont(ft);
        lblTitleWest.setForeground(Color.WHITE);
        pnWest.setLayout(new BorderLayout());
        pnWest.add(lblTitleWest,BorderLayout.CENTER);
        pnBorder.add(pnWest,BorderLayout.WEST);
        JPanel pnEast=new JPanel();
        pnEast.setBackground(Color.BLUE);
        pnEast.setPreferredSize(new Dimension(70, 0));
        JLabel lblTitleEast=new JLabel("East",JLabel.CENTER);
        lblTitleEast.setFont(ft);
        lblTitleEast.setForeground(Color.WHITE);
        pnEast.setLayout(new BorderLayout());
        pnEast.add(lblTitleEast,BorderLayout.CENTER);
        pnBorder.add(pnEast,BorderLayout.EAST);
        JPanel pnCenter=new JPanel();
        pnCenter.setBackground(Color.YELLOW);
        pnCenter.setLayout(new BorderLayout());
        JLabel lblTitleCenter=new JLabel("Center",JLabel.CENTER);
        lblTitleCenter.setFont(ft);
        pnCenter.add(lblTitleCenter,BorderLayout.CENTER);
        pnBorder.add(pnCenter,BorderLayout.CENTER);
        Container con=getContentPane();
        con.add(pnBorder);
    }
    public static void main(String[] args) {
        MyBorderLayout bor=new MyBorderLayout("Demo BorderLayout");
        bor.doShow();
    }
}
```



Hình 3.25: Kết quả minh họa BorderLayout

Sử dụng Border Layout trong NetBeans:

Tương tự như cách sử dụng Flow Layout, ngoại trừ chọn Border Layout



Hình 3.26: Chọn kiểu BorderLayout

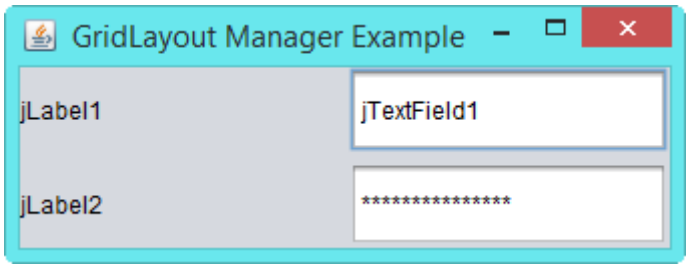
Ưu và nhược điểm của Border Layout Manager:

Border Layout cho phép chỉ định trực tiếp nơi mà một thành phần được đặt. Tuy nhiên khi sử dụng Borde rLayout, chúng ta chỉ có tối đa 5 vị trí trong một container.

Chúng ta có thể kết hợp FlowLayout , BoxLayout, BorderLayout để thiết kế giao diện, theo kinh nghiệm của Tôi thì chúng ta chỉ cần biết 3 Layout này là có thể thiết kế giao diện đẹp mắt được.

3.2.2.4. GridLayout

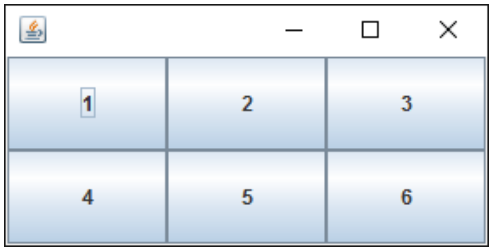
Grid Layout bố trí các thành phần theo dòng và cột. Mỗi ô (nơi giao nhau của dòng và cột) trong Grid Layout có cùng kích cỡ.



Hình 3.27: Cách bố trí GridLayout

```
import java.awt.*;
import javax.swing.*;
public class MyGridLayout{
   JFrame f;
   MyGridLayout(){
      f=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");
      f.add(b1);f.add(b2);f.add(b3);      f.add(b4);f.add(b5);f.add(b6);
      f.setLayout(new GridLayout(2,3));
      //thiet lap 2 hang va 3 cot cho grid layout
      f.setSize(300,150);
```

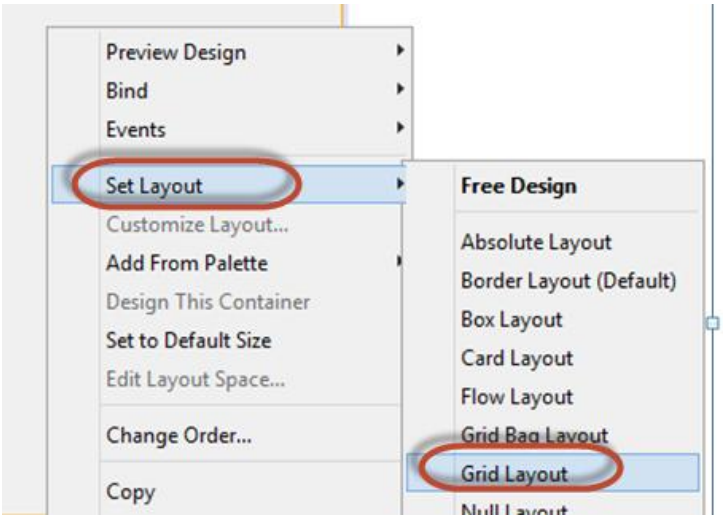
```
        f.setVisible(true);
    }
    public static void main(String[] args) {
     new MyGridLayout();
    }
}
```



Hình 3.28: Kết quả minh họa GridLayout

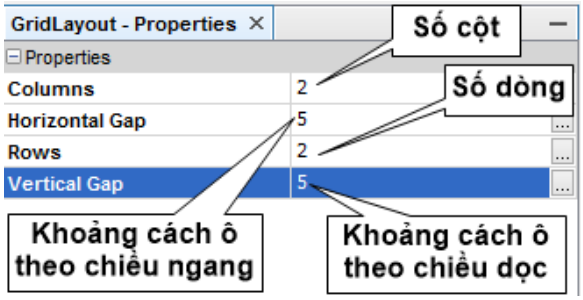
Sử dụng Grid Layout trong NetBeans

Chuột phải vào container -> chọn **Set Layout** -> chọn **Grid Layout**



Hình 3.29: Chọn kiểu GridLayout

Chỉ định số dòng và số cột



Hình 3.30: Chọn số dòng, số cột cho GridLayout

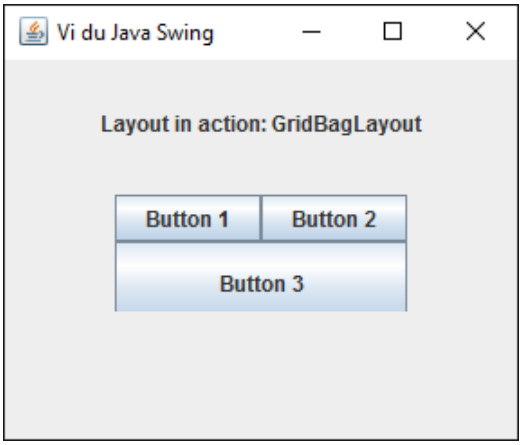
3.2.2.5. GridBagLayout

Bố trí component trải trên nhiều hàng và cột với kích thước khác nhau Thiết lập 1 mạng lưới ô liên kết với container và bố trí các component trên một hoặc nhiều ô.

import java.awt.;*

```
import java.awt.event.*;
import javax.swing.*;
public class GridBagLayoutDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  private JLabel msglabel;
  public GridBagLayoutDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    GridBagLayoutDemo swingLayoutDemo = new GridBagLayoutDemo();
    swingLayoutDemo.showGridBagLayoutDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du Java Swing");
    mainFrame.setSize(300,250);
    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 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;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.gridwidth = 2;
        panel.add(new JButton("Button 3"),gbc);
        controlPanel.add(panel);
        mainFrame.setVisible(true);
    }
}
```



Hình 3.30: Kết quả minh họa GridBagLayout

3.3. CÁC THÀNH PHẦN GUI CƠ BẢN

- JLabel:Hiển thị văn bản hay những biểu tượng.
- JTextField: Trường nhập dữ liệu từ bàn phím, cũng có thể hiển thị thông tin.
- JPasswordField: Trường nhập dữ liệu password.
- JTextArea: Vùng văn bản cho phép thao tác soạn thảo nhiều dòng văn bản.
- JButton: Nút nhấn dùng kích hoạt một sự kiện khi nhấp chuột.

3.3.1. JLabel

JLabel là một thành phần cho phép trình bày các dạng thông tin sau:

- Dòng văn bản chỉ đọc
- Hình ảnh
- Văn bản và hình ảnh

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

public class Jlabel extends JComponent implements SwingConstants, Accessible

Constructor của lớp JLabel trong Java Swing:

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

Các phương thức quan trọng của JLabel là:

Tên phương thức	Mô tả
setText(String label)	Gán nội dung cho JLabel
setToolTipText (String label)	Nội dung chú thích khi rê chột vào JLabel
setFont(Font f)	Thay đổi phông chữ của Label
getText()	Lấy nội dung của Jlabe
setDisabledIcon(Icon disabledIcon)	Thiết lập icon để được hiển thị nếu JLabel này là "disabled" (JLabel.setEnabled(false))
setDisplayedMnemonic(char aChar)	Xác định displayedMnemonic như là một giá trị char
setDisplayedMnemonic(int key)	Xác định keycode mà chỉ dẫn một mnemonic key
setDisplayedMnemonicIndex(int index)	Cung cấp một hint cho L&F, từ đó ký tự trong text nên được trang trí để biểu diễn mnemonic
setHorizontalAlignment(int alignment)	Thiết lập sự căn chỉnh nội dung của label theo trục X
setHorizontalTextPosition(int textPosition)	Thiết lập vị trí theo chiều ngang của phần text của label, cân xứng với hình ảnh của nó
setIcon(Icon icon)	Định nghĩa icon mà thành phần này sẽ hiển thị
setIconTextGap(int iconTextGap)	Nếu cả hai thuộc tính icon và text được thiết lập, thuộc tính này xác định khoảng

	trống giữa chúng
setLabelFor(Component c)	Thiết lập thành phần đang gán nhãn cho
setUI(LabelUI ui)	Thiết lập đối tượng L&F mà biểu diễn thành phần này
setVerticalAlignment(int alignment)	Thiết lập sự căn chỉnh nội dung của label theo trục Y
setVerticalTextPosition(int textPosition)	Thiết lập vị trí theo chiều dọc của phần text của label, cân xứng với hình ảnh của nó
updateUI()	Phục hồi thuộc tính UI về một giá trị từ L&F hiện tại

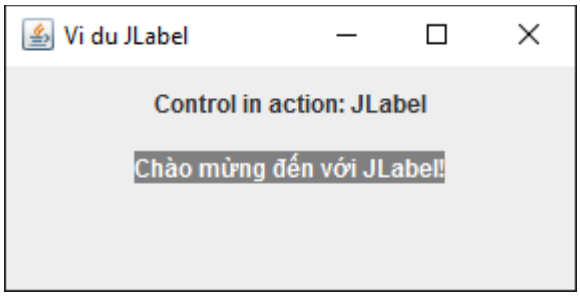
Bảng 3.5: Các phương thức của JLabel

Chương trình minh họa lớp JLabel:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JlabelDemo {
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  public JlabelDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    JlabelDemo  swingControlDemo = new JlabelDemo();
    swingControlDemo.showLabelDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du JLabel");
    mainFrame.setSize(300,150);
    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(150,100);
    controlPanel = new JPanel();
    controlPanel.setLayout(new FlowLayout());
```

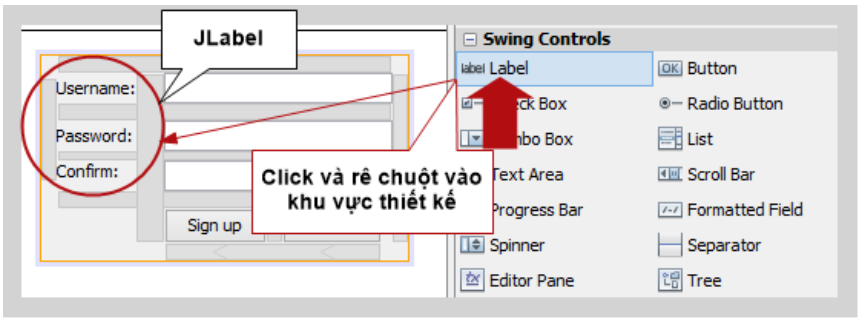
```
        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
    private void showLabelDemo(){
        headerLabel.setText("Control in action: JLabel");
        JLabel label  = new JLabel("", JLabel.CENTER);
        label.setText("Chào mừng đến với JLabel!");
        label.setOpaque(true);
        label.setBackground(Color.GRAY);
        label.setForeground(Color.WHITE);
        controlPanel.add(label);
        mainFrame.setVisible(true);
    }
}
```

Kết quả:



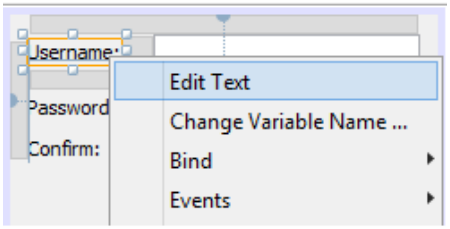
Hình 3.31: Kết quả minh họa JLabel

Tạo JLabel trong NetBeans:



Hình 3.32: Minh họa thiết kế JLabel

Thay đổi nội dung hiển thị: chuột phải vào **JLabel** -> chọn **Edit Text**



Hình 3.33: Minh họa điều chỉnh nhãn hiện thị

3.3.2. JTextField

JTextField cho phép người dùng nhập và chỉnh sửa một dòng văn bản.

Cú pháp khai báo của lớp javax.swing.JTextField:

public class JTextField extends JTextComponent implements SwingConstants

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 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.

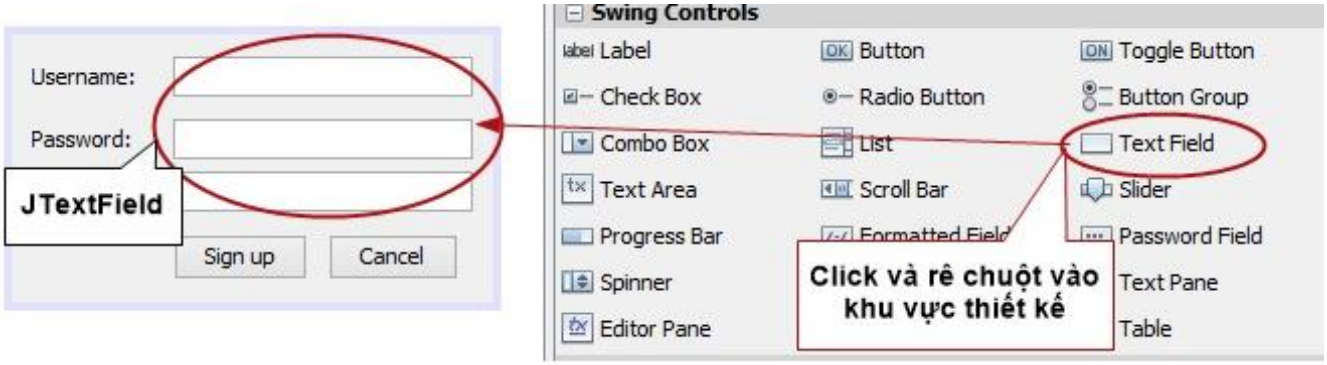
Các phương thức quan trọng của JtextField là:

Tên phương thức	Mô tả
getText();	Lấy nội dung của JtextField
setText(String value);	Gán nội dung của JtextField
setEditable(boolean editable)	Thiết lập cho chỉnh sửa nội dung: Nếu editable =true, được phép chỉnh. Ngược lại thì không.
setActionCommand(String command)	Thiết lập chuỗi lệnh được sử dụng cho action event
setColumns(int columns)	Thiết lập số cột trong TextField này, và sau đó làm mất hiệu lựa layout đó
setDocument(Document doc)	Liên kết editor với một tài liệu text
setFont(Font f)	Thiết lập font
setHorizontalAlignment(int alignment)	Thiết lập căn chỉnh ngang cho text
setScrollOffset(int scrollOffset)	Thiết lập scroll offset, giá trị pixel
void actionPropertyChanged(Action action, String propertyName)	Cập nhật trạng thái của textfield trong phản hồi các thay đổi của thuộc tính trong action.
addActionListener(ActionListener l)	Thêm action listener đã cho để nhận các action event từ textfield này
configurePropertiesFromAction(Action a)	Thiết lập các thuộc tính của textfield này để kết nối chúng trong Action đã cho
createActionPropertyChangeListener(Action 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
createDefaultModel()	Tạo trình triển khai mặc định của model

	để được sử dụng tại sự xây dựng nếu không được cung cấp tường minh
getActions()	Gọi danh sách lệnh cho trình soạn thảo Editor
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
removeActionListener(ActionListener l)	Xóa action listener đã cho để nó không bao giờ nhận action event từ textfield này nữa
scrollRectToVisible(Rectangle r)	Cuốn trường này sang trái hoặc phải
setAction(Action a)	Thiết lập Action cho ActionEvent source

Bảng 3.6: Các phương thức của Jtextfield

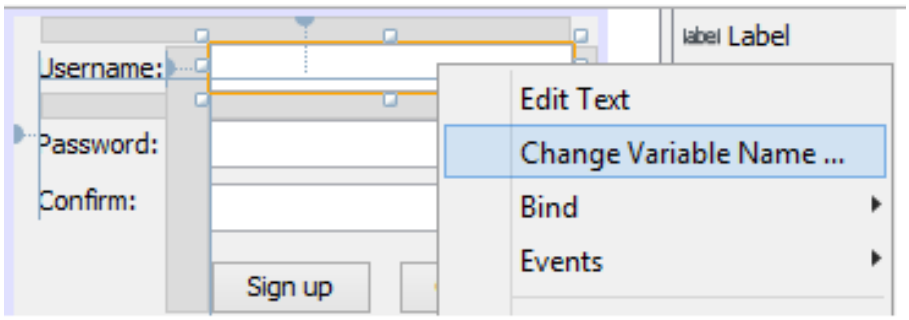
Tạo JTextField trong NetBeans:



Hình 3.34: Minh họa thiết kế JTextField

Thay đổi nội dung của JTextField: chuột phải và **JTextField** -> chọn **Edit Text:**

Thay đổi tên đối tượng (Tên được sử dụng để truy cập các phương thức và thuộc tính của JTextField): Chuột phải vào **JTextField** -> chọn **Change Variable Name:**



Hình 3.35: Minh họa đặt tên TextField

3.3.3. JPasswordField

JPasswordField tương tự như JTextField ngoại trừ nội dung trong JPasswordField sẽ chuyển thành dấu hoa thị (*).

Các phương thức quan trọng của JpasswordField giống như JTextField là:

Tên phương thức	Mô tả
String text = getText();	Lấy nội dung của JPasswordField
setText(String value);	Thiết lập nội dung của JPasswordField
setEchoChar()	Thiết lập ký tự hiển thị

Bảng 3.7: Các phương thức của JPasswordFiel

Chương trình minh họa lớp JTextField và JPasswordField:

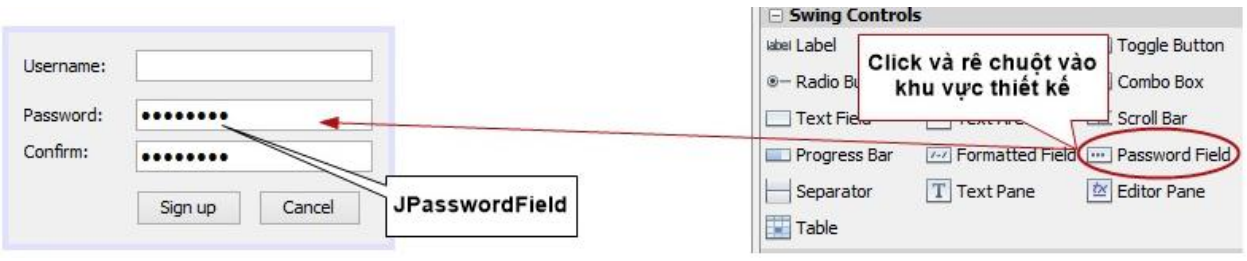
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JTextFieldDemo{
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  public JTextFieldDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    JTextFieldDemo  swingControlDemo = new JTextFieldDemo();
    swingControlDemo.showTextFieldDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du JTextField");
    mainFrame.setSize(400,150);
    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);
      }
    }
```

Kết quả:

Hình 3.36: Kết quả minh họa JtextField và JPasswordField

Tạo JPasswordField trong NetBeans:



Hình 3.37: Minh họa thiết kế JPasswordField

3.3.4. JTextArea

JTextArea cho phép nhập và chỉnh sửa nhiều dòng văn bản

Cú pháp khai báo của lớp javax.swing.JTextArea:

public class JTextArea extends JTextComponent

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

Các phương thức quan trọng của JTextArea:

Tên phương thức	Mô tả
String text = getText();	Lấy nội dung của JTextArea
setText(String value);	Thiết lập nội dung của JTextArea
setEditable(boolean editable)	Thiết lập cho phép chỉnh sửa nội dung: Nếu editable =true, được phép chỉnh. Ngược lại thì không.
copy()	Sao chép văn bản được chọn vào clipboard
cut()	Di chuyển văn bản được chọn vào clipboard
paste()	Chuyển nội dung từ clipboard vào JtextArea
setRows(int rows)	Sử dụng để thiết lập số hàng đã cho.
setColumns(int cols)	Sử dụng để thiết lập số cột đã cho.
setFont(Font f)	Sử dụng để thiết lập font đã cho.
insert(String s, int position)	Sử dụng để chèn text vào vị trí đã cho.

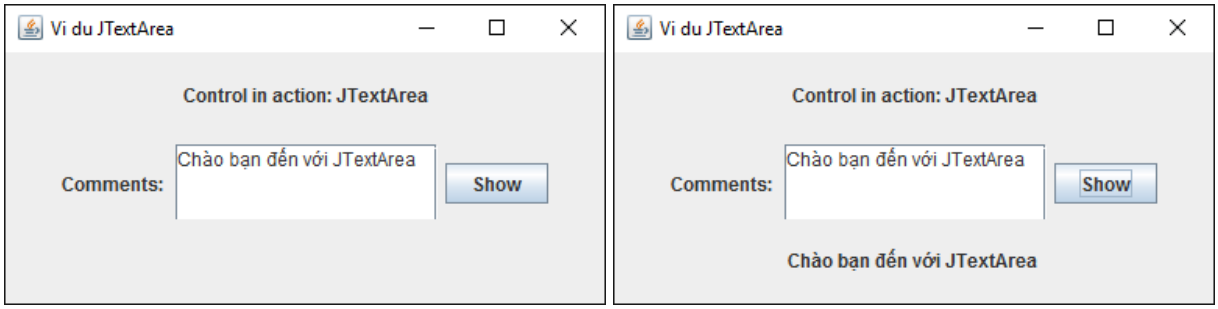
Bảng 3.8: Các phương thức của JTexArea

Chương trình minh họa lớp JTextArea:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JTextAreaDemo{
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  public JTextAreaDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    JTextAreaDemo  swingControlDemo = new JTextAreaDemo();
    swingControlDemo.showTextAreaDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du JTextArea");
    mainFrame.setSize(400,200);
```

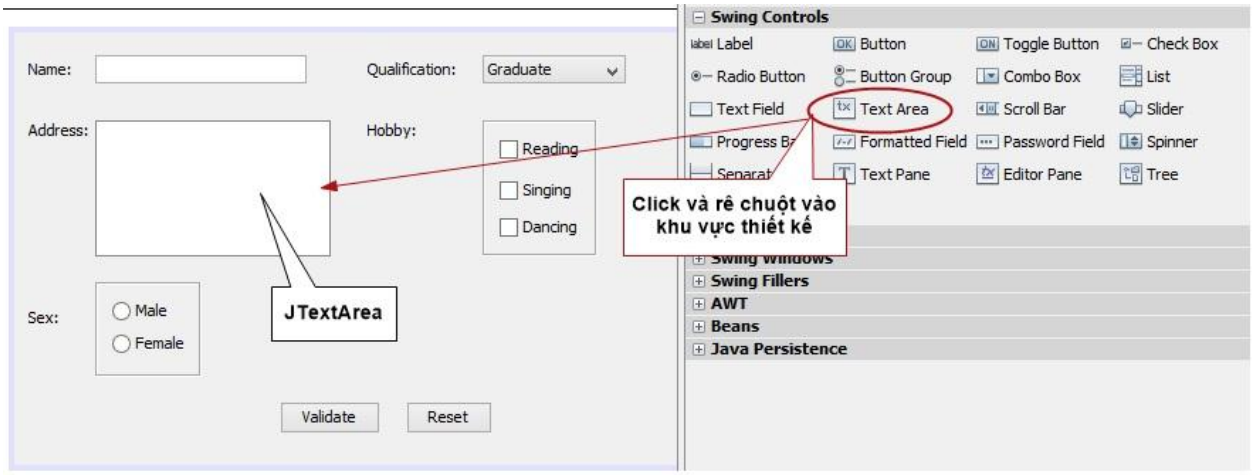
```
    mainFrame.setLayout(new GridLayout(3, 1));
    mainFrame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent windowEvent){
        System.exit(0);
      }
    });
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("",JLabel.CENTER);
    statusLabel.setSize(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("Chào bạn đến với JTextArea ",3,15);
    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);
  }
}
```

Kết quả:



Hình 3.38: Kết quả minh họa JTextArea

Tạo JTextArea trong NetBeans:



Hình 3.37: Minh họa thiết kế JTextField

3.3.5. JButton

JButton là một thành phần hình chữ nhật với một văn bản hoặc biểu tượng hoặc cả hai làm nhãn và có thể phát sinh sự kiện khi người dùng nhấn chuột.

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

public class JButton extends AbstractButton implements Accessible

Lớp JButton có các constructor sau:

- JButton(): Tạo button mà không thiết lập text hoặc icon.
- JButton(Action a): Tạo button các thuộc tính được nhận từ Action đã cung cấp.
- JButton(Icon icon): Tạo button với một icon.
- JButton(String text): Tạo button với text.
- JButton(String text, Icon icon): Tạo button với text ban đầu và một icon.

- Các phương thức quan trọng của JTextArea:

Tên phương thức	Mô tả
setText(String s)	Sử dụng để thiết lập text đã cho trên button.
getText()	Sử dụng để trả về text của button.
setIcon(Icon b)	Sử dụng để thiết lập Icon đã cho trên button.
getIcon()	Sử dụng để lấy Icon của button
setEnabled(boolean b)	Sử dụng để kích hoạt hoặc vô hiệu hóa button.
setMnemonic(int a)	Sử dụng để thiết lập thuộc tính mnemonic trên button.
addActionListener(ActionListener a)	Sử dụng để thêm action listener tới đối tượng này.
updateUI()	Phục hồi thuộc tính UI về một giá trị từ L&F hiện tại
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
removeNotify()	Ghi đè JComponent.removeNotify để kiểm tra xem button này được thiết lập như là button mặc định trên RootPane hay không, 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

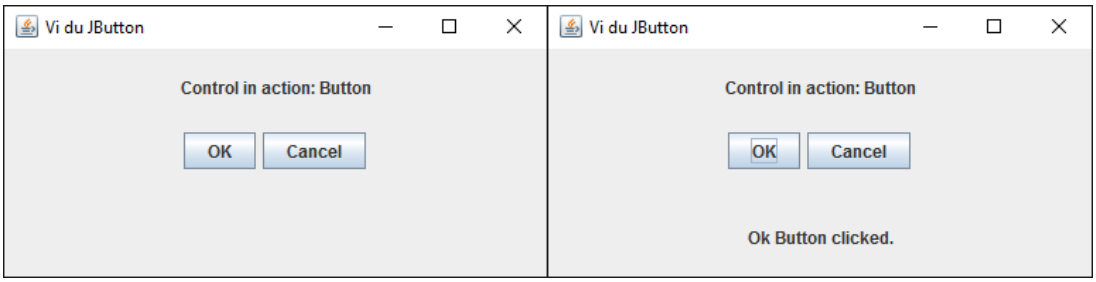
Bảng 3.9: Các phương thức của JButton

Chương trình minh họa lớp JButton:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JButtonDemo{
  private JFrame mainFrame;
  private JLabel headerLabel;
  private JLabel statusLabel;
  private JPanel controlPanel;
  public JButtonDemo(){
    prepareGUI();
  }
  public static void main(String[] args){
    JButtonDemo  swingControlDemo = new JButtonDemo();
    swingControlDemo.showButtonDemo();
  }
  private void prepareGUI(){
    mainFrame = new JFrame("Vi du JButton");
    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);
```

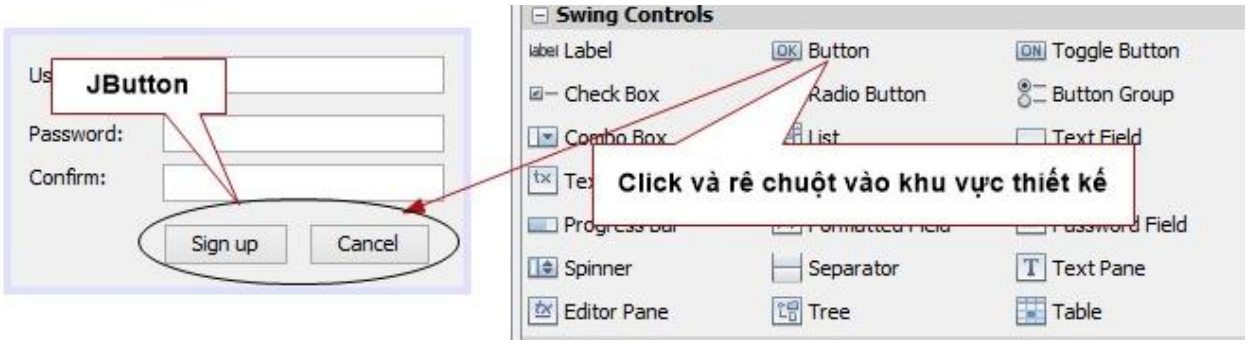
```
    }
    private void showButtonDemo(){
      headerLabel.setText("Control in action: Button");
      JButton okButton = new JButton("OK");
      JButton cancelButton = new JButton("Cancel");
      cancelButton.setHorizontalTextPosition(SwingConstants.LEFT);
      okButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          statusLabel.setText("Ok Button clicked.");
        }
      });
      cancelButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
          statusLabel.setText("Cancel Button clicked.");
        }
      });
      controlPanel.add(okButton);
      controlPanel.add(cancelButton);
      mainFrame.setVisible(true);
    }
  }
```

Kết quả:



Hình 3.38: Kết quả minh họa JButton

Tạo JButton trong NetBeans:



Hình 3.39: Minh họa thiết kế Jbutton

3.4. XỬ LÝ SỰ KIỆN

3.4.1. Các lớp Event

Sự kiện 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 chuật, 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 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.

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:

public class EventObject extends Object implements Serializable

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.

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.

- 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:

  public class AWTEvent extends EventObject

- 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:

  public class ActionEvent extends AWTEvent

- 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:

  public abstract class InputEvent extends ComponentEvent

- 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_RELASED
  KEY_TYPED

  Cú pháp khai báo KeyEvent như sau:

  public class KeyEvent extends InputEvent

- 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:

  public class MouseEvent extends InputEvent

- 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:

  public class WindowEvent extends ComponentEvent
- 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:

  public class AdjustmentEvent extends AWTEvent
- 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:

  public class ComponentEvent extends AWTEvent
- 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:

  public class ContainerEvent extends ComponentEvent
- Lớp MouseMotionEvent: 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:

  public class MouseMotionEvent extends InputEvent
- 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:

  public class PaintEvent extends ComponentEvent

3.4.2. Các lớp Listener

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. Cú pháp để khai báo:

public interface EventListener

Các Event Listener Interface trong Java Swing:

- ActionListener Interface: Sử dụng để nhận action event
- ComponentListener Interface: Sử dụng để nhận component event

- ItemListener Interface: Sử dụng để nhận item event
- KeyListener Interface: Sử dụng để nhận key event
- MouseListener Interface: Sử dụng để nhận mouse event
- WindowListener Interface: Sử dụng để nhận window event
- AdjustmentListener Interface: Sử dụng để nhận adjusmtent event
- ContainerListener Interface: Sử dụng để nhận container event
- MouseMotionListener Interface: Sử dụng để nhận mouse motion event
- FocusListener Interface: Sử dụng để nhận focus event

3.4.3. Các lớp Adapter

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. Một số adapter được sử dụng phổ biến khi nghe các GUI event trong Java Swing:

- FocusAdapter: Một lớp abstract để nhận các focus event.
- KeyAdapter: Một lớp abstract để nhận các key event
- MouseAdapter: Một lớp abstract để nhận các mouse event
- MouseMotionAdapter: Một lớp abstract để nhận các mouse motion event
- WindowAdapter: Một lớp abstract để nhận các window event

3.4.4. Xử lý sự kiện chuột

Sự kiện chính là sự phản ứng của chương trình khi có sự tương tác của người dùng.

Ví dụ khi người dùng nhấn chuột vào nút nhấn (JButton), … Và khi một sự kiện được phát sinh thì chương trình phải thực hiện xử lý sự kiện đó. Bảng liệt kê những sự kiện thường gặp.

Components	Listeners	Methods
Button, Menu, List	ActionListener	void actionPerformed(ActionEvent ae)
Scrollbar	AdjustmentListener	void adjustmentValueChanged(AdjustmentEvent ae)
Check box, List	ItemListener	void itemStateChanged(ItemEvent ie)
Mouse	MouseListener MouseMotionListener	void mouseClicked(MouseEvent me) void mouseEntered(MouseEvent me) void mouseExited(MouseEvent me) void mousePressed(MouseEvent me) void mouseReleased(MouseEvent me) void mouseDragged(MouseEvent me) void mouseMoved(MouseEvent me)

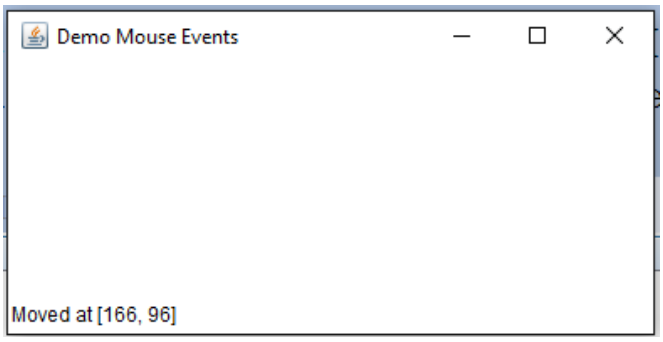
Bảng 3.10: Các sự kiện thường dùng

Ví dụ: Chương trình MouseTracker dùng những phương thức của các interfaces MouseListener và MouseMotionListener để "bẫy" và xử lý các sự kiện chuột tương ứng.

```
import java.awt.*;
import java.awt.event.*;
public class MouseTracker extends Frame implements
                                        MouseListener, MouseMotionListener {
   private Label statusBar;
   // set up GUI and register mouse event handlers
   public MouseTracker() {
      super("Demo Mouse Events");
      statusBar = new Label();
      this.add(statusBar, BorderLayout.SOUTH);
      // application listens to its own mouse events
      addMouseListener(this);
      addMouseMotionListener(this);
      setSize(400, 200);
      setVisible(true);
   }
   // MouseListener event handlers handle event
   //when mouse released immediately after press
   public void mouseClicked(MouseEvent event) {
      statusBar.setText("Clicked at [" + event.getX() + ", " + event.getY() + "]");
   }
   // handle event when mouse pressed
   public void mousePressed(MouseEvent event) {
      statusBar.setText("Pressed at [" + event.getX() + ", " + event.getY() + "]");
   }
   // handle event when mouse released after dragging
   public void mouseReleased(MouseEvent event) {
      statusBar.setText("Released at [" + event.getX() + ", " + event.getY()+
"]");
   }
   // handle event when mouse enters area
   public void mouseEntered(MouseEvent event) {
      statusBar.setText("Mouse in window");
   }
   // handle event when mouse exits area
   public void mouseExited(MouseEvent event) {
      statusBar.setText("Mouse outside window");
   }
   // MouseMotionListener event handlers
   // handle event when user drags mouse with button pressed
   public void mouseDragged(MouseEvent event) {
      statusBar.setText("Dragged at [" + event.getX() + ", " + event.getY() +
"]");
```

```
    }
    // handle event when user moves mouse
    public void mouseMoved(MouseEvent event) {
       statusBar.setText("Moved at [" + event.getX() + ", " + event.getY()+ "]");
    }
    // execute application
    public static void main(String args[]) {
       MouseTracker application = new MouseTracker();
    }
  } // end class MouseTracker
```

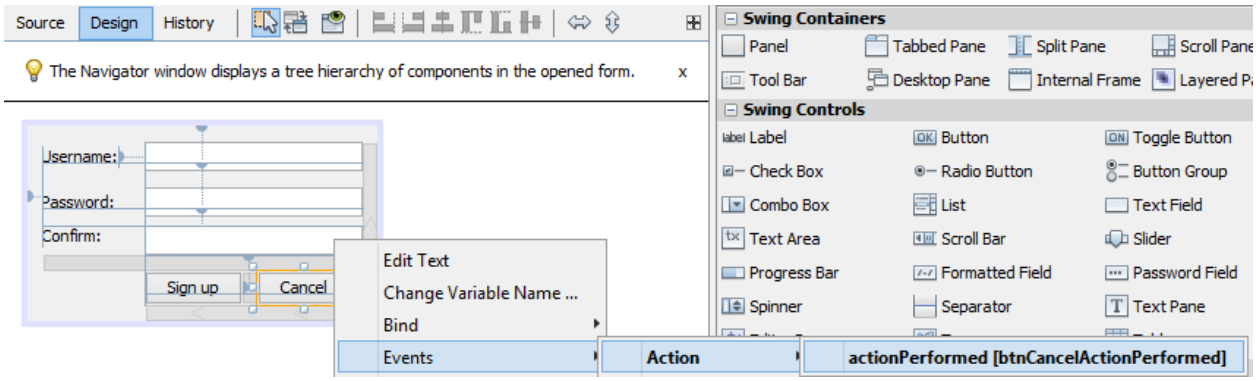
Kết quả:



Hình 3.40: Kết quả minh họa Demo Mouse Events

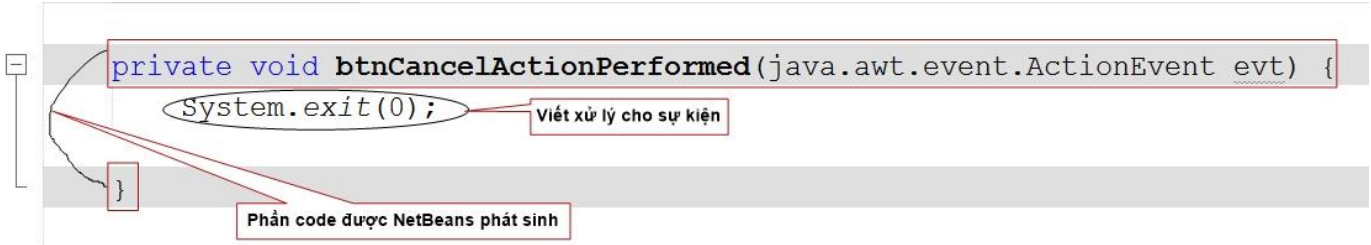
Đăng ký và xử lý sự kiện trong NetBeans:

Chọn chế độ **Design** -> chuột phải lên thành phần muốn xử lý sự kiện ➔ chọn **Events** ➔ lựa chọn loại sự kiện muốn xử lý. Bên dưới là ví dụ về đăng ký sự kiện cho nút nhấn **Cancel** (Sự kiện click).



Hình 3.41: Màn hình đăng ký sự kiện

Và kết quả nhận được:



Hình 3.42: Màn hình Code cho sự kiện

3.4.5. Xử lý sự kiện bàn phím

Java cung cấp giao diện KeyListener cho phép xử lý những sự kiện phím được sinh ra khi những phím trên bàn phím được nhấn và thả.

Một lớp hiện thực KeyListener phải cài đặt các phương thức keyPressed, keyReleased và keyTyped. Mỗi phương thức này có một tham số là một đối tượng kiểu KeyEvent.

KeyEvent là lớp con của lớp InputEvent.

Các phương thức của interface KeyListener:

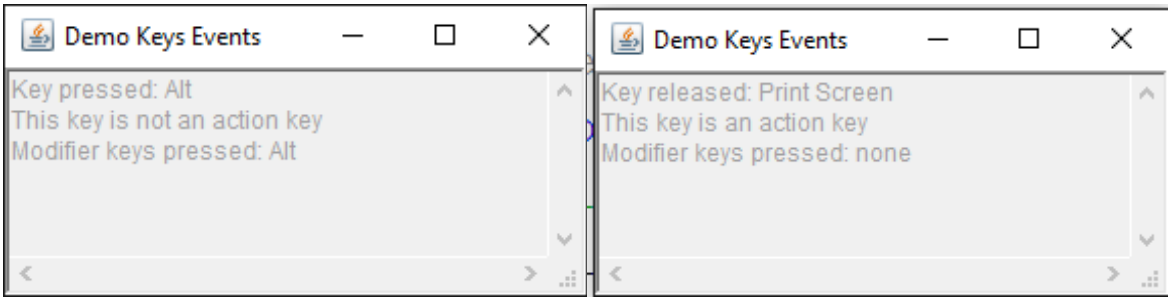
- Phương thức keyPressed được gọi khi một phím bất kỳ được nhấn.
- Phương thức keyTyped được gọi thực hiện khi người dùng nhấn một phím không phải "phím hành động" (như phím mũi tên, phím Home, End, Page Up, Page Down, các phím chức năng như: Num Lock, Print Screen, Scroll Lock, Caps Lock, Pause).
- Phương thức keyReleased được gọi thực hiện khi nhả phím nhấn sau khi sự kiện keyPressed hoặc keyTyped.

Ví dụ: Minh họa việc xử lý sự kiện chuột thông qua các phương thức của interface KeyListener. Lớp KeyDemo hiện thực interface KeyListener, có 3 phương thức trong KeyListener phải được cài đặt trong chương trình.

```
import java.awt.*;
import java.awt.event.*;
public class KeyDemo extends Frame implements KeyListener {
   private String line1 = "", line2 = "";
   private String line3 = "";
   private TextArea textArea;
   public KeyDemo() {
      super("Demo Keys Events");       // set up TextArea
      textArea = new TextArea(10, 15);
      textArea.setText("Press any key on the keyboard...");
      textArea.setEnabled(false);
      this.add(textArea);        // allow frame to process Key events
      addKeyListener(this);
      setSize(300, 150);
      setVisible(true);
   }    // handle press of any key
   public void keyPressed(KeyEvent event) {
      line1 = "Key pressed: " + event.getKeyText(event.getKeyCode());
      setLines2and3(event);
   }    // handle release of any key
   public void keyReleased(KeyEvent event) {
      line1 = "Key released: " + event.getKeyText(event.getKeyCode());
```

```
        setLines2and3(event);
    }    // handle press of an action key
    public void keyTyped(KeyEvent event) {
        line1 = "Key typed: " + event.getKeyChar();
        setLines2and3(event);
    }    // set second and third lines of output
    private void setLines2and3(KeyEvent event) {
        line2 = "This key is " + (event.isActionKey() ? "" : "not ") + "an action
key";
        String temp = event.getKeyModifiersText(event.getModifiers());
        line3 = "Modifier keys pressed: " + (temp.equals("") ? "none" : temp);
        textArea.setText(line1 + "\n" + line2 + "\n" + line3 + "\n");
    }   // execute application
    public static void main(String args[]) {
        KeyDemo application = new KeyDemo();
    }
} // end class KeyDemo
```

Kết quả:

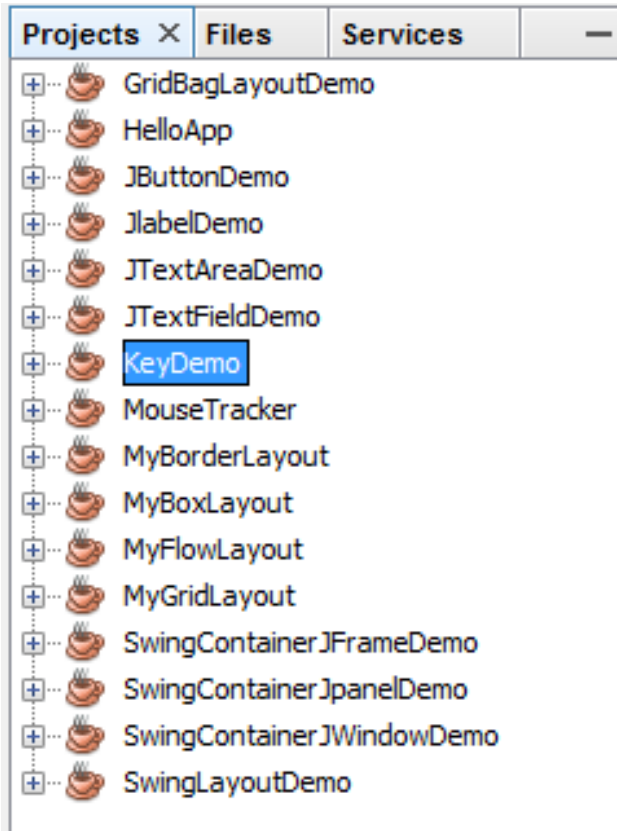


Hình 3.43: Màn hình kết quả KeyDemo

TÓM TẮT

Trong bài học này người học làm quen với các kiến thức sau:

- Cấu trúc và kỹ năng lập trình tổ chức một giao diện người dùng đồ họa trong Ngôn ngữ lập trình Java với các lớp thuộc 2 gói awt và swing.
- Thiết kế các thành phần là vật chưa các thành phần khác: JFrame, JWindow, JPanel
- Thiết kế và quản lý bố cục giáo diện với các lớp Layout: BoxLayout, BorderLayout, FlowLayout, GridLayout, GridBagLayout, …
- Lập trình tùy biến các đối tượng GUI trong thư viện Swing: Jlabel, JtextField, JPasswordField, JTextArea, JButton.
- Hiểu cơ chế xử lý sự kiện tương tác người dùng và nắm vững kỹ năng lập trình phát triển các đối tượng xử lý sự kiện với các lớp: Event, Listener, Adapter.
- Kỹ năng tự tìm hiểu và sử dụng các GUI component có sẵn trong thư viện.
- Thiết kế giao diện người dùng trên công cụ trực quan NetBeans IDE.
- Bài học đã thực hiện minh họa 16 ứng dụng Demo



---------- **Hết Bài 3** ----------