

**TRƯỜNG ĐẠI HỌC NAM CÀN THƠ
KHOA CÔNG NGHỆ THÔNG TIN**

Trung tâm đào tạo chuẩn đầu ra & Phát triển nguồn nhân lực



MODULE 02
LẬP TRÌNH GIAO DIỆN ĐỒ HỌA

Bùi Thị Diễm Trinh
Khoa CNTT, Đại Học Nam Cần Thơ
Email: btdtrinh@nctu.edu.vn

NỘI DUNG CHÍNH

- 1. Giới thiệu**
- 2. Các loại vật chứa (Container)**
- 3. Các thành phần giao diện Swing**
- 4. Bộ cục giao diện**
- 5. Xử lý sự kiện**
- 6. Trình đơn, thanh công cụ**
- 7. Thiết kế giao diện đồ họa với NetBeans**



Thư viện lập trình đồ họa

❑ Java cung cấp 2 bộ thư viện hàm dùng cho việc xây dựng giao diện đồ họa là: **AWT** và **SWING**.

❑ **Abstract Window Toolkit (AWT)**

- 2 gói thường dùng là `java.awt` (*điều khiển*) và `java.awt.event` (*sự kiện*).
- Cung cấp giao diện phụ thuộc vào nền GUI của hệ điều hành (*thay đổi theo hệ điều hành*).
- Các thành phần được gọi là **heavyweight components**.

❑ **Swing**

- Bản nâng cấp của AWT.
- Là 1 phần trong JFC (Java Foundation Classes)
- Giao diện độc lập với nền GUI của hệ điều hành (*không thay đổi theo hệ điều hành, viết hoàn toàn bằng Java*).
- Các thành phần được gọi là **lightweight components**.



Gói AWT (Abstract Window Toolkit)

❑ Gói `java.awt` bao gồm các lớp

- Thành phần GUI (*Button, TextField, and Label, ...*)
- Vật chứa GUI (*Frame, Panel, Dialog, ScrollPane, ...*)
- Sắp xếp bố cục (*FlowLayout, BorderLayout, GridLayout, ...*)
- Tùy chọn (*Graphics, Color, Font, ...*)

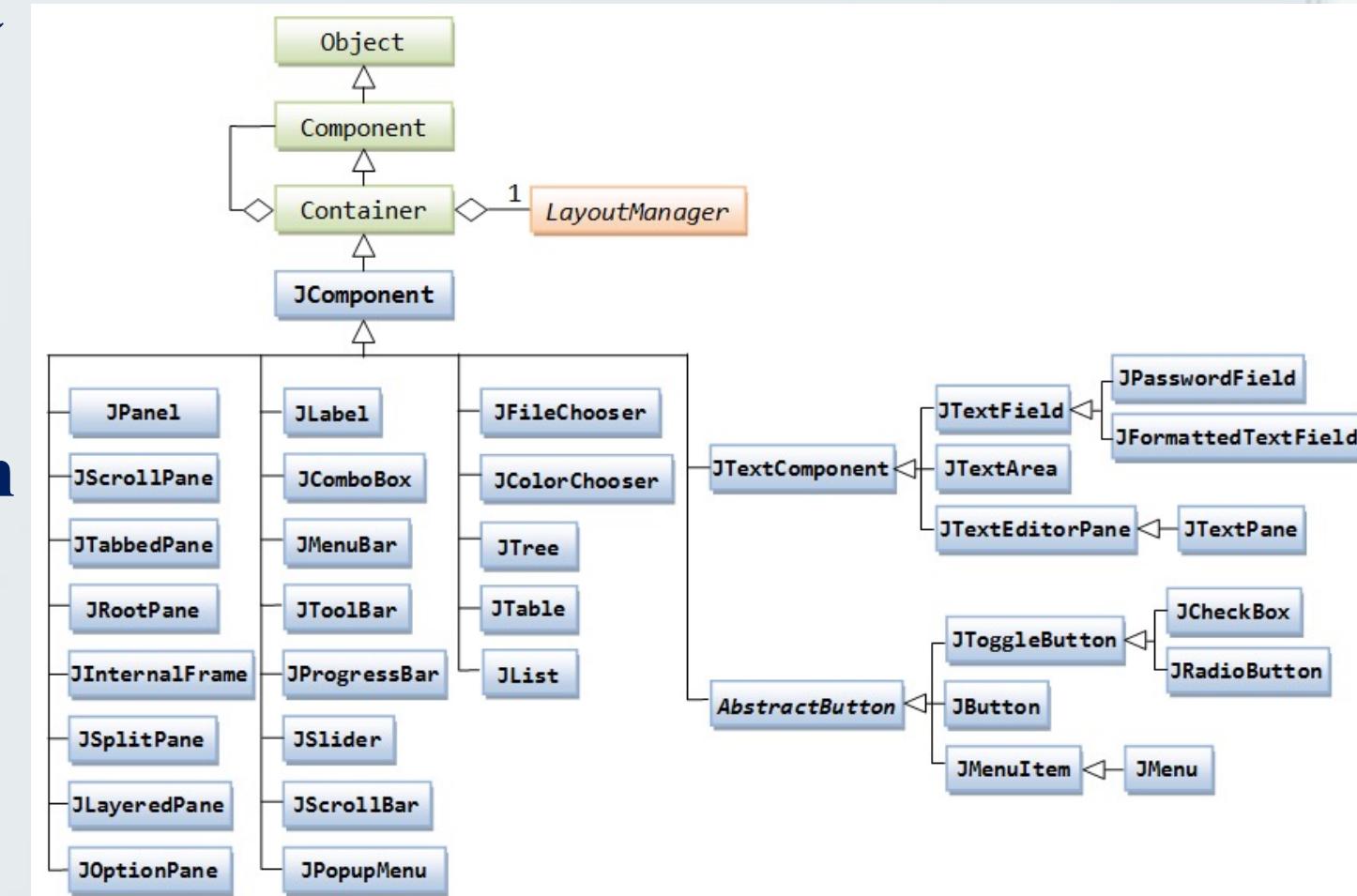
❑ Gói `java.awt.event` bao gồm các lớp

- Sự kiện (*ActionEvent, MouseEvent, KeyEvent, WindowEvent*)
- Lắng nghe sự kiện (*ActionListener, MouseListener, KeyListener, WindowListener, ...*)
- Các lớp Adapter (*MouseAdapter, KeyAdapter, and WindowAdapter*)



Gói Swing

- ❑ Là thành phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng window-based.
- ❑ JFC là một bộ các thành phần GUI cho phép đơn giản hóa phát triển các ứng dụng desktop.
- ❑ Các thành phần của gói Swing



Các thành phần của một giao diện đồ họa

❑ Thành phần giao diện người dùng (User Interface)

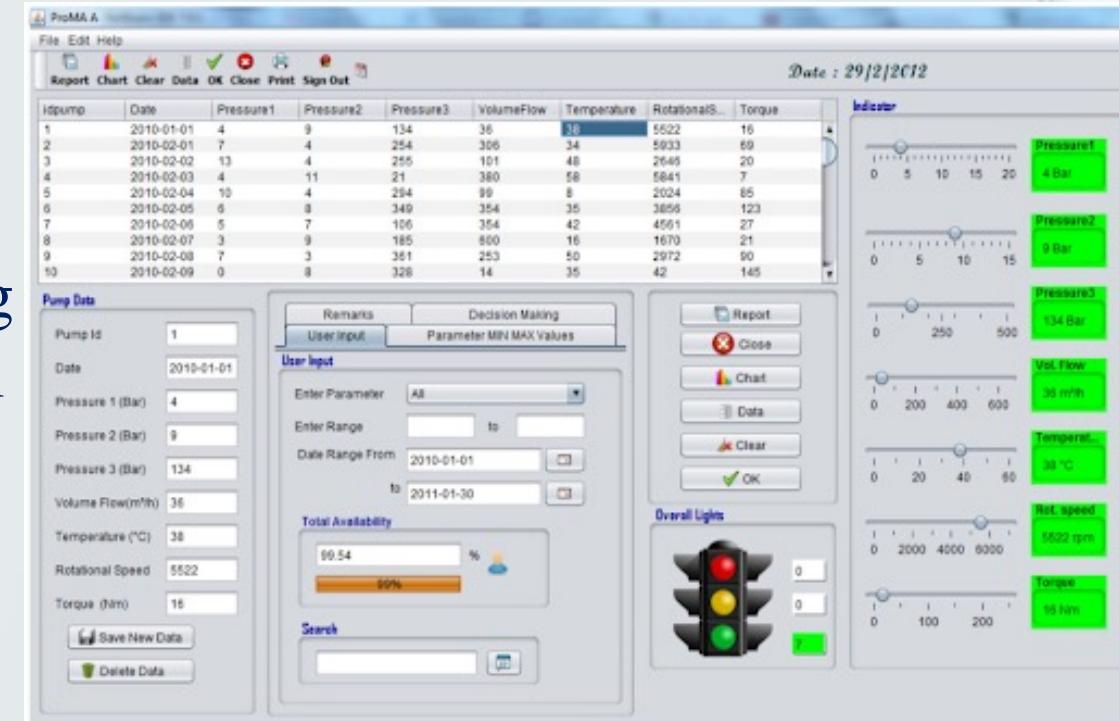
- Các thành phần mà người dùng có thể nhìn thấy và tương tác được.
- Cho phép nhận dữ liệu từ phía người dùng hoặc trình bày kết quả chương trình trả lại cho người dùng.

❑ Bố cục giao diện (Layouts)

- Cho phép định nghĩa cách các thành phần UI được bố trí trong một vật chứa (container).

❑ Hành vi (Behavior)

- Là những sự kiện phát sinh khi người dùng tương tác với các phần tử giao diện người dùng.



Khái niệm

❑ Lớp vật chứa cấp cao (*top-level*)

- JFrame
- JDialog
- JOptionPane
- JFileChooser
- JColorChooser

❑ Lớp vật chứa thứ cấp (*secondary*)

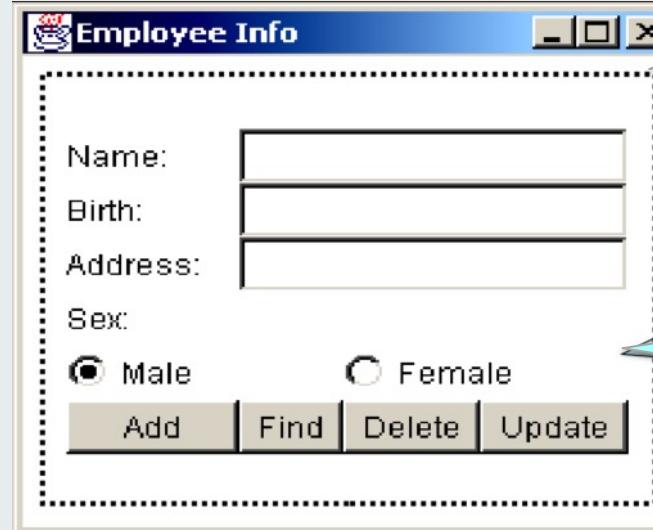
- Được sử dụng để nhóm và sắp xếp bố cục các thành phần
- Thường dùng là JPanel
- Các thành phần GUI **không** thể thêm trực tiếp vào 1 vật chứa cấp cao mà **chỉ** được thêm vào ô nội dung (*content-pane*) của nó.



Đưa 1 Component vào GUI

❑ Các bước để đưa 1 component vào GUI (viết code)

- Tạo 1 đối tượng component phù hợp
- Xác định hình thức bên ngoài lúc đầu của component.
- Định vị component này trên GUI
- Thêm component này vào GUI.
- Ví dụ



Container

Component

- 3 label
- 3 textField
- 1 radioButton groups chứa 2 radioButton
- 4 button

JFrame

- Sử dụng lớp `javax.swing.JFrame`.
- Là một cửa sổ có khung, icon, tiêu đề, các nút điều khiển (*thu nhỏ, phóng to, đóng cửa sổ*).
- Có thể di chuyển, thay đổi kích thước.
- Có thanh menu (tùy chọn), ô nội dung.
- Được dùng để chứa các thành phần khác.



Cấu trúc JFrame

JFrame

❑ Các hàm tạo thông dụng của JFrame

- `JFrame()`: khởi tạo một frame mới invisible (không hiển thị).
- `JFrame(String title)`: khởi tạo một frame mới invisible với tiêu đề được chỉ định.

❑ Một số phương thức cơ bản

- `setSize(int width, int height)`: đặt kích thước cho Jframe.
- `setLocation(int x,int y)`: đặt vị trí cho JFrame (mặc định thì một JFrame sẽ hiển thị ở vị trí góc trên – trái của màn hình).
- `setVisible(boolean b)`: cho phép JFrame ẩn/hiện.

JFrame

- `setDefaultCloseOperation(int operation)`: đặt hành động mặc định sẽ xảy ra khi người dùng “đóng” Frame. Mặc định là HIDE_ON_CLOSE, các lựa chọn khác gồm DO NOTHING_ON_CLOSE, DISPOSE_ON_CLOSE, EXIT_ON_CLOSE.
- `setTitle(String title)`: đặt tiêu đề cho JFrame.
- `setResizable(boolean b)`: đặt JFrame có được thay đổi kích thước hay không.
- `getContentPane()`: nhận về ô nội dung của JFrame để thêm các thành phần GUI vào.

JFrame

❑ Các bước cơ bản để tạo một cửa sổ JFrame

- Khởi tạo một đối tượng của lớp JFrame.
- Thiết lập kích thước cho JFrame.
- Thiết lập tiêu đề cho JFrame (nếu không đặt thì thanh tiêu đề sẽ trắng).
- Thiết lập hành động cho việc “đóng” JFrame.
- Cho phép JFrame hiển thị.

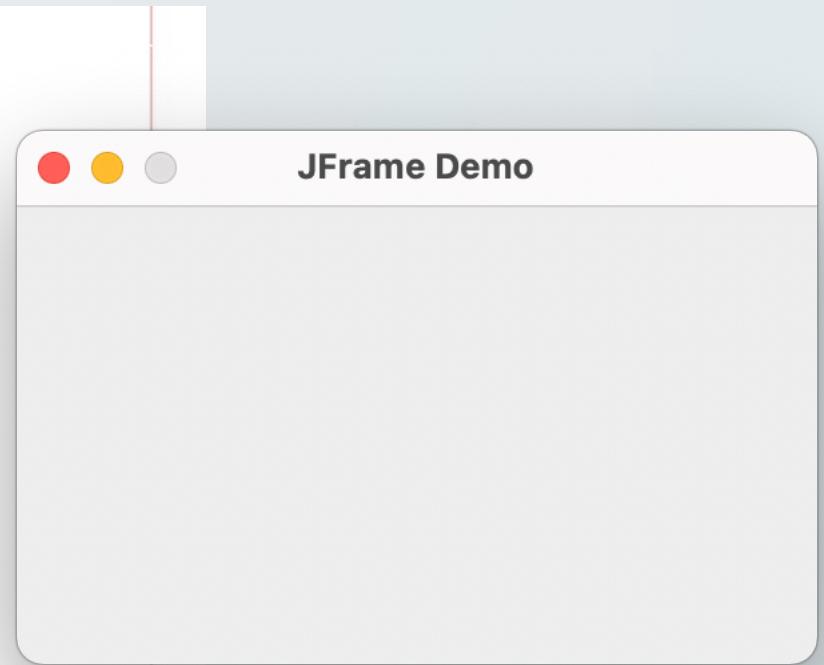
Ví dụ tạo JFrame đơn giản

```
import javax.swing.JFrame;

public class viDuJFrame {

    public viDuJFrame() {
        // khởi tạo frame
        JFrame frame = new JFrame();
        // đặt tiêu đề
        frame.setTitle("JFrame Demo");
        // đặt vị trí hiển thị frame
        frame.setLocation(300, 300);
        // không cho phép thay đổi kích thước
        frame.setResizable(false);
        // thiết lập kích thước cho frame
        frame.setSize(300, 300);
        // thiết lập hành động khi đóng frame
        frame.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        // hiển thị frame
        frame.setVisible(true);
    }

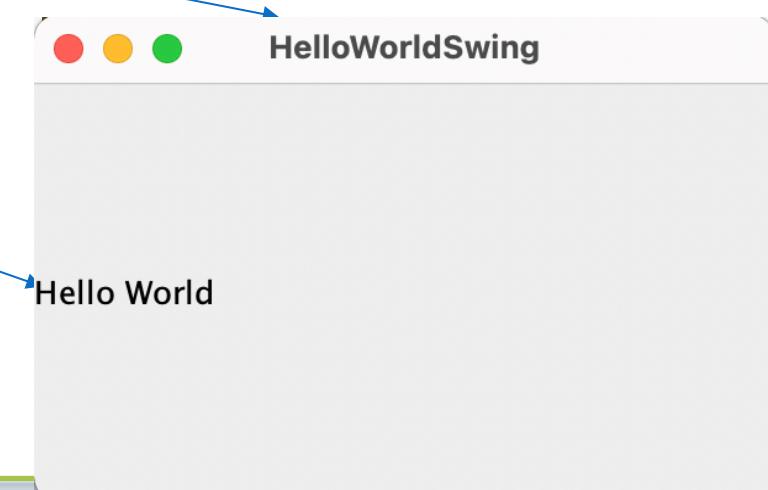
    public static void main(String args[]) {
        viDuJFrame frame = new viDuJFrame();
    }
}
```



Ví dụ: Tạo cửa sổ với Swing

- Ứng dụng HelloWorld cơ bản
- Tạo một cửa sổ với “HelloWorldString” trong thanh tiêu đề và hiển thị label “Hello World”

```
public class HelloWorldSwing extends JFrame{  
    public HelloWorldSwing() {  
        setTitle("HelloWorldSwing"); //Tiêu đề của JFrame  
        setSize(300, 200); //Kích thước của JFrame  
        setDefaultCloseOperation(EXIT_ON_CLOSE); //Thoát chương trình khi click nut exit  
        setLocationRelativeTo(null); //Canh giữa màn hình  
        setResizable(true); //cho phép thay đổi kích thước  
        JLabel label = new JLabel("Hello World");  
        getContentPane().add(label);  
    }  
    public static void main(String[] args) {  
        new HelloWorldSwing().setVisible(true);  
    }  
}
```



JDialog

- ❑ Sử dụng lớp `javax.swing.JDialog`.
- ❑ Cửa sổ dạng hộp thoại thông báo (*có khung viền và thanh tiêu đề*).
- ❑ Có thể được khởi tạo dạng modal hoặc không
 - Dạng “modal”: khi hiển thị hộp thoại thì khóa truy xuất của người dùng đến các cửa sổ khác.

JDialog

❑ Các hàm tạo thông dụng

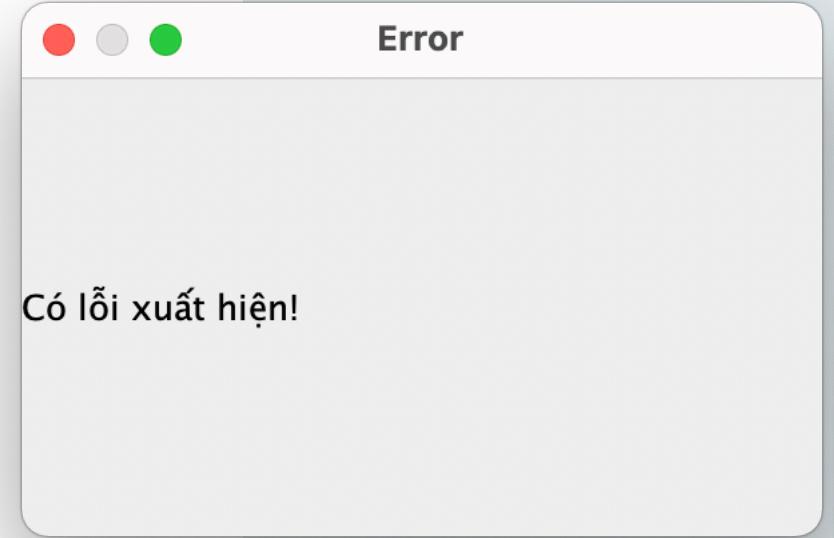
- **JDialog(Frame owner)**: tạo dialog dạng không “modal” với vật chủ frame được chỉ định, không có tiêu đề.
- **JDialog(Frame owner, boolean modal)**: có thể tạo dialog dạng “modal” với vật chủ frame được chỉ định, không có tiêu đề.
- **JDialog(Frame owner, String title)**: tạo dialog dạng không “modal” với vật chủ frame được chỉ định và tiêu đề được chỉ định.
- **JDialog(Frame owner, String title, boolean modal)**: có thể tạo dialog dạng “modal” với vật chủ frame được chỉ định và có tiêu đề được chỉ định.

❑ Một số phương thức cơ bản

- Như phương thức của JFrame.

Tạo JDialog hiển thị thông báo đơn giản

```
import javax.swing.*;  
public class viDuJDialog {  
  
    public viDuJDialog(JFrame frame) {  
        // Tạo dialog với tiêu đề dạng modal  
        JDialog dialog = new JDialog(frame, "Error", true);  
        // tạo nhãn label  
        JLabel label = new JLabel("Có lỗi xuất hiện!");  
        //thêm nhãn vào dialog  
        dialog.add(label);  
        // thiết lập kích thước cho dialog  
        dialog.setSize(200, 200);  
        // hiển thị dialog  
        dialog.setVisible(true);  
    }  
  
    public static void main(String args[]) {  
        JFrame frame = new JFrame(); //tạo frame làm vật chủ cho dialog  
        viDuJDialog dialog = new viDuJDialog(frame);  
    }  
}
```



JOptionPane

- ❑ Sử dụng lớp **javax.swing.JOptionPane**.
- ❑ Cung cấp các phương thức chuẩn để gọi một hộp thoại dialog chuẩn cho việc nhận một giá trị hoặc thông báo người dùng về một vấn đề gì đó.
- ❑ Được sử dụng rộng rãi hơn Jdialog vì có nhiều phương thức và nhiều tùy chọn hơn.

JOptionPane

❑ Các hàm tạo thông dụng

- **JOptionPane(Object message, int messageType):** tạo một thể hiện của JOptionPane để hiển thị một thông điệp với kiểu thông điệp đã cho và các tùy chọn mặc định.
- **JOptionPane(Object message, int messageType, int optionType):** tạo một thể hiện của JOptionPane để hiển thị một thông điệp với kiểu thông điệp đã cho và các tùy chọn được xác định.
- **JOptionPane(Object message, int messageType, int optionType, Icon icon):** tạo một thể hiện của JOptionPane để hiển thị một thông điệp với kiểu thông điệp, tùy chọn và icon đã cho.

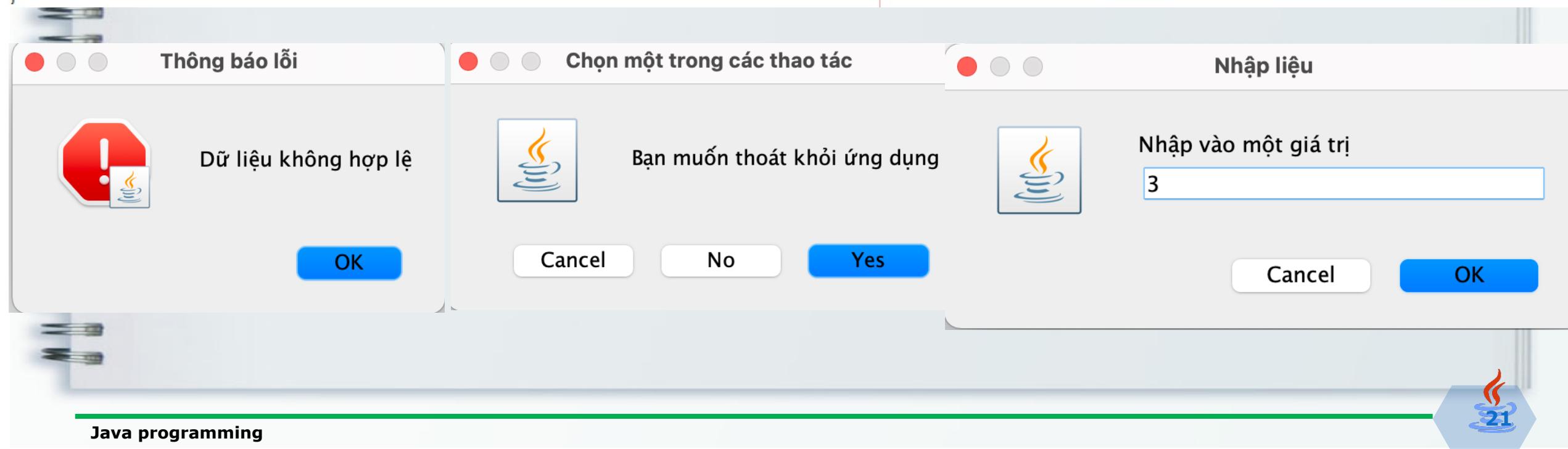
JOptionPane

❑ Phương thức thông dụng

- `showMessageDialog(Component parentComponent, Object message, String title, int messageType)`: hiển thị thông báo với thông điệp, tiêu đề và kiểu hộp thoại cho trước.
- `showConfirmDialog(Component parentComponent, Object message, String title, int optionType)`: hiển thị hộp thoại xác nhận với tiêu đề và thông điệp, có thể bao gồm các nút lệnh Yes, No và Cancel tùy theo giá trị biến tùy chọn.
- `showInputDialog(Component parentComponent, Object message, String title, int messageType)`: hiển thị hộp thoại nhập liệu với thông điệp, tiêu đề và kiểu hộp thoại cho trước.

Ví dụ sử dụng JOptionPane

```
import javax.swing.*;  
public class viDuJOptionPane {  
    public viDuJOptionPane() {  
        JFrame frame = new JFrame();  
        JOptionPane.showMessageDialog(frame, "Du lieu khong hop le!", "Thong bao loi", JOptionPane.ERROR_MESSAGE);  
        JOptionPane.showConfirmDialog(frame, "Ban muon thoat khoi ung dung!", "Chon mot trong cac thao tac", JOptionPane.YES_NO_CANCEL_OPTION);  
        JOptionPane.showInputDialog(frame, "Nhap vao mot gia tri", "Nhap lieu", JOptionPane.QUESTION_MESSAGE);  
    }  
    public static void main(String args[]) {  
        viDuJOptionPane option = new viDuJOptionPane();  
    }  
}
```



JFileChooser

- ❑ Sử dụng lớp **Javax.swing.JFileChooser**.
- ❑ Cho phép tạo một hộp thoại để người dùng chọn nhanh một tập tin.
- ❑ Các hàm tạo thông dụng
 - **JFileChooser()**: xây dựng một JFileChooser trả tới thư mục hiện hành của người dùng.
 - **JFileChooser(String currentDirectoryPath)**: xây dựng một JFileChooser trả tới thư mục được xác định bởi đường dẫn do người dung cung cấp.

JFileChooser

□ Phương thức thông dụng

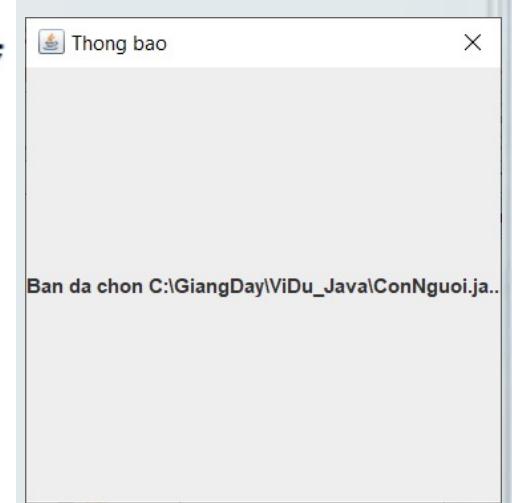
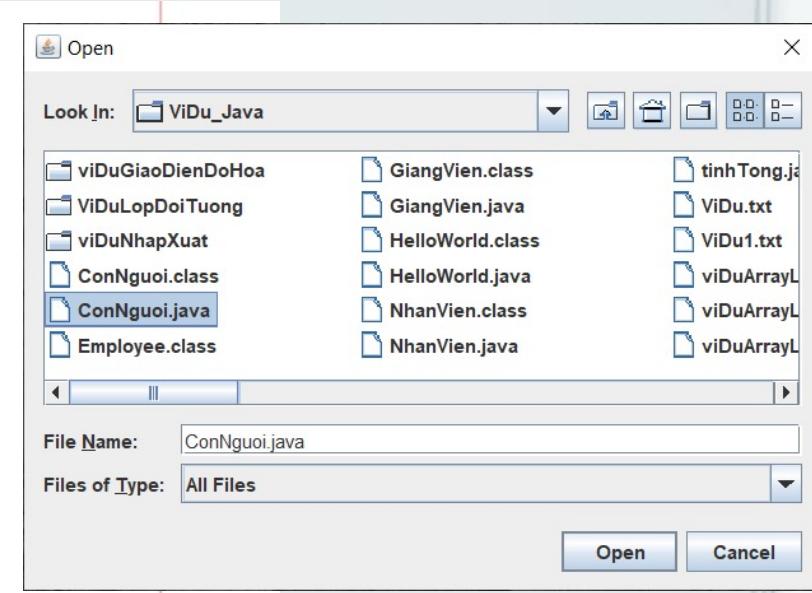
- `int showOpenDialog(Component parent)`: hiển thị hộp thoại dạng “Open file”.
- `int showSaveDialog(Component parent)`: hiển thị hộp thoại dạng “Save file”.
- `void setCurrentDirectory(File dir)`: cho phép thiết lập thư mục mặc định của hộp thoại được xác định bởi dir.

Ví dụ sử dụng JFileChooser

```

import javax.swing.*;
import java.io.*;
public class viDuJFileChooser {
    public viDuJFileChooser() {
        JFrame frame = new JFrame();
        JFileChooser filechooser = new JFileChooser();
        //mở hộp thoại dạng Open file
        int result = filechooser.showOpenDialog(frame);
        if(result == JFileChooser.APPROVE_OPTION) {
            File selectedFile = filechooser.getSelectedFile();
            //thiết lập dialog để hiển thị tập tin được chọn
            JDialog dialog = new JDialog(frame, "Thông báo", true);
            JLabel label = new JLabel("Bạn đã chọn "+selectedFile.getAbsolutePath());
            dialog.add(label);
            dialog.setSize(300, 300);
            dialog.setVisible(true);
        }
    }
    public static void main(String args[]) {
        viDuJFileChooser filechooser = new viDuJFileChooser();
    }
}

```

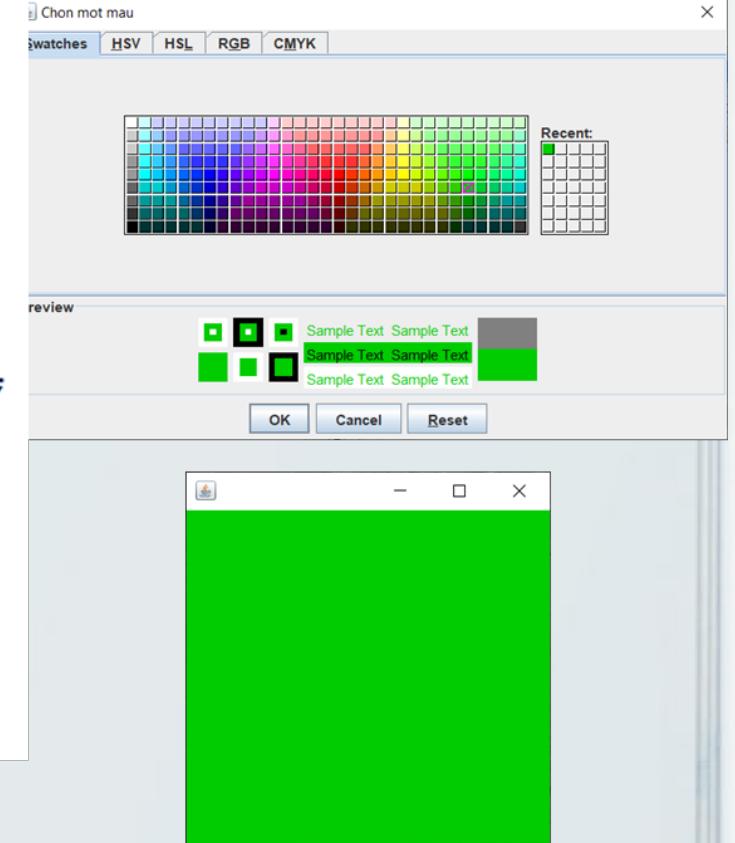


JColorChooser

- ❑ Sử dụng lớp **Javax.swing.JColorChooser**.
- ❑ Cho phép tạo một hộp thoại để người dùng chọn nhanh một màu (color).
- ❑ Các hàm tạo thông dụng
 - **JColorChooser()**: xây dựng một JColorChooser với màu được chọn mặc định là màu trắng.
 - **JColorChooser(color initialcolor)**: xây dựng một JColorChooser với màu mặc định được chọn do người dùng cung cấp.
- ❑ Phương thức thông dụng
 - **Color showDialog(Component c, String title, Color initialColor)**: cho phép hiển thị hộp thoại chọn màu.

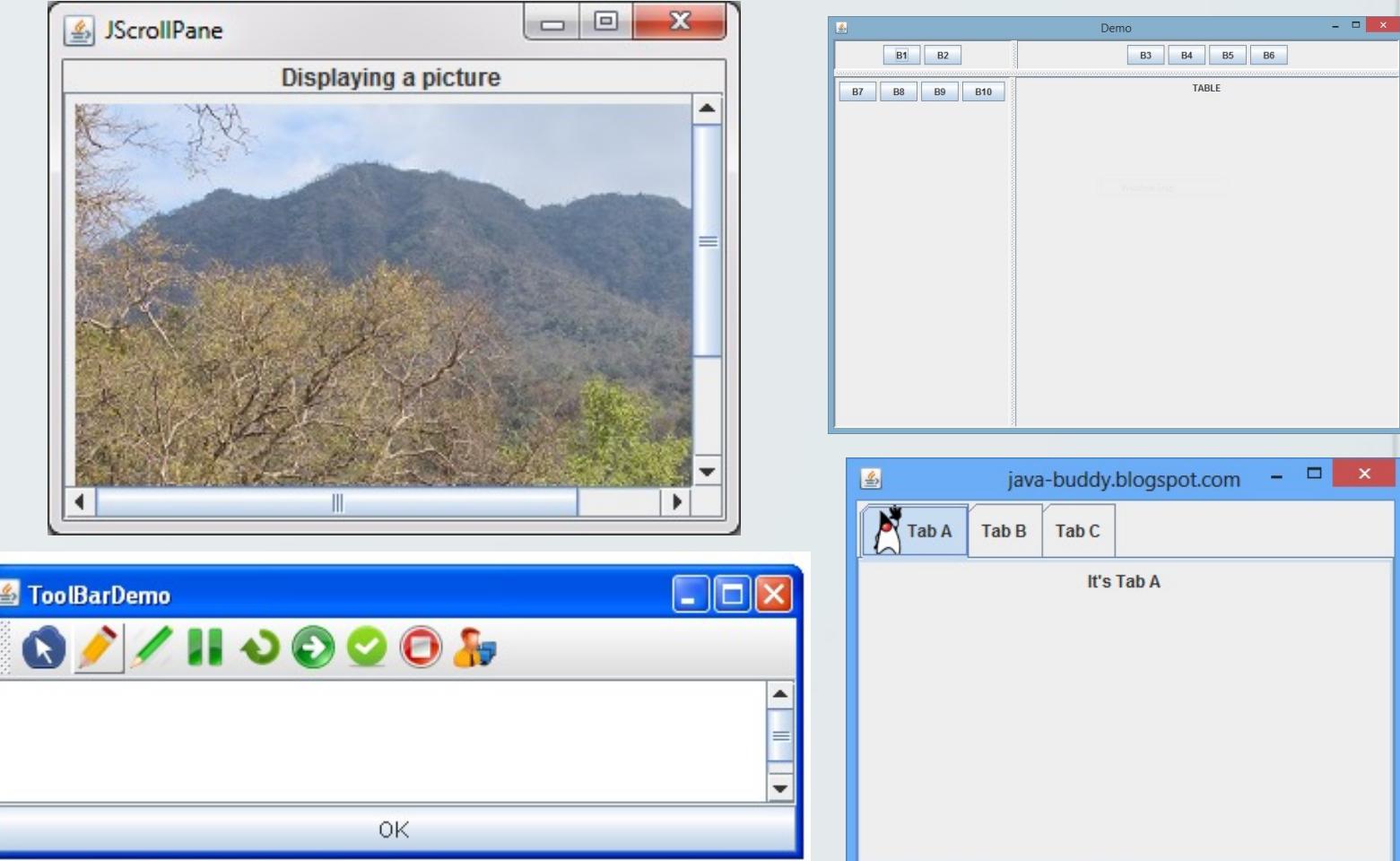
Ví dụ sử dụng JColorChooser

```
import javax.swing.*;
import java.awt.*;
public class viDuJColorChooser {
    public viDuJColorChooser(){
        JFrame frame = new JFrame();
        frame.setSize(300, 300);
        Color init = Color.GRAY;
        JColorChooser colorchooser = new JColorChooser();
        Color selectedColor = colorchooser.showDialog(frame,"Chon mot mau", init);
        //đổi màu nền cho frame
        frame.getContentPane().setBackground(selectedColor);
        frame.setVisible(true);
    }
    public static void main(String args[]) {
        viDuJColorChooser colorchooser = new viDuJColorChooser();
    }
}
```

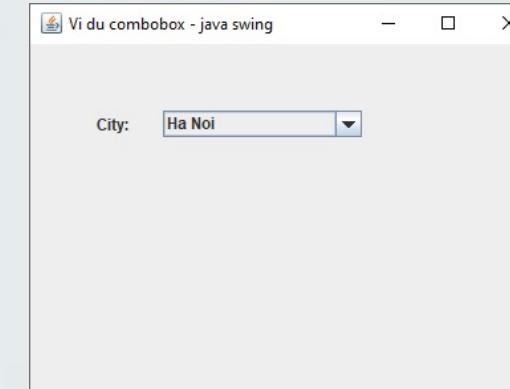
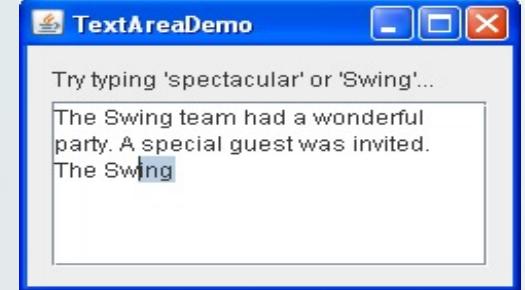


Một số loại vật chứa khác

- ❑ Dùng để “chứa” các thành phần GUI.
- ❑ Một số loại vật chứa thông dụng
 - JScrollPane
 - JSplitPane
 - JTabbedPane
 - JToolbar



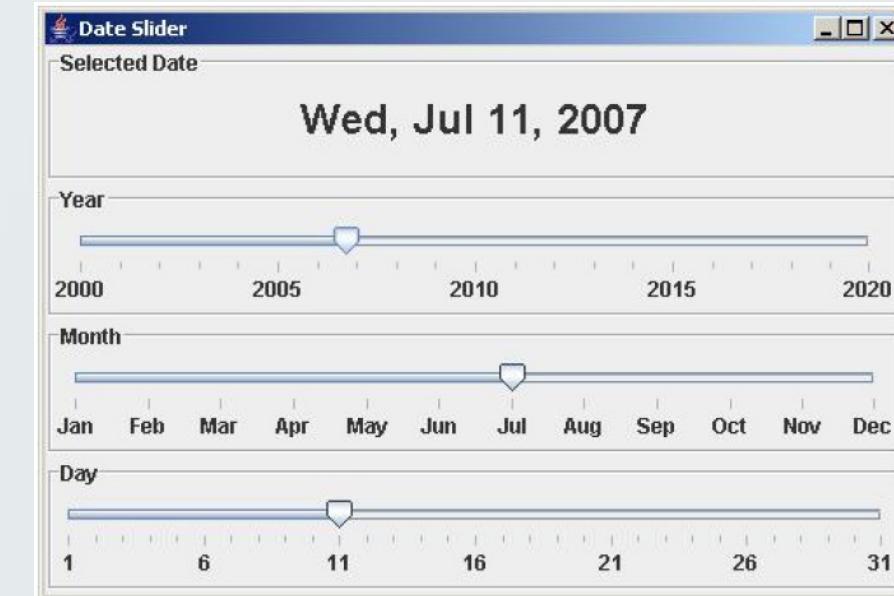
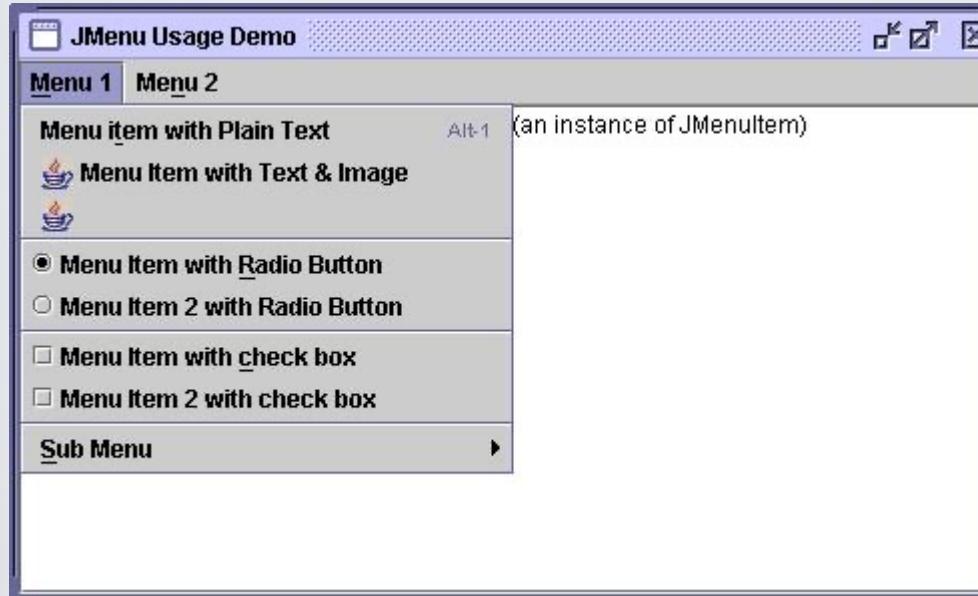
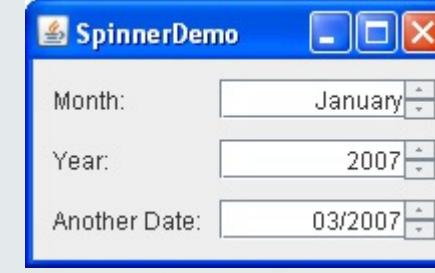
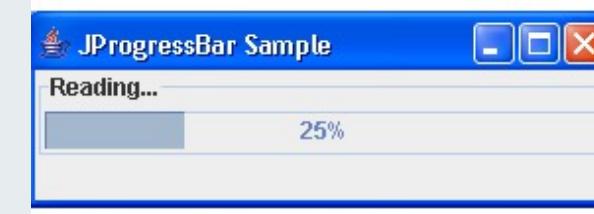
Các thành phần giao diện Swing



Swing List [Selection]

MS	Mississippi	Jackson
MT	Montana	Helena
NC	North Carolina	Raleigh
ND	North Dakota	Bismarck
NE	Nebraska	Lincoln
NH	New Hampshire	Concord
NJ	New Jersey	Trenton
NM	New Mexico	SantaFe
NV	Nevada	Carson City
NY	New York	Albany
OH	Ohio	Columbus

Giới thiệu



Giới thiệu

□ Phần lớn các thành phần giao diện Swing đều hỗ trợ

- Text và Icon
- Phím tắt để truy cập (keyboards shortcut)
- Chú giải (Tooltips)
- Look and Feel: giao diện giống như các thành phần của hệ điều hành

❑ Một số phương thức chung, thông dụng

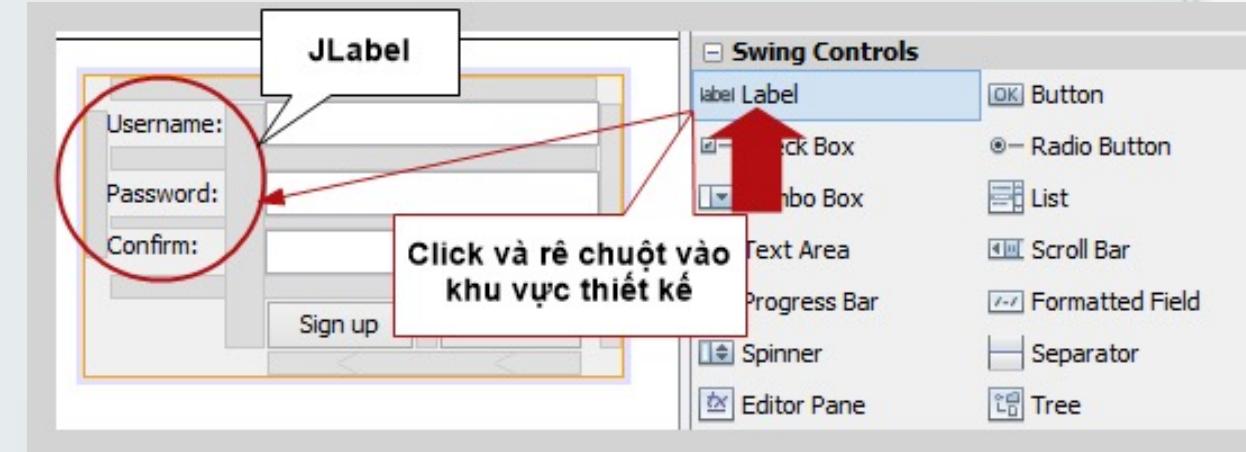
- `public void setBackground(Color bgColor)`: đặt màu nền
- `public void setForeground(Color fgcolor)`: đặt màu chữ
- `public void setFont(Font font)`: đặt font chữ
- `public void setBorder(Border border)`: thiết lập đường viền (độ dày, hình thái ...) cho thành phần swing
- `public void setPreferredSize(Dimension dim)`: thiết lập kích thước ưa thích (tối ưu)
- `public void setOpaque(boolean isOpaque)`: thiết lập chế độ “trong suốt”
- `public void setToolTipText(String toolTipMsg)`: đặt chú giải sẽ hiển thị khi người dùng rê chuột vào thành phần.

- ❑ Sử dụng lớp `javax.swing.JLabel`
- ❑ Cho phép hiển thị hoặc text, hoặc hình ảnh hoặc cả hai.
- ❑ 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.
- ❑ Hàm tạo thông dụng
 - `JLabel(String text)`: tạo một JLabel với nhãn là text đã cho.
 - `JLabel(Icon image)`: tạo một JLabel với hình ảnh là image đã cho.
 - `JLabel(String text, Icon icon, int horizontalAlignment)`: tạo một JLabel với text, hình ảnh, và căn chỉnh ngang.

JLabel

□ Thêm JLabel vào vật chứa

- `jframe.add(jLabel1);`
- `jdialog.add(jLabel2);`
- `jpanel.add(jLabel3);`



□ Các phương thức thường dùng

- `void setText(String text)`: đặt lại giá trị nhãn cho JLabel.
- `void setIcon(Icon icon)`: định nghĩa icon mà JLabel sẽ hiển thị.

- Sử dụng lớp javax.swing.JTextField.**
- Cho phép hiển thị, nhập một dòng text đơn.**
- Hàm tạo thông dụng**
 - `JTextField()`: tạo một JTextField rỗng.
 - `JTextField(int columns)`: tạo một JTextField rỗng với số cột đã cho.
 - `JTextField(String text)`: tạo một JTextField với text đã cho.
 - `JTextField(String text, int columns)`: tạo một JTextField với text và các cột đã cho.

JTextField

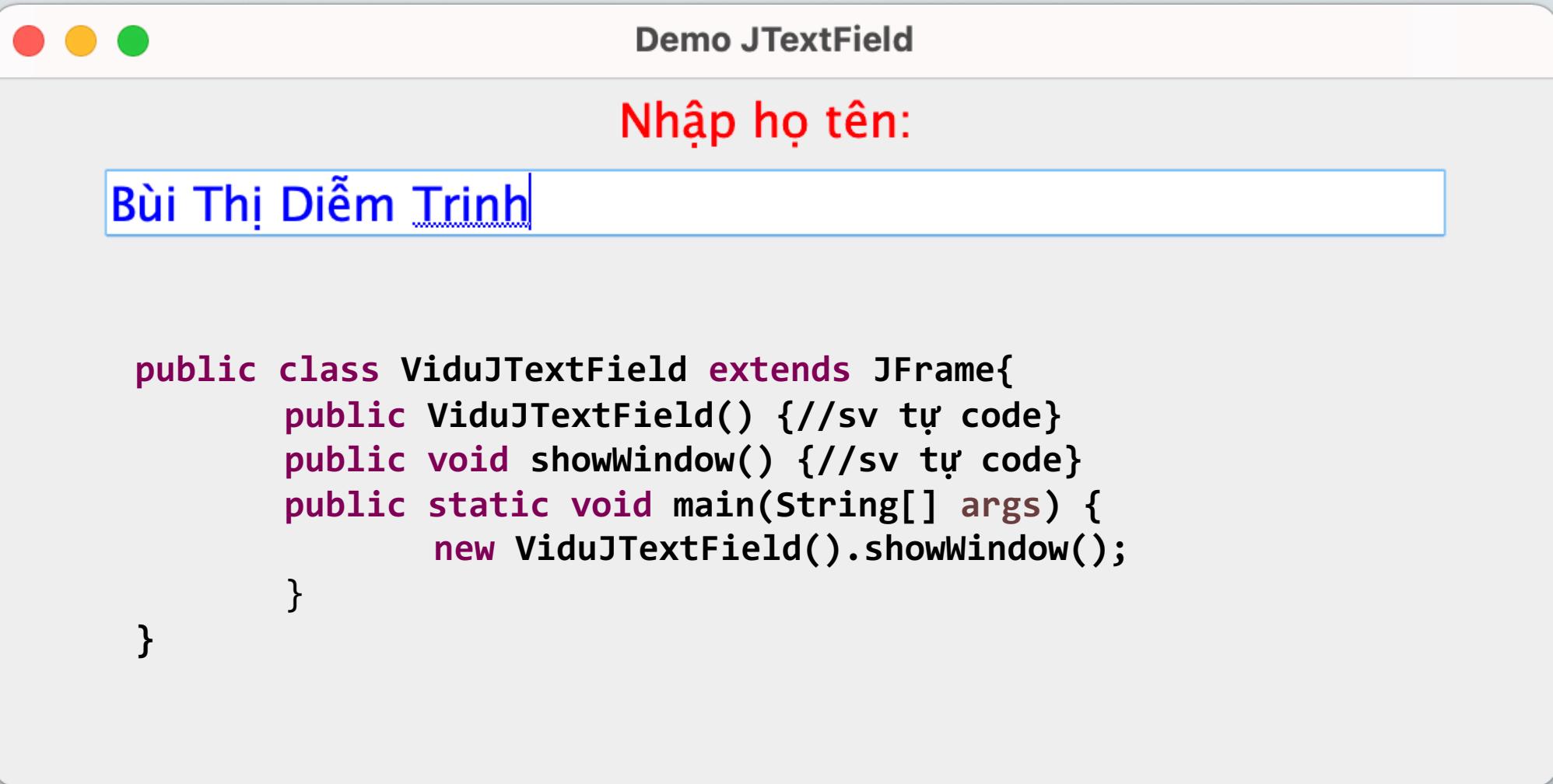
❑ Thêm JTextField vào vật chứa

- `jframe.add(jtextfield1);`
- `jdialog.add(jtextfield2);`
- `jpanel.add(jtextfield3);`

❑ Phương thức thông dụng

- `String getText()`: trả về giá trị được nhập vào JTextField.
- `void setText(String text)`: đặt lại giá trị của JTextField là text.
- `void setEnabled(Boolean enable)`: thiết lập việc được phép hay không được phép nhập liệu với JTextField.
- `void requestFocus()`: yêu cầu được gán focus.

Demo JTextField



Demo JTextField

```
public VieuJTextField() {  
    Container con = getContentPane();  
    JPanel pn = new JPanel();  
    pn.setLayout(new FlowLayout());  
    //  
    JLabel lb = new JLabel("Nhập họ tên: ");  
    lb.setForeground(Color.RED);  
    lb.setSize(150, 180);  
    lb.setFont(new Font("VNI", Font.PLAIN, 18));  
  
    JTextField tf = new JTextField(30);  
    tf.setSize(150, 180);  
    tf.setFont(new Font("VNI", Font.PLAIN, 18));  
    tf.setForeground(Color.BLUE);
```

```
pn.add(lb);  
pn.add(tf);  
//  
con.add(pn);  
}
```

```
private void showWindow() {  
    setTitle("Demo JTextField");  
    setSize(600, 300);  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    setLocationRelativeTo(null);  
    setVisible(true);  
}
```

JPasswordField

- Sử dụng lớp javax.swing.JPasswordField.**
- Thừa kế từ JTextField.**
- Cho phép nhập dòng text đơn nhưng các ký tự hiển thị là ký tự đại diện.**
- Hàm tạo thông dụng**
 - JPasswordField(): tạo một JPasswordField rỗng.
 - JPasswordField(int columns): tạo một JPasswordField rỗng với số cột đã cho.
 - JPasswordField(String text): tạo một JPasswordField với text đã cho.
 - JPasswordField(String text, int columns): tạo một JPasswordField với text và các cột đã cho.

JPassworldField

❑ Phương thức cần dùng

- void setEchoChar(char c): thiết lập ký tự hiển thị của JPasswordField.



JTextArea

- ❑ Sử dụng lớp `javax.swing.JTextArea`.
- ❑ Có tính năng tương tự như `JTextField` nhưng có thể bao gồm nhiều dòng.
- ❑ Hàm tạo thông dụng
 - `JTextArea()`: tạo một TextArea rỗng.
 - `JTextArea(String s)`: tạo một TextArea với text đã cho.
 - `JTextArea(int row, int column)`: tạo một TextArea rỗng với số hàng và cột đã cho.
 - `JTextArea(String s, int row, int column)`: tạo một TextArea với text, số hàng và cột đã cho.

JTextArea

- ❑ Thêm JTextArea vào vật chứa giống như cách thêm JTextField
- ❑ Phương thức thông dụng

- void `textArea.setEditable (Boolean b)`: cho phép hoặc không người dùng sửa nội dung.
- `JScrollPane scrollPane = new JScrollPane (textArea)`: bổ sung thanh cuộn.
- `void textArea.append (String s)`: cộng thêm chuỗi vào nội dung của JTextArea.
- `void textArea.setLineWrap (Boolean b)`: cho phép nội dung của JTextArea tự động xuống hàng hay không.

JCheckBox

- ❑ Sử dụng lớp **javax.swing.JCheckBox**.
- ❑ Là một thành phần mà có thể được lựa chọn (selected) hoặc không được lựa chọn (unselected), và hiển thị trạng thái của nó tới người dùng.
- ❑ Việc chọn hay không chọn 1 checkbox nào đó là độc lập với checkbox khác.

☐ Hàm tạo thông dụng

- **JCheckBox()**: tạo một unselected checkbox ban đầu không có text và icon.
- **JCheckBox(String text)**: tạo một unselected checkbox ban đầu với nhãn được chỉ định bởi text.
- **JCheckBox(String text, boolean selected)**: tạo một checkbox với nhãn là text và xác định rằng ban đầu nó là selected hoặc không.
- **JCheckBox(String text, Icon icon, boolean selected)**: tạo một checkbox có nhãn và icon, đồng thời xác định rằng ban đầu nó là selected hoặc không.

JCheckBox

❑ Phương thức cần dùng

- Boolean isSelected(): cho phép kiểm tra checkbox có được chọn hay không.

JRadioButton

- Sử dụng lớp `Sử dụng lớp javax.swing.JRadioButton.`**
- Có tác dụng như JCheckBox nhưng được sử dụng theo nhóm và khi đó chỉ có một JRadioButton được chọn.**
- Hàm tạo thông dụng**
 - `JRadioButton()`: tạo một unselected JRadioButton ban đầu không có text và icon.
 - `JRadioButton(String text)`: tạo một unselected JRadioButton ban đầu với nhãn được chỉ định bởi text.
 - `JRadioButton(String text, boolean selected)`: tạo một JRadioButton với nhãn là text và xác định rằng ban đầu nó là selected hoặc không.
 - `JRadioButton(String text, Icon icon, boolean selected)`: tạo một JRadioButton có nhãn và icon, đồng thời xác định rằng ban đầu nó là selected hoặc không.

❑ Dùng ButtonGroup để tạo nhóm

- `ButtonGroup group = new ButtonGroup();`
- `group.add(option1);`
- `group.add(option2);`

- ❑ Sử dụng lớp `javax.swing.JComboBox`.
- ❑ Là một thành phần có sự kết hợp giữa một button, một trường có thể chỉnh sửa và một danh sách xổ xuống. Tại một thời điểm chỉ có một phần tử có thể được lựa chọn từ danh sách.
- ❑ Hàm tạo thông dụng
 - `JComboBox()`: tạo một JComboBox với danh sách phần tử rỗng.
 - `JComboBox(Object[] items)`: tạo một JComboBox chứa các phần tử trong mảng tham số.
 - `JComboBox(Vector items)`: tạo một JComboBox chứa các phần tử trong Vector tham số.

JComboBox

□ Phương thức thông dụng

- `void addItem(Object anObject)`: được sử dụng để thêm một phần tử mới vào danh sách phần tử của JComboBox.
- `void removeItem(Object anObject)`: được sử dụng để xóa một phần tử ra khỏi danh sách phần tử của JComboBox.
- `void removeAllItems()`: được sử dụng để xóa tất cả phần tử của danh sách của JComboBox.
- `void setEditable(boolean b)`: được sử dụng để xác định xem có hay không JComboBox là được nhập vào phần tử không có danh sách.
- `Object getSelectedItem()`: trả về phần tử được chọn.

JList

- ❑ Sử dụng lớp **javax.swing.JList**.
- ❑ Cho phép hiển thị một danh sách các phần tử và cho phép người dùng lựa chọn một hoặc nhiều phần tử.
- ❑ Hàm tạo thông dụng
 - **JList()**: xây dựng một JList rỗng và chỉ đọc.
 - **JList(Object[] listData)**: xây dựng một JList bao gồm các phần tử trong mảng đã cho.
 - **JList(Vector listData)**: xây dựng một JList bao gồm các phần tử trong Vector đã cho.

JList

□ Phương thức thông dụng

- Các phương thức tương tự như của lớp JComboBox.
- `List getSelectedValuesList()`: trả về danh sách các phần tử được chọn.
- `int [] getSelectedValuesList()`: trả về mảng chỉ số của các phần tử được chọn.
- `Object getElementAt(int index)`: trả về phần tử tại vị trí index của JList.

JButton

- ❑ Sử dụng lớp **javax.swing.JButton**.
- ❑ Được sử dụng để tạo một nút có thể nhấn (click). Thành phần này có một nhãn và tạo một sự kiện (event) khi được nhấn. Chúng ta cũng có thể chèn ảnh vào nút lệnh (button).
- ❑ Phải bổ sung một “lắng nghe” sự kiện “click” để xử lý khi người dùng nhấn nút lệnh.
- ❑ **Hàm tạo thông dụng**
 - JButton(Icon icon): tạo một button với một icon.
 - JButton(String text): tạo một button với nhãn của nút được xác định bởi text.
 - JButton(String text, Icon icon): tạo một button với nhãn của nút được xác định bởi text và một icon.

JButton

❑ Thêm 1 nút bấm vào 1 container

- `jframe.add(jButton);`
- `jdialog.add(jButton);`
- `jpanel.add(jButton);`

❑ Thiết lập shortcut key

- `jbutton.setMnemonic('C');`//tổ hợp phím Alt + C

❑ Thiết lập là nút lệnh mặc định (thực thi xử lý tương ứng khi người dùng nhấn Enter)

- `getRootPane().setDefaultButton(jButton);`

JProgressBar

- ❑ Sử dụng lớp **javax.swing.JProgressBar**.
- ❑ Cho phép hiển thị tiến trình của 1 tác vụ nào đó. Cho phép hiển thị tỷ lệ phần trăm khối lượng tác vụ đã hoàn thành.
- ❑ Hàm tạo thông dụng
 - **JProgressBar()**: được sử dụng để tạo một thanh tiến trình ngang nhưng không có nhãn.
 - **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 hướng đã cho, có thể là Vertical hoặc Horizontal bằng cách 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.

□ Phương thức thông dụng

- `void setString(String s)`: được sử dụng để thiết lập giá trị nhãn cho tiến trình.
- `void setValue(int value)`: được sử dụng để thiết lập, cập nhật lại giá trị hiện tại trên thanh tiến trình.

❑ Sử dụng lớp javax.swing.JSpinner.

- Cho phép người dùng lựa chọn một số hoặc một giá trị đối tượng trong một dãy các phần tử đã qua sắp xếp.

❑ Hàm tạo thông dụng

- **JSpinner():** xây dựng một spinner sử dụng một Integer SpinnerNumberModel với giá trị khởi tạo 0 và không có giới hạn lớn nhất, nhỏ nhất.
- **JSpinner(SpinnerModel model):** Xây dựng một spinner đầy đủ với một cặp nút next/previous và một editor cho SpinnerModel.

JSpinner

❑ Lớp SpinnerListModel

- **SpinnerListModel(List<?> values)**: khởi tạo đối tượng SpinnerModel bao gồm các phần tử được định nghĩa bởi danh sách List.
- **SpinnerListModel(Object[] values)**: khởi tạo đối tượng SpinnerModel bao gồm các phần tử được định nghĩa bởi mảng các đối tượng.

□ Phương thức thông dụng

- **Object getValue()**: trả về giá trị hiện tại đang được chọn, và đó là giá trị đang được hiển thị.
- **Object getNextValue()**: trả về giá trị đứng sau giá trị hiện tại trong danh sách.
- **Object getPreviousValue()**: trả về giá trị đứng trước giá trị hiện tại trong danh sách

❑ Sử dụng lớp javax.swing.JSlider.

- Cho phép người dùng lựa chọn một giá trị số trong một phạm vi giới hạn bằng cách di chuyển thanh trượt trên slider. Điều này đảm bảo giá trị nhập vào luôn thuộc khoảng phạm vi định trước.

❑ Hàm tạo thông dụng

- `JSlider(int min, int max, int value)`: tạo một thanh slider ngang sử dụng giá trị min, max và value (giá trị hiện hành) đã cho.
- `JSlider(int orientation, int min, int max, int value)`: tạo một slider sử dụng orientation (`JSlider.HORIZONTAL` hoặc `JSlider.VERTICAL`), min, max và value đã cho.

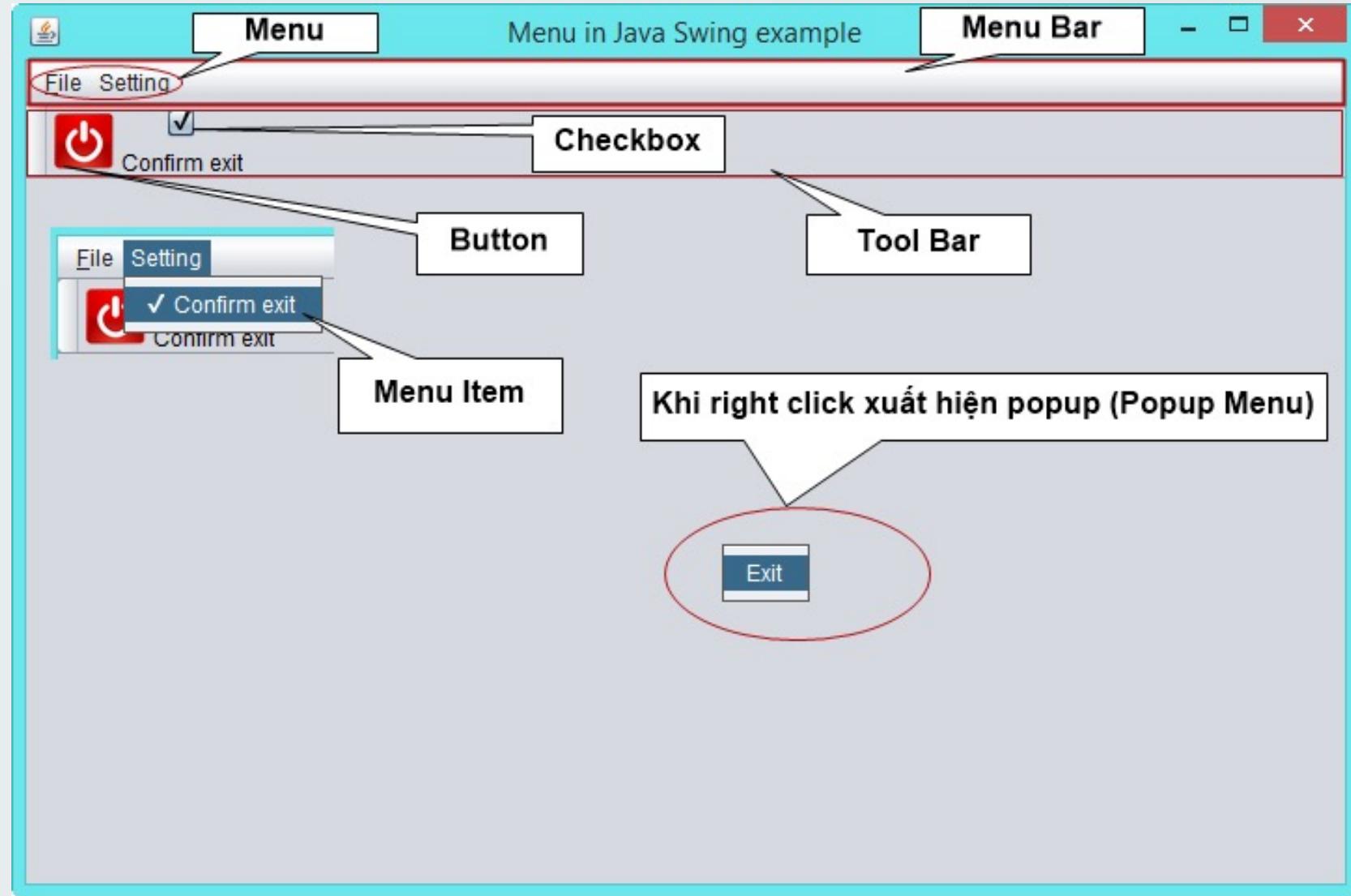
JSlider

□ Phương thức thông dụng

- `int getValue()`: trả về giá trị hiện hành của slider.
- `void setValue(int value)`: thiết lập giá trị hiện hành của slider.
- `void setMinorTickSpacing(int n)`: thiết lập bước nhảy cho slider.

- ❑ Hệ thống menu cung cấp một **thanh menu (Menu Bar)** bên dưới thanh tiêu đề của cửa sổ.
- ❑ Thanh menu này chứa nhiều thành phần và được gọi là **Menu**.
- ❑ Khi chúng ta nhấn vào một menu, nó sẽ hiển thị các lựa chọn khác nhau và được gọi là **Menu Item**.
- ❑ Khi chúng ta nhấn chuột phải (right-click) trên màn hình, một **Menu** chứa các Menu Item xuất hiện bên cạnh nơi mà chúng ta nhấn chuột và được gọi là **Popup Menu**.

Menu trong Java



Menu trong Java

Java Swing có sẵn 3 lớp hỗ trợ tạo menu là: **JMenuBar**, **JMenu** và **JMenuItem**.

□ **Thứ tự thực hiện như sau**

- 1- Tạo một đối tượng menubar từ lớp JMenuBar.
- 2- Dùng lớp JMenu để tạo một menu, sau đó có thể thiết lập phím tắt cho menu này.
- 3- Tạo các item trong một menu từ lớp JMenuItem với tên item và ảnh icon (nếu cần thiết), sau đó có thể gán phím tắt cho menu con này.
- 4- Thêm item vào menu rồi thêm menu đó vào menubar.

□ **Xem cách khởi tạo, phương thức tại**

- <https://docs.oracle.com/javase/8/docs/api/javax/swing/JMenuBar.html>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/JMenu.html>
- <https://docs.oracle.com/javase/7/docs/api/javax/swing/JMenuItem.html>

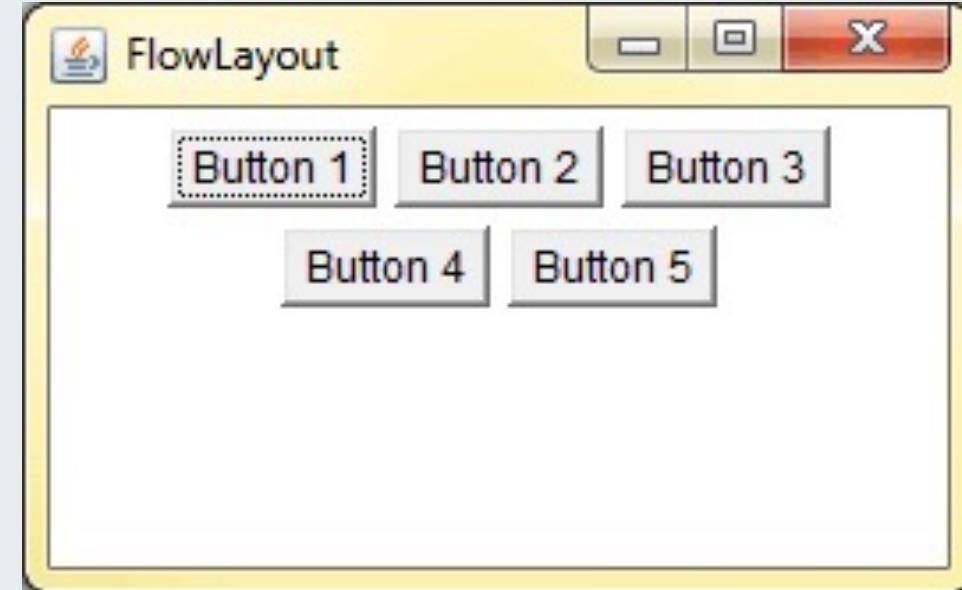
❑ AWT và Swing cung cấp nhiều kiểu sắp xếp bố cục (cho phép xác định vị trí và kích thước của các thành phần):

- java.awt.BorderLayout
- java.awt.FlowLayout
- java.awt.GridLayout
- java.awt.GridBagLayout
- javax.swing.BoxLayout
- javax.swing.GroupLayout
- javax.swing.SpringLayout

- ❑ **Bố cục mặc định là FlowLayout.**
- ❑ **Cài đặt bố cục**
 - Khởi tạo bố cục.
 - Sử dụng phương thức setLayout của vật chứa để thiết lập bố cục.
- ❑ **Thêm thành phần GUI vào vật chứa với tham số vị trí thích hợp với bố cục được thiết lập ở bước trên.**
- ❑ **Có thể sử dụng NetBeans, Eclipse để có thể bố trí các thành phần GUI một cách dễ dàng hơn.**

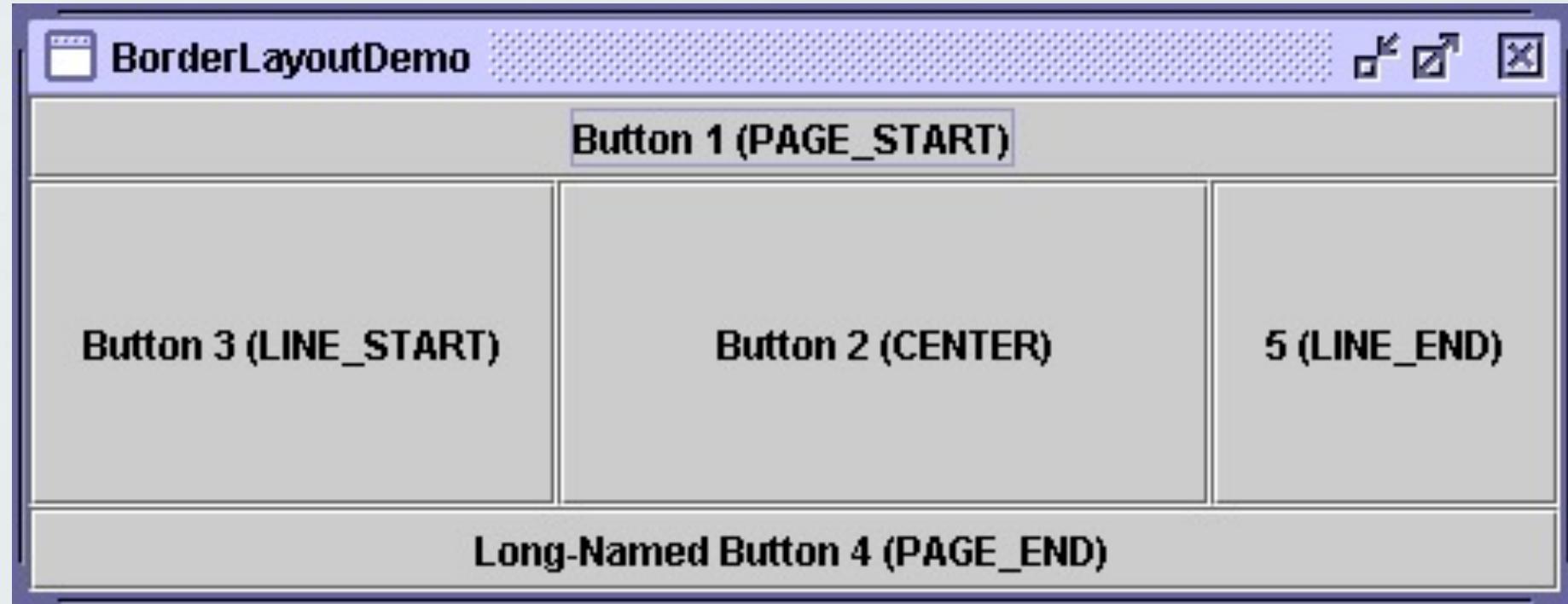
Sắp xếp bố cục

- ❑ Là cách bố cục mặc định của mỗi một JPanel.
- ❑ Các thành phần được đặt trong khung theo thứ tự được thêm vào vật chứa, nếu vượt quá chiều ngang thì sẽ chuyển xuống hàng dưới.



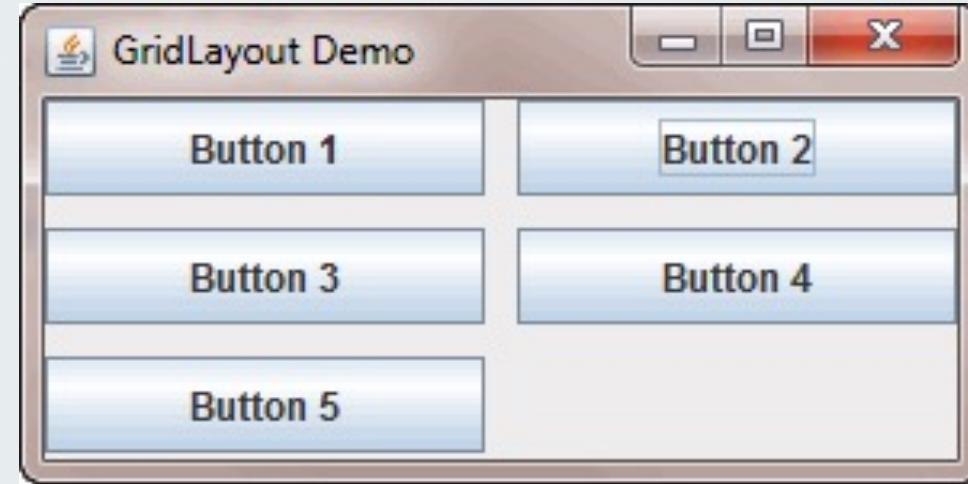
BorderLayout

- ❑ Mọi contentpane luôn được khởi tạo với bố cục BorderLayout.
- ❑ JToolBar khi tạo ra phải thuộc BorderLayout.
- ❑ Có 5 vị trí: xung quanh và ở giữa.



GridLayout

- ❑ Dạng lưới được xác định bằng số hàng, số cột.
- ❑ Kích thước các ô trong lưới là bằng nhau.



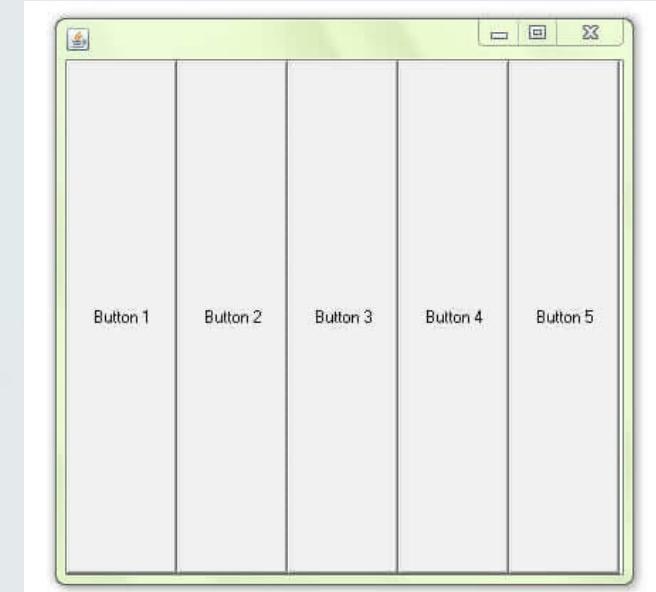
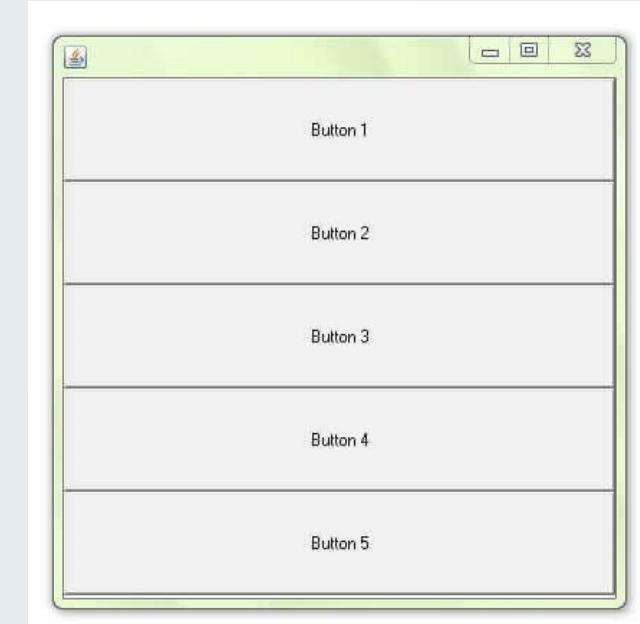
GridBagLayout

- ❑ Phức tạp, mạnh mẽ và mềm dẻo hơn bố cục GridLayout.
- ❑ Dạng lưới trong đó các hàng có thể có chiều cao khác nhau, các cột có thể có độ rộng khác nhau.



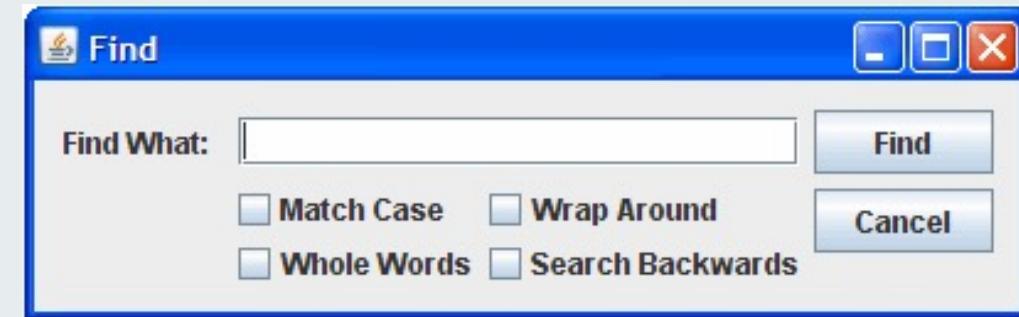
BoxLayout

- ❑ Được sử dụng để sắp xếp các thành phần hoặc theo chiều dọc hoặc theo chiều ngang.
- ❑ Cách bố trí được xác định bởi 4 hằng số
 - X_AXIS
 - Y_AXIS
 - LINE_AXIS
 - PAGE_AXIS



GroupLayout

- ❑ Nhóm các thành phần theo cấu trúc thứ bậc để đặt chúng trong một vật chứa.
- ❑ Các dòng và cột được bố trí riêng biệt.
- ❑ Linh hoạt, kích thước và vị trí mỗi thành phần độc lập.
- ❑ Nên dùng NetBeans để bố cục dạng này.



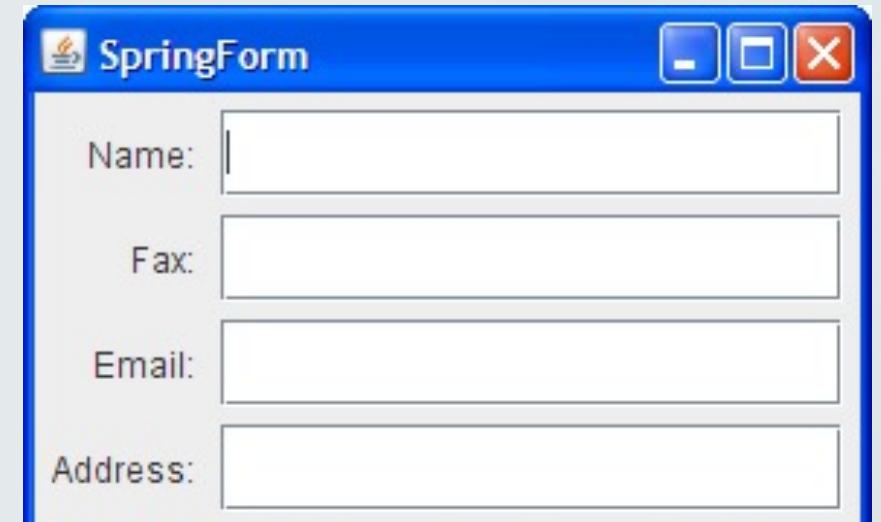
SpringLayout

❑ Đặt vị trí các thành phần con của Container liên kết với nó theo một tập các ràng buộc.

- **BASELINE**: Xác định baseline của một thành phần.
- **EAST**: Xác định cạnh phải của hình chữ nhật bao của một thành phần.
- **HEIGHT**: Xác định chiều cao của hình chữ nhật bao của một thành phần.
- **HORIZONTAL_CENTER**: Xác định sự căn chỉnh ngang của hình chữ nhật bao của một thành phần.
- **NORTH**: Xác định cạnh trên của hình chữ nhật bao của một thành phần.
- **SOUTH**: Xác định cạnh dưới của hình chữ nhật bao của một thành phần.

SpringLayout

- **VERTICAL_CENTER**: Xác định sự căn chỉnh dọc của hình chữ nhật bao của một thành phần.
- **WEST**: Xác định cạnh trái của hình chữ nhật bao của một thành phần.
- **WIDTH**: Xác định độ rộng của hình chữ nhật bao của một thành phần.
- Được sử dụng để xây dựng GUI Builder.



❑ Xử lý sự kiện sẽ liên quan đến 3 đối tượng:

- **Đối tượng nguồn (source)**, đối tượng phát sinh sự kiện: Button, TextField, ...
- **Sự kiện (event)**: sự kiện phát sinh khi 1 đối tượng nguồn bị tác động.
 - Ví dụ: 1 Button được bấm, 1 ComboBox thay đổi giá trị được chọn, 1 cửa sổ được đóng, ...
- **Bộ lắng nghe sự kiện (listener)**: khi sự kiện được tạo ra, nó sẽ gửi thông báo đến các bộ nghe (đã được đăng ký). Phương thức xử lý sự kiện tương ứng sẽ được kích hoạt.

Các loại sự kiện và bộ lắng nghe tương ứng

Event Classes	Listener Interfaces
ActionEvent	ActionListener
MouseEvent	MouseListener and MouseMotionListener
MouseWheelEvent	MouseWheelListener
KeyEvent	KeyListener
ItemEvent	ItemListener
TextEvent	TextListener
AdjustmentEvent	AdjustmentListener
WindowEvent	WindowListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener

❑ **Button, MenuItem**

- `void addActionListener(ActionListener a)`: lắng nghe sự kiện click trên nút lệnh, mục menu.

❑ **TextField, TextArea**

- `void addTextListener(TextListener a)`: lắng nghe sự kiện nội dung text bị thay đổi.
- `void addKeyListener(KeyListener a)`: lắng nghe sự kiện mỗi khi một phím trên bàn phím được nhấn, nhả.

❑ **Checkbox, RadioButton, ComboBox, ListBox**

- `void addItemListener(ItemListener a)`: lắng nghe sự kiện một thành phần được nhấn.
- ...

Các bước xử lý sự kiện

1) Tạo lớp lắng nghe sự kiện có cài đặt (implements) bộ lắng nghe sự kiện tương ứng

- class ButtonClickListener implements ActionListener.
- Tuy nhiên có thể sử dụng các bộ lắng nghe nguyên thủy do Java cài đặt.

2) Đăng ký bộ lắng nghe cho nguồn (source)

- okButton.addActionListener(new ButtonClickListener());
- okButton.addActionListener(this);

3) Tái định nghĩa các hàm xử lý sự kiện

```
public void actionPerformed(ActionEvent e) {  
    String command = e.getActionCommand();  
    if( command.equals( "OK" ) ) {  
        // ...  
    }  
}
```

❑ Sử dụng Frame được bố cục theo FlowLayout với 1 trường TextField nhập vào User name và 1 trường PasswordField để nhập Password cùng nút lệnh có nhãn Sign in. Khi người dùng nhấn nút lệnh thì sử dụng Dialog hiển thị thông tin đã được nhập vào.

❑ Hướng dẫn

- Tạo một frame
- Thiết lập kiểu bố cục FlowLayout, canh giữa (ví dụ)
- Tạo các label, textfield, passwordfield, button
- Đăng ký lắng nghe sự kiện click cho nút lệnh
- Thêm vào frame
- Tái định nghĩa hàm xử lý sự kiện

Ví dụ xử lý sự kiện

```
public void xuLySuKien_1() {  
    frame = new JFrame();  
    frame.setSize(500, 120);  
    FlowLayout layout = new FlowLayout(FlowLayout.CENTER);  
    frame.setLayout(layout);  
    JLabel userLabel = new JLabel("Username:");  
    JLabel passLabel = new JLabel("Password:");  
    userText = new JTextField(15);  
    passText = new JPasswordField(15);  
    passText.setEchoChar('*');//đặt ký tự hiển thị cho ô nhập mật khẩu  
    JButton button = new JButton("Sign in");  
    button.addActionListener(this);//đăng ký lắng nghe sự kiện click  
    frame.add(userLabel);  
    frame.add(userText);  
    frame.add(passLabel);  
    frame.add(passText);  
    frame.add(button);  
    frame.setLocation(200, 150);  
    frame.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
    frame.setResizable(false);  
    frame.setVisible(true);  
}  
  
public void actionPerformed(ActionEvent e) {//tái định nghĩa hàm xử lý sự kiện  
    if(e.getActionCommand().equals("Sign in")){  
        String message = "Bạn nhập vào username là "+userText.getText()+"  
        " và password là "+String.valueOf(passText.getPassword());  
        JOptionPane.showMessageDialog(frame, message);  
    }  
}
```

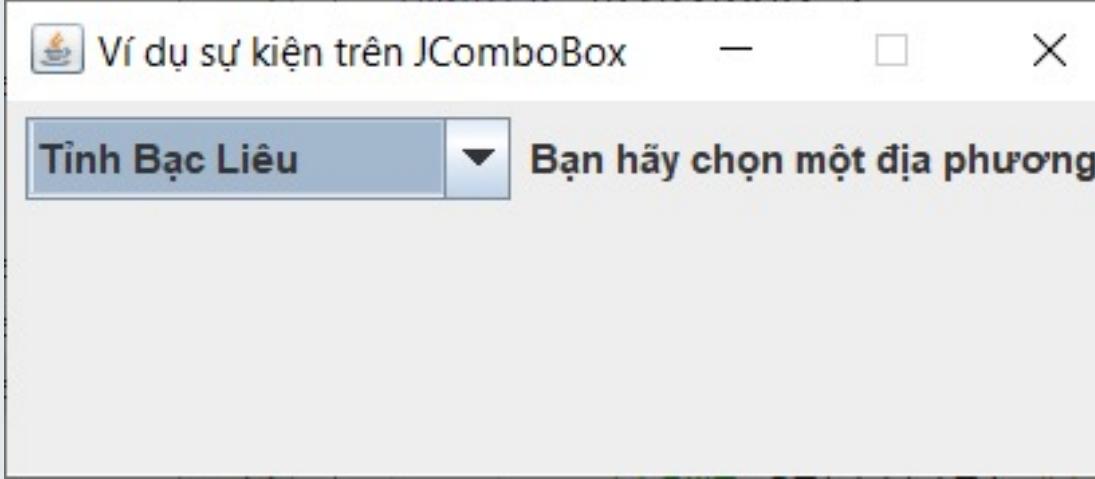
Ví dụ xử lý sự kiện



- ❑ Sử dụng sự kiện phù hợp để không cho phép người dùng nhập vào textfield ký tự khác với ký tự số.
- ❑ Hướng dẫn
 - Tạo một frame
 - Thiết lập kiểu bố cục FlowLayout
 - Tạo textfield
 - Đăng ký lắng nghe sự kiện bàn phím cho textfield
 - Thêm textfield vào frame
 - Tái định nghĩa hàm xử lý sự kiện keyPressed

- ❑ Thiết kế một Combobox bao gồm các phần tử được định nghĩa trước.
- ❑ Hãy sử dụng sự kiện phù hợp để hiển thị phần tử hiện được chọn của ComboBox bằng một Label mỗi khi người dùng thay đổi phần tử hiện hành của ComboBox.
- ❑ Hướng dẫn
 - Tạo một frame
 - Thiết lập kiểu bố cục FlowLayout
 - Tạo combobox, label
 - Đăng ký lắng nghe sự kiện phần tử được chọn cho combobox
 - Thêm combobox, label vào frame
 - Tái định nghĩa hàm xử lý sự kiện itemStateChanged để đặt lại nhãn của label

Ví dụ xử lý sự kiện



- Tạo 1 trình đơn sử dụng đối tượng Jmenubar.**
 - Sử dụng phương thức khởi tạo `JMenuBar()`.
- Thêm các đối tượng JMenu vào JMenubar.**
 - Sử dụng phương thức khởi tạo `JMenu(String menuName)`.
- Thêm các đối tượng JMenuItem vào Jmenu.**
 - Sử dụng phương thức khởi tạo `JMenuItem(String menuItemName)`.
 - Sử dụng phương thức `setActionCommand(String menuItemName)` để xác định mục menu nào được nhấn.
 - Có thể thiết lập phím tắt bằng phương thức `setMnemonic(char key)`.

Tạo trình đơn (menu)

❑ Gắn JMenuBar vào JFrame.

- Mặc định các JMenuItem là enabled.
- Có thể vô hiệu hóa JMenuItem bằng cách gọi phương thức `setEnabled(boolean b)`

Ví dụ tạo trình đơn

```
private void prepareGUI() {
    mainFrame = new JFrame("Ví dụ Menu");
    mainFrame.setSize(400, 400);
    mainFrame.setLayout(new GridLayout(3, 1));

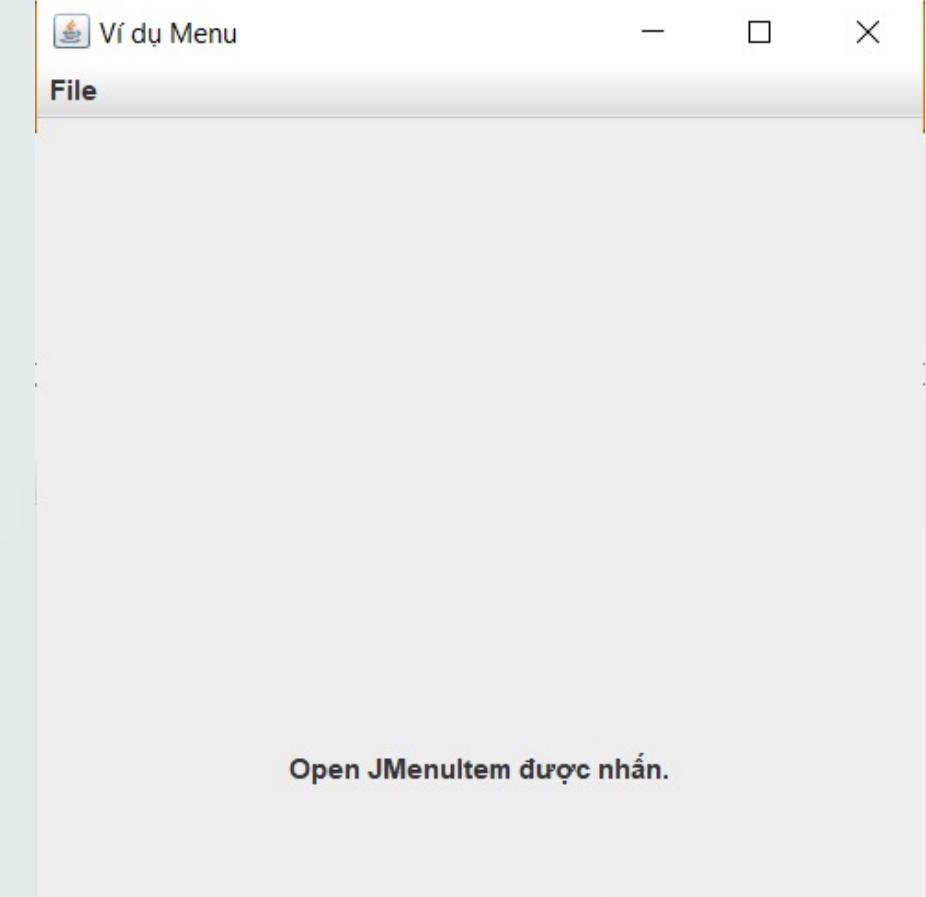
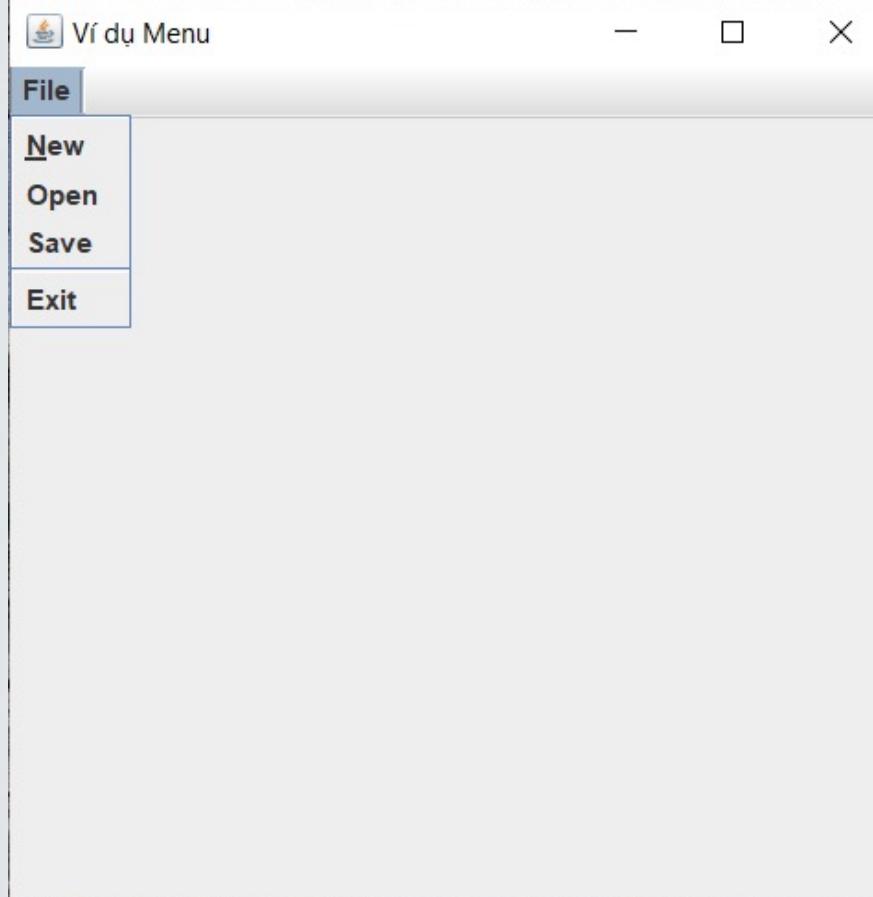
    headerLabel = new JLabel("", JLabel.CENTER);
    statusLabel = new JLabel("", JLabel.CENTER);

    statusLabel.setSize(350, 100);
    //sử dụng sự kiện để thoát hệ thống khi frame được đóng
    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 showMenuDemo() {
    //Tạo menu bar
    final JMenuBar menuBar = new JMenuBar();
    //Tạo các menu
    JMenu fileMenu = new JMenu("File");
    //Tạo các menu 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");
    //đăng ký lắng nghe sự kiện click cho các menu item
    MenuItemListener menuItemListener = new MenuItemListener();
    newItem.addActionListener(menuItemListener);
    openMenuItem.addActionListener(menuItemListener);
    saveMenuItem.addActionListener(menuItemListener);
    exitMenuItem.addActionListener(menuItemListener);
    //Thêm các menu item vào menu
    fileMenu.add(newItem);
    fileMenu.add(openMenuItem);
    fileMenu.add(saveMenuItem);
    fileMenu.addSeparator();
    fileMenu.add(exitMenuItem);
    //Thêm menu vào menu bar
    menuBar.add(fileMenu);
    //Thêm menubar vào frame
    mainFrame.setJMenuBar(menuBar);
    mainFrame.setVisible(true);
}
```

Ví dụ tạo trình đơn



❑ Sử dụng lớp JToolbar.

- JToolBar là một thanh công cụ dài gồm các nút với biểu tượng tương ứng thường nằm dưới menu bar.
- Thành phần GUI thường được thêm vào thanh công cụ là:
 - Button
 - Textfield
- Thường gắn Toolbar vào cạnh trên (trang đầu) của BorderLayout.
- Có thể thêm vào ngăn cách bằng cách gọi phương thức addSeparator().

Ví dụ tạo thanh công cụ

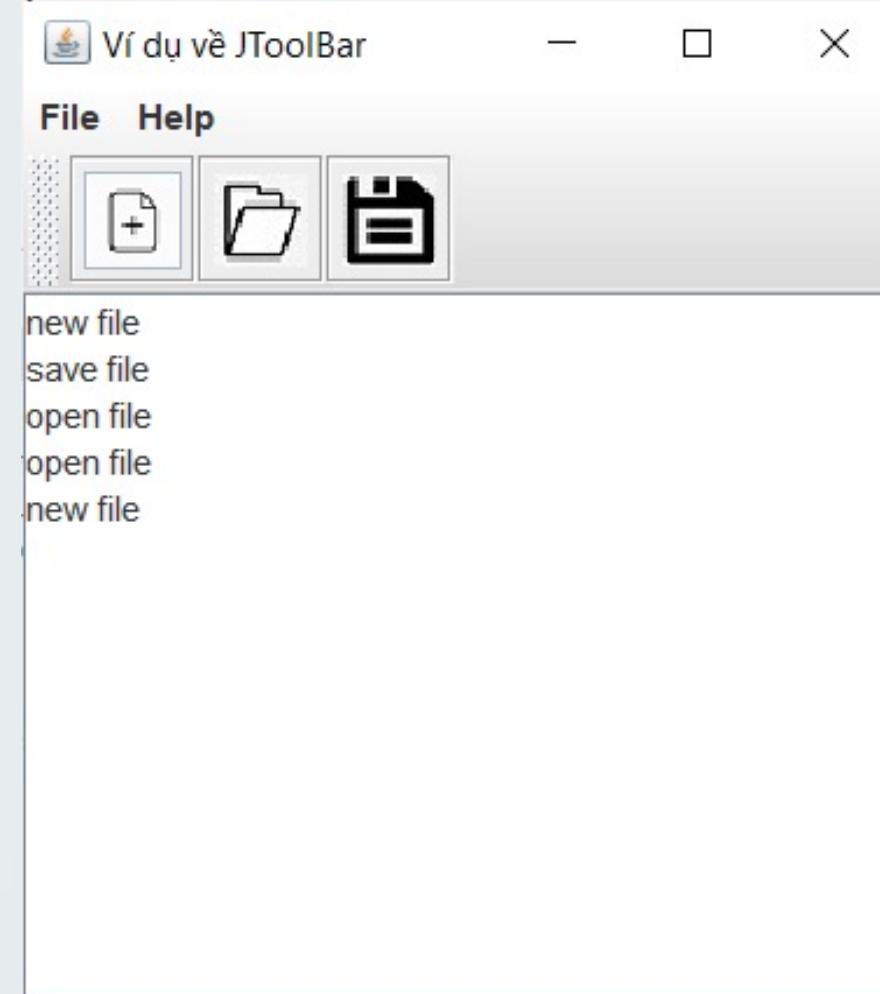
```
//Tạo tool bar
private JToolBar createToolBar() {
    JToolBar tb = new JToolBar();
    tb.add(createButton(iconNew, "New", NEW, "Create a new file"));
    tb.add(createButton(iconOpen, "Open", OPEN, "Open a file"));
    tb.add(createButton(iconSave, "Save", SAVE, "Save this file"));
    return tb;
}

//Tạo các nút lệnh với icon
private JButton createButton(String iconLink, String text, String command, String toolTip) {
    JButton btn = new JButton();
    btn.setActionCommand(command);
    btn.setToolTipText(toolTip);

    ImageIcon icon = createIcon(iconLink);
    if (icon == null) {
        btn.setText(text);
    } else {
        btn.setIcon(icon);
    }
    btn.setActionCommand(command);
    btn.addActionListener(this);
    return btn;
}

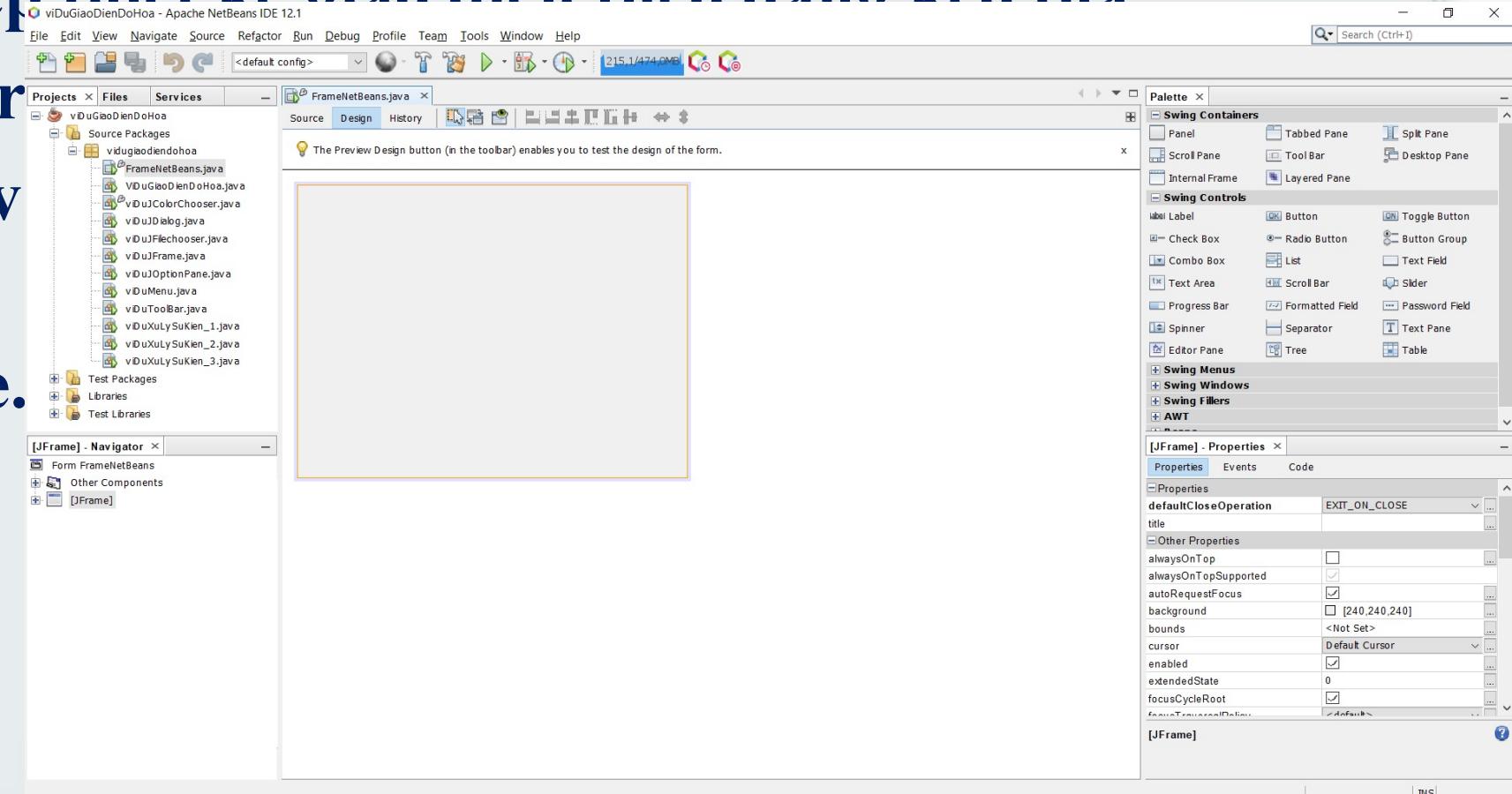
//Tạo các icon cho nút lệnh
private ImageIcon createIcon(String iconLink) {
    if (iconLink != null) {
        return new ImageIcon(iconLink);
    } else {
        System.out.println("image " + iconLink + " not found");
        return null;
    }
}
```

Ví dụ tạo thanh công cụ



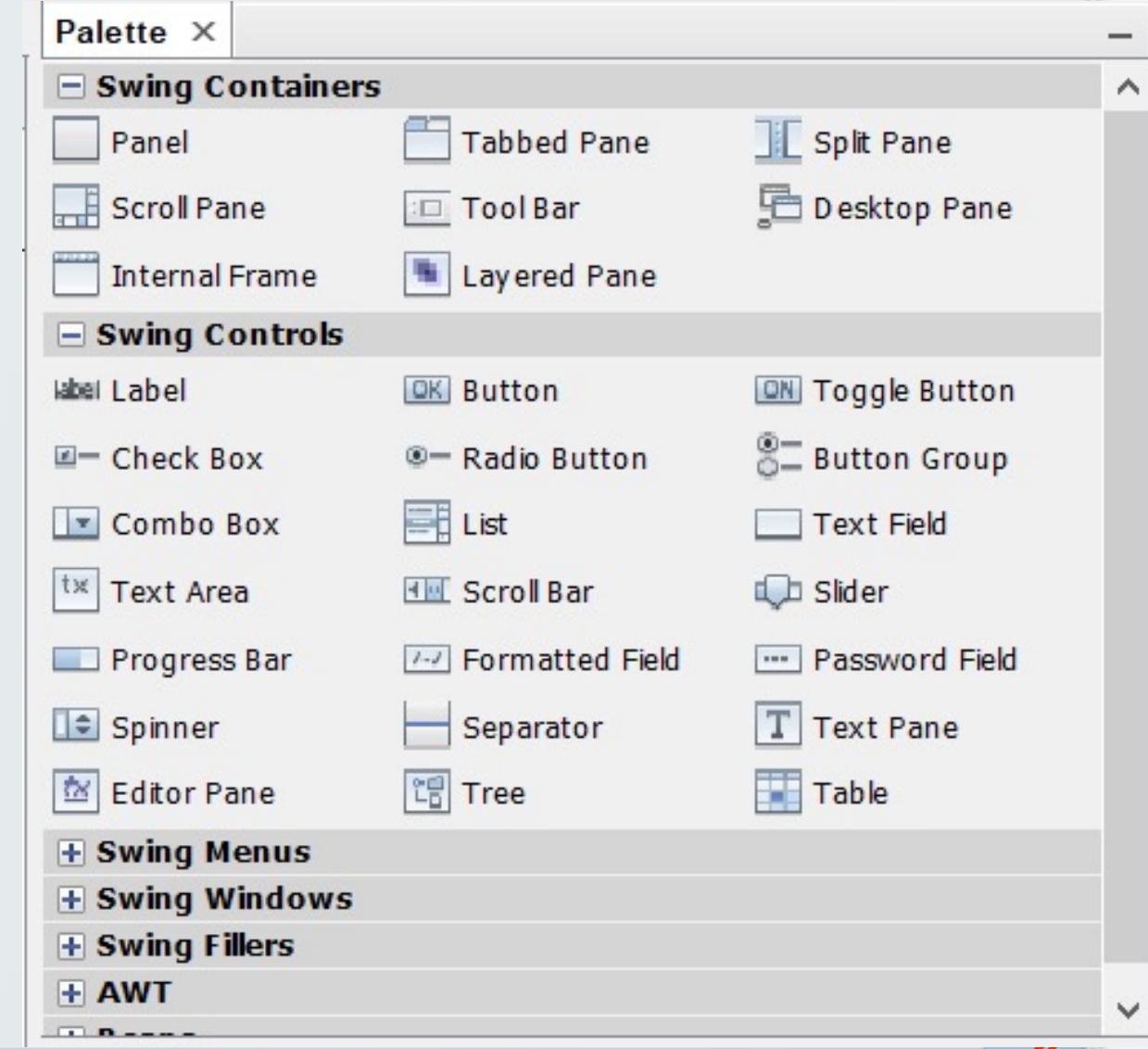
Các bước thực hiện để tạo một Frame

- ❑ NetBeans cho phép **thiết kế giao diện theo dạng kéo thả**
- ❑ Tạo một project
- ❑ Nhấn chuột phải vào form
- ❑ Đặt tên cho frame.



Tìm hiểu giao diện thiết kế của NetBeans

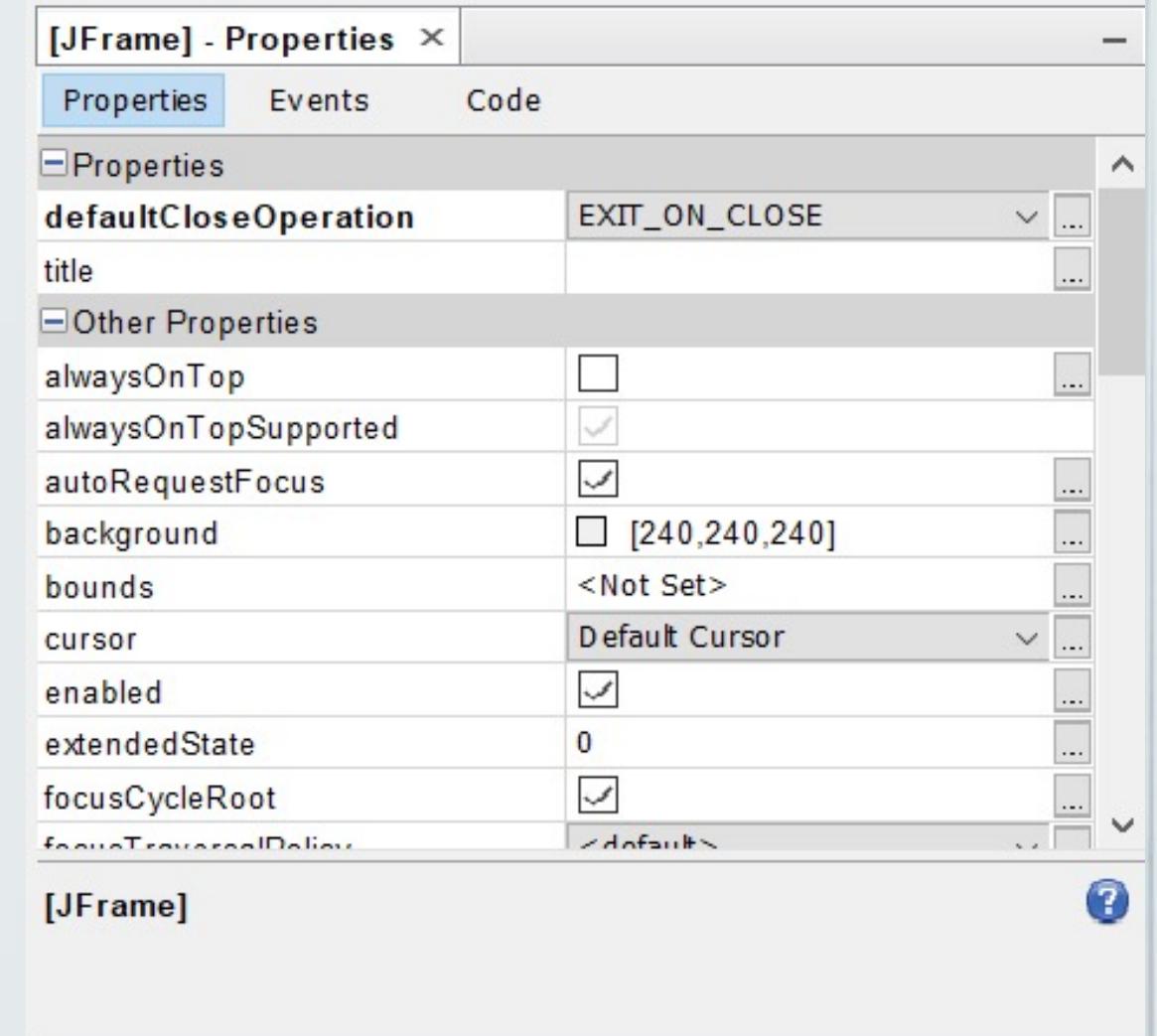
- ❑ Palette là hộp công cụ “chứa đựng” tất cả các thành phần GUI và được tổ chức theo nhóm.
- ❑ Cần dùng thành phần nào thì chúng ta chọn và kéo thả vào khu vực thiết kế (design).
- ❑ Các thành phần vật chứa (Containers) được sử dụng để có thể thiết kế các bộ cục khác nhau.



❑ Cửa sổ Properties cho phép người lập trình thiết lập nhanh các “đặc tính” của thành phần GUI hiện hành.

❑ Các đặc tính này bao gồm:

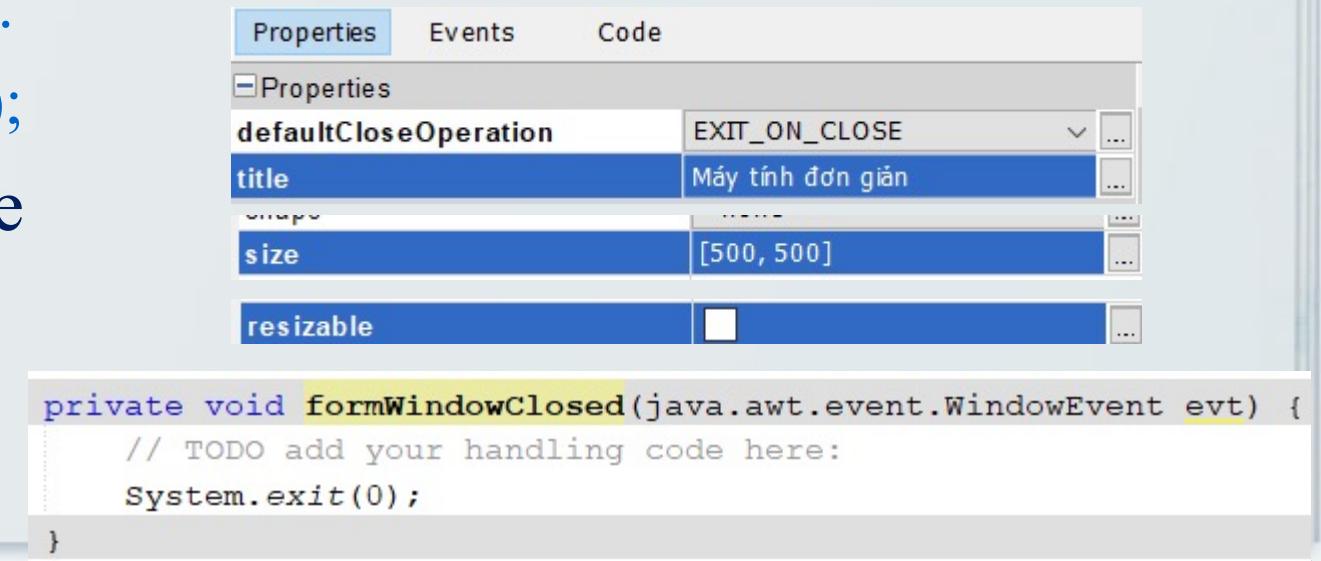
- Thuộc tính (Properties)
- Sự kiện (Events)
- Và mã nguồn tương ứng (sinh tự động).



Ví dụ thiết kế Máy tính đơn giản

Bước 1

- Đặt tiêu đề cho JFrame là “Máy tính đơn giản”.
- Thiết lập kích thước cho Jframe là 500, 500.
- Thiết lập cho phép thay đổi kích thước của JFrame.
- Thiết lập chế độ thoát hệ thống khi người dùng đóng Jframe.
 - Chọn sự kiện WindowClosed.
 - Nhập câu lệnh System.exit(0);
- Đổi icon mặc định của Jframe
 - Chuẩn bị một icon.
 - Thay đổi icon mặc định.



Ví dụ thiết kế Máy tính đơn giản

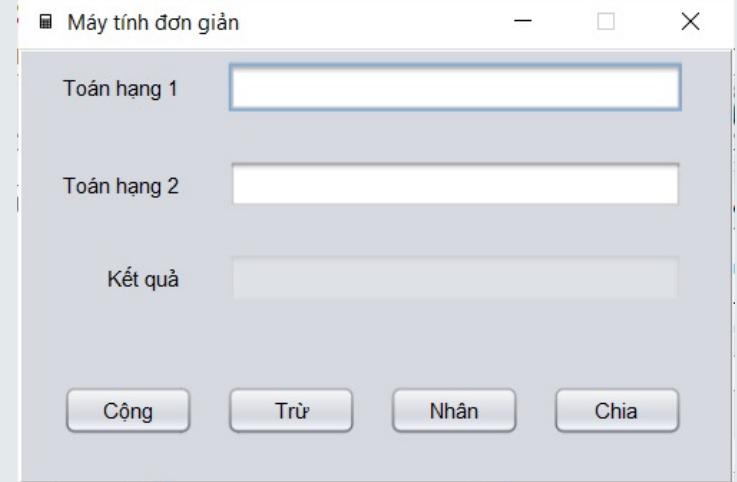


Ví dụ thiết kế Máy tính đơn giản



Bước 2

- Đưa vào JFrame một JPanel để tạo viền cho máy tính.
 - Thiết lập border cho JPanel.
- Thiết kế nhãn
 - Toán hạng 1
 - Toán hạng 2
 - Kết quả
- Thiết kế ô nhập liệu
 - JTextField1 cho toán hạng 1 và JTextField2 cho toán hạng 2.
 - JTextField3 cho kết quả, không cho phép sửa nội dung.
- Thiết kế các nút lệnh với nhãn
- Cộng
- Trừ
- Nhân

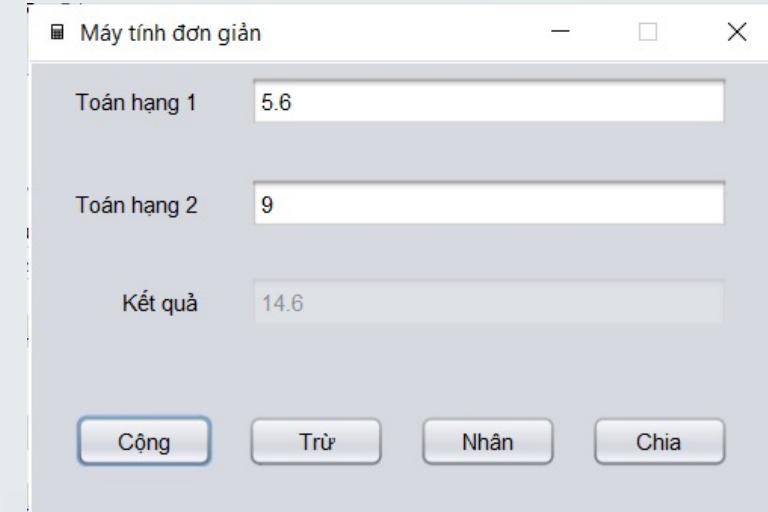


Ví dụ thiết kế Máy tính đơn giản

Bước 3

- Nhấn vào từng nút lệnh để viết mã cho sự kiện click tương ứng.
- Chú ý lệnh cho phép chia nên kiểm tra trường hợp toán hạng 2 có giá trị bằng 0.

```
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    double d1 = Double.parseDouble(this.jTextField1.getText());  
    double d2 = Double.parseDouble(this.jTextField2.getText());  
    if(d2 == 0){  
        this.jTextField3.setText("Không thể chia cho 0");  
    }  
    else{  
        double kq = d1 / d2;  
        this.jTextField3.setText(String.valueOf(kq));  
    }  
}
```



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    double d1 = Double.parseDouble(this.jTextField1.getText());  
    double d2 = Double.parseDouble(this.jTextField2.getText());  
    double kq = d1 + d2;  
    this.jTextField3.setText(String.valueOf(kq));  
}
```



Stop!

