

## Chương 6

## Giao diện trong Java với AWT - SWING

# NỘI DUNG

- **Giới thiệu thiết kế GUI trong java**
- **Đối tượng khung chứa (Container)**
- **Các thành phần cơ bản (Component)**
- **Bộ quản lý trình bày (Layout Manager)**
- **Lập trình xử lý sự kiện**

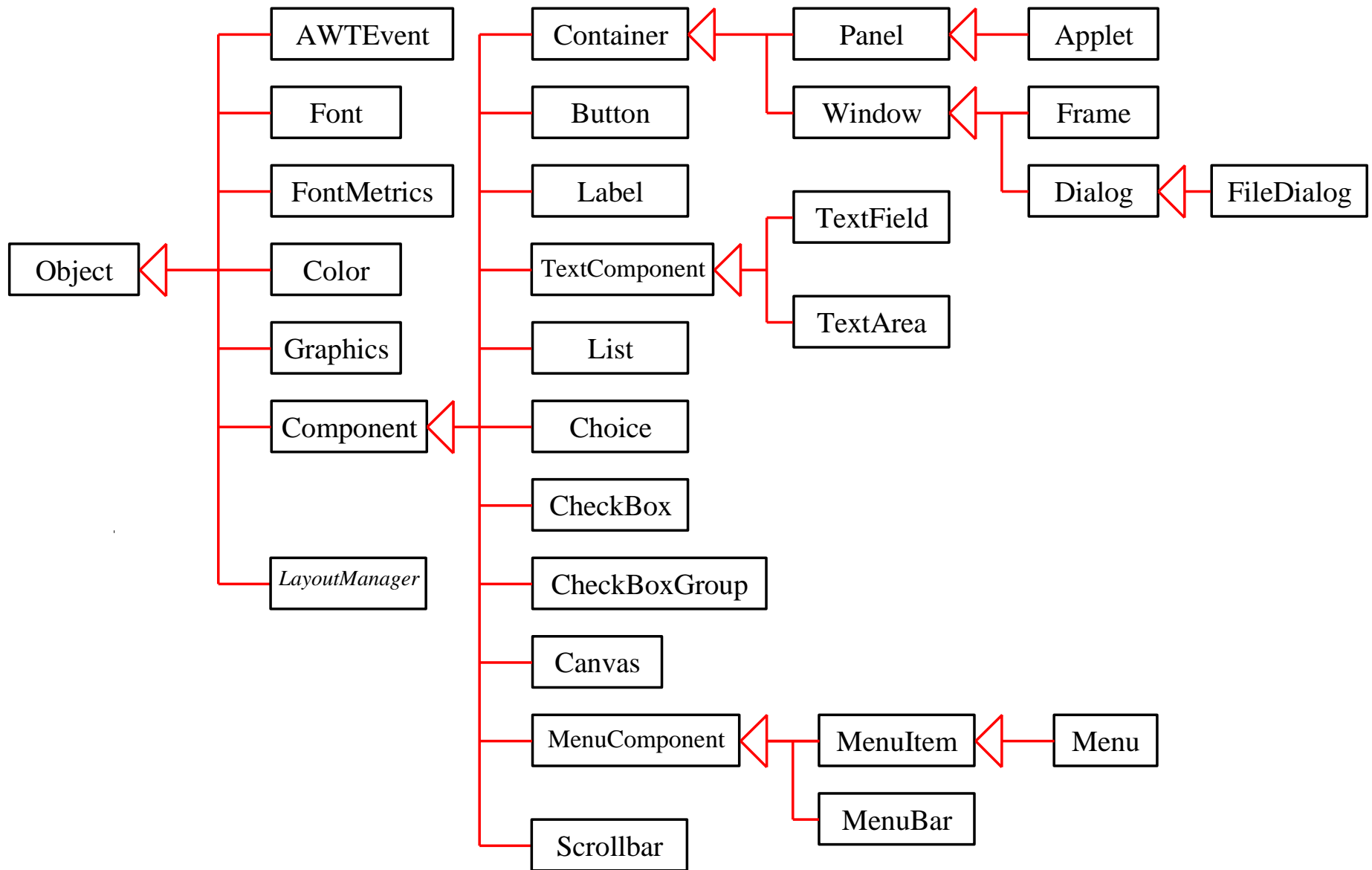
# GIỚI THIỆU VỀ THIẾT KẾ GUI

- Thư viện hỗ trợ: tập hợp các lớp java cung cấp hỗ trợ thiết kế, xây dựng GUI (Graphic User Interface) là:
  - `awt (java.awt.*)`
  - `swing (javax.swing.*)`

# GIỚI THIỆU AWT

- AWT viết tắt của **Abstract Windowing Toolkit**
- AWT là tập hợp các lớp Java cho phép chúng ta tạo một GUI.
- Cung cấp các mục khác nhau để tạo hoạt động và hiệu ứng GUI
  - `import java.awt.*;`
  - `import java.awt.event.*;`

# GIỚI THIỆU AWT

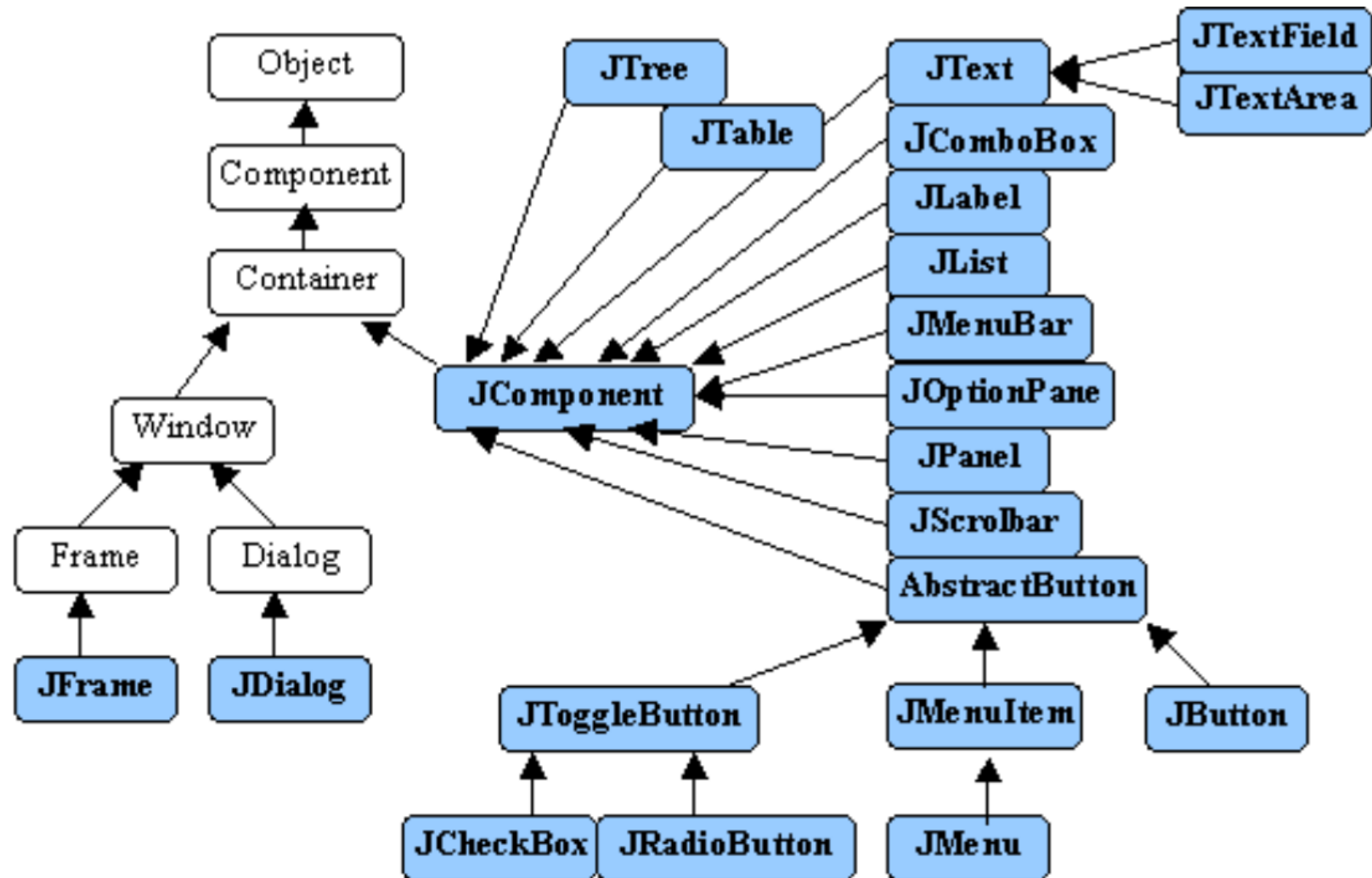


# Java Swing

- Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng window-based. Nó được xây dựng trên API AWT (Abstract Windowing Toolkit) và được viết hoàn toàn bằng Java.
- Không giống như AWT, Java Swing cung cấp các thành phần không phụ thuộc vào nền tảng và nhẹ hơn.
- Gói **javax.swing** cung cấp các lớp cho java swing API như JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser

# Java swing – Phân cấp lớp

## Kiến trúc SWING



# NGUYÊN TẮC XÂY DỰNG GUI

- Lựa chọn một container: Frame, Window, Dialog, Applet,...
- Tạo các control: (buttons, text areas, list, choice, checkbox,...)
- Đưa các control vào vùng chứa
- Sắp xếp các control trong vùng chứa (Layout).
- Thêm các xử lý sự kiện (Listeners)



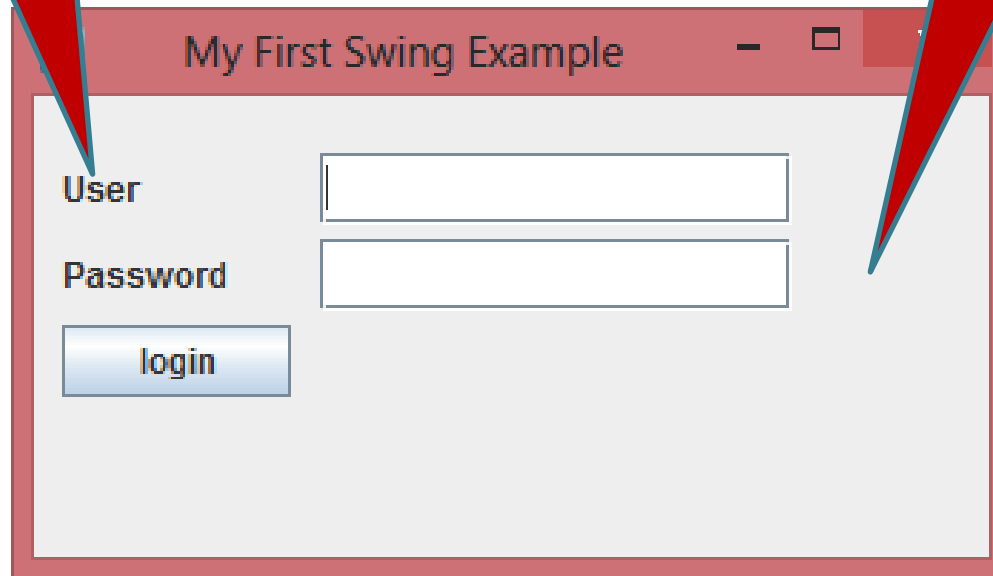
# Đưa component vào GUI

- 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.

# Khái niệm

component

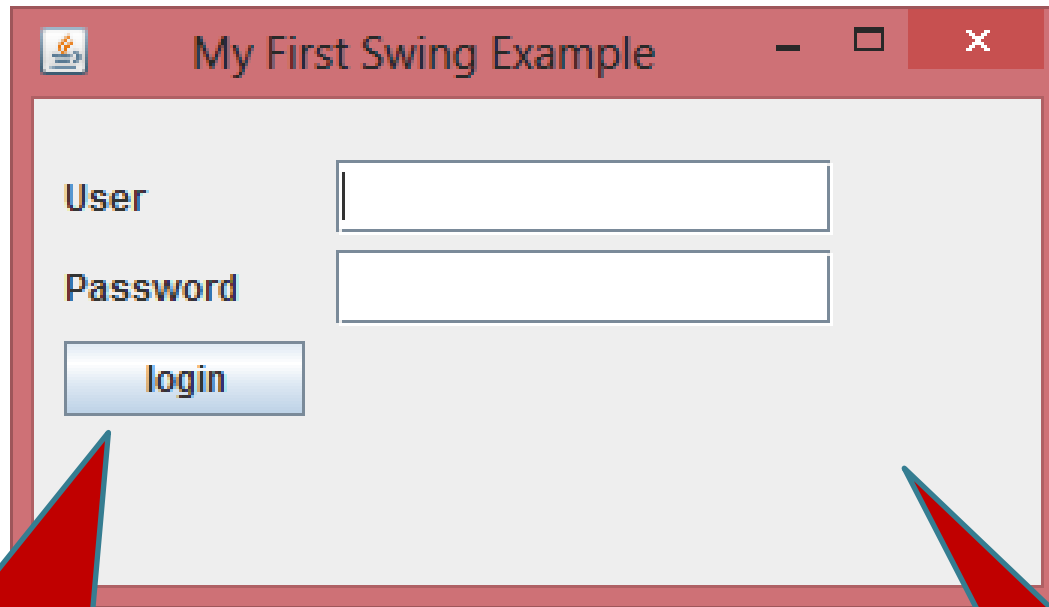
container



GUI = Containers + Components

# Đưa component vào GUI

- Ví dụ

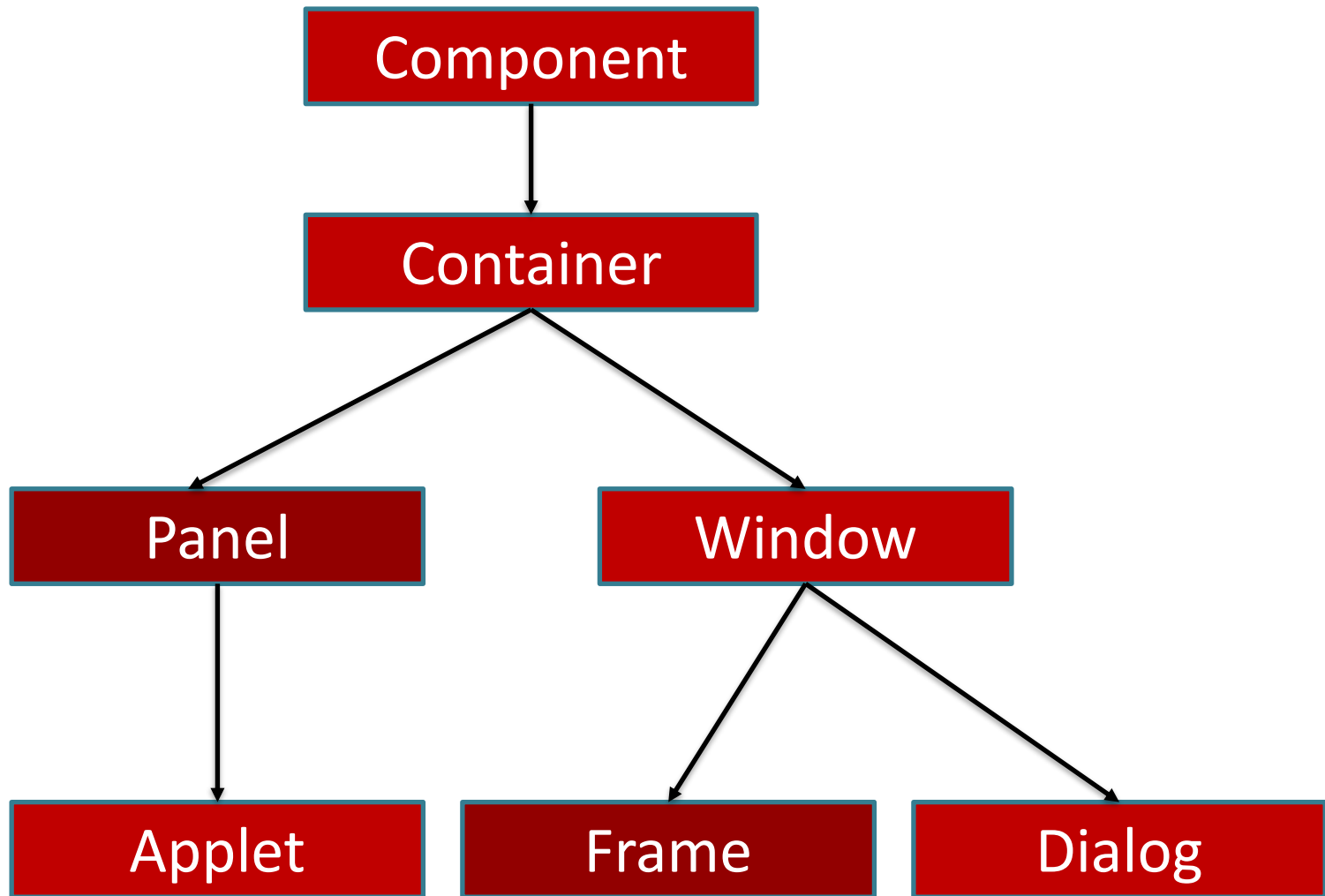


2 label  
2 text-field  
1 button

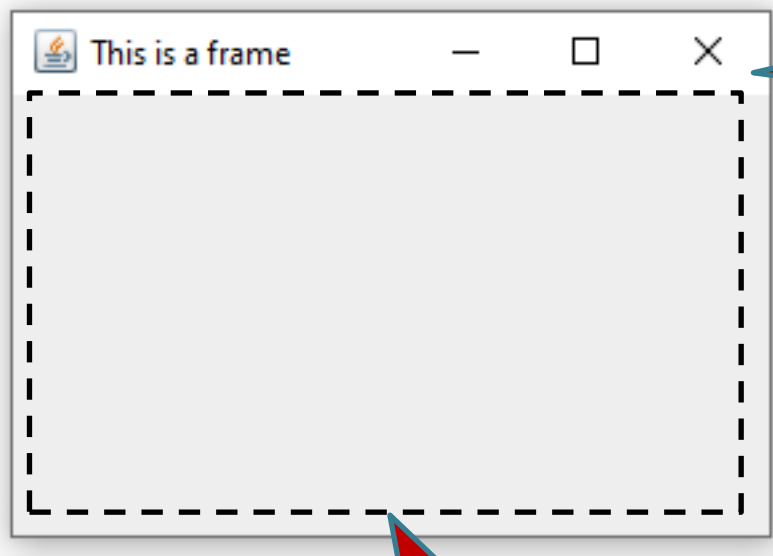
container

# **ĐỐI TƯỢNG KHUNG CHỨA (CONTAINERS)**

# Phân cấp thừa kế



# Phân cấp thừa kế



frame

Frame là khung chữ nhật  
có đường viền và có nút  
điều khiển cửa sổ

Panel

Panel không có đường viền

# Khái niệm

- **Container**: Đối tượng chứa các element, cho phép vẽ, tô màu lên container.
- **Frame** và **Panel** là các class thường dùng.
- Panel thường dùng để chứa các element trong 1 GUI phức tạp, 1 Frame có thể chứa nhiều Panel.
- **Applet** thường dùng để tạo 1 ứng dụng nhúng vào Browser.

# Frame

- **Frame** được dùng để xây dựng các ứng dụng GUI chạy độc lập.
- **Frame** là một cửa sổ có thanh tiêu đề và các đường biên. Bố cục mặc định của Frame là BorderLayout.
- **Frame** kế thừa từ Window, nó có thể nghe các sự kiện xảy ra trên cửa sổ khi cài đặt giao tiếp **WindowListener**.
- Các ứng dụng độc lập thường tạo ra cửa sổ kế thừa từ lớp Frame.
- Chú ý:
  - Frame không có các phương thức init, start... như trong Applet.
  - Các ứng dụng độc lập dùng Frame phải có hàm main và được chạy trực tiếp bằng lệnh java.
  - Cần có lệnh setSize, setVisible(true) để có thể hiển thị Frame.



# Frame

Có hai cách để tạo khung (Frame):

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

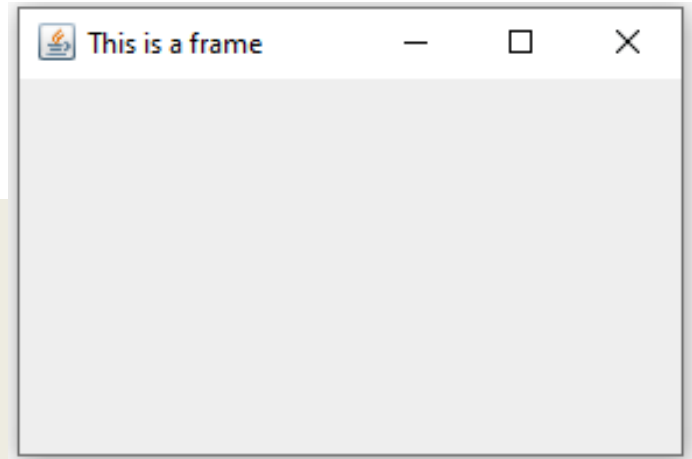
# Ví dụ

- Tạo frame:

```
package container;

import javax.swing.JFrame;

public class ContainerDemo {
    public static void main(String[] args) {
        JFrame jframe1 = new JFrame();
        jframe1.setSize(300, 200);
        jframe1.setTitle("This is a frame");
        jframe1.setVisible(true);
    }
}
```



# Ví dụ

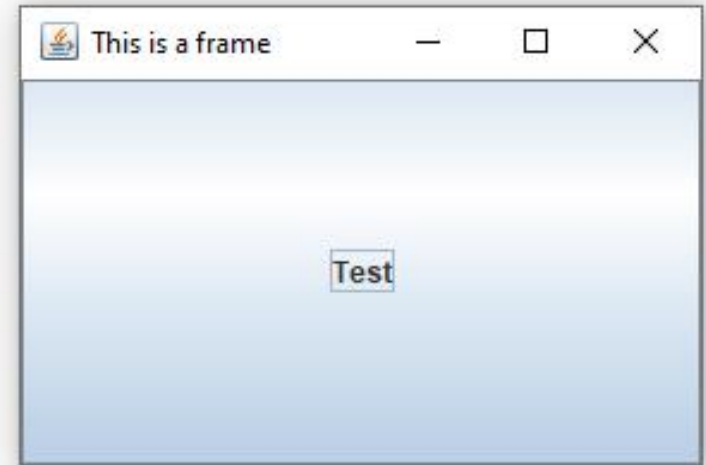
- Tạo frame và thêm nút:

```
package container;

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

public class ContainerDemo {
    public static void main(String[] args) {
        JFrame jframe1 = new JFrame();
        JButton jbutton1 = new JButton("Test");
        jframe1.add(jbutton1); // thêm button vào JFrame

        jframe1.setSize(300, 200);
        jframe1.setTitle("This is a frame");
        jframe1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jframe1.setVisible(true);
        jframe1.setLocationRelativeTo(null); // thiết lập JFrame giữa
        màn hình
    }
}
```



# Ví dụ

- Tạo frame và thêm nút:

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

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

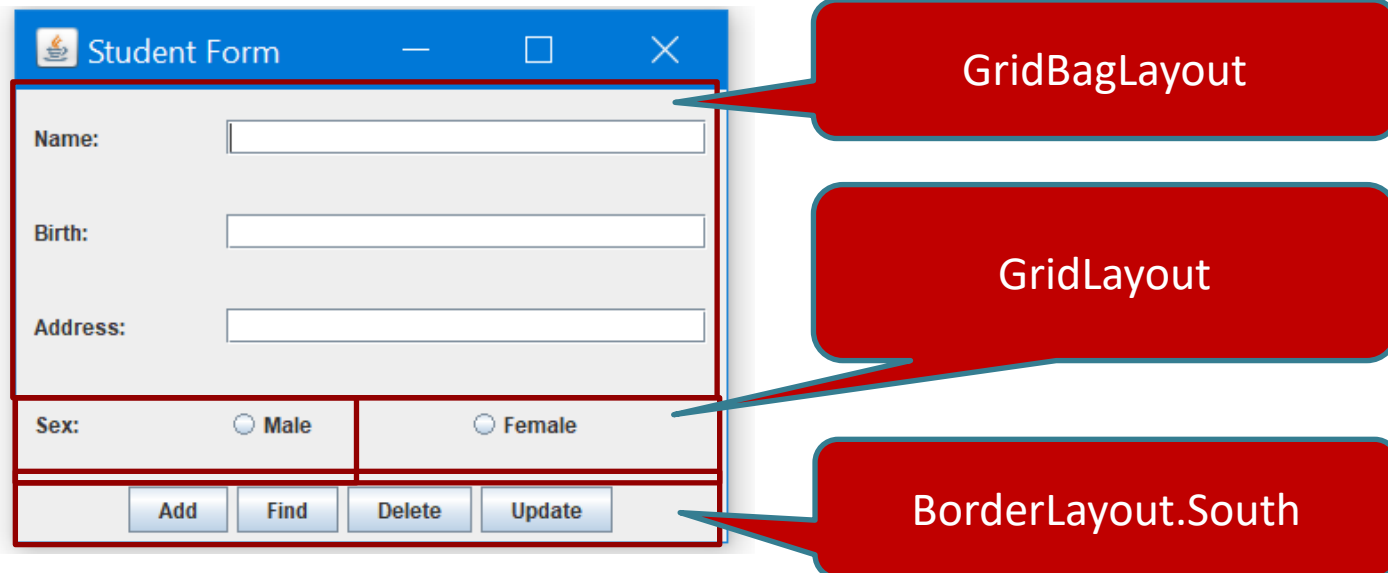
    public ContainerDemo() {
        JButton btt = new JButton("Submit") ; // tạo button
        b.setBounds(130, 50, 100, 40); // trục x , y , width, height

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

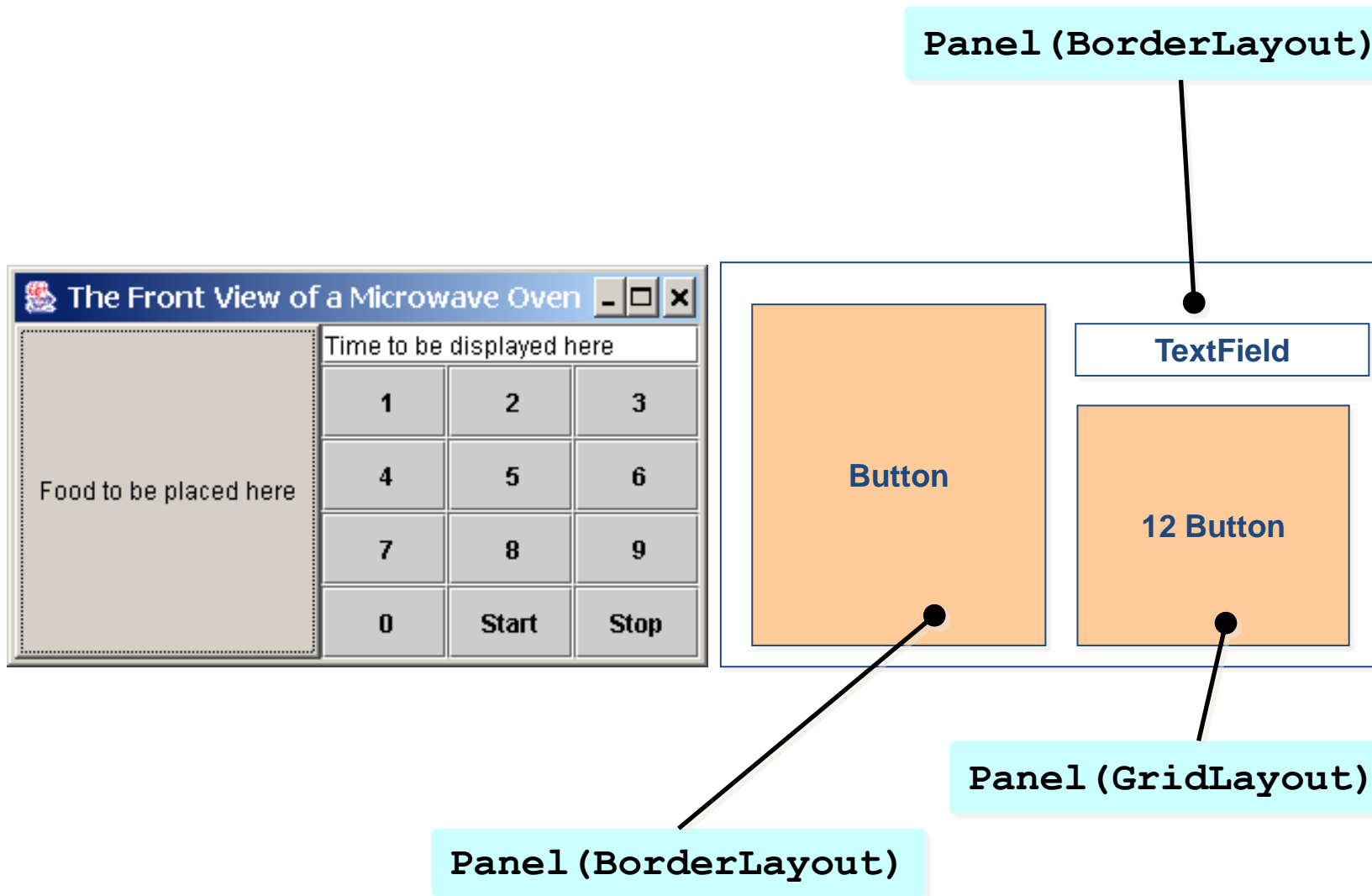
    public static void main(String[] args) {
        new ContainerDemo();
    }
}
```

# PANEL

- Lớp Panel kế thừa từ Container. Nó có thể được dùng để tạo ra các giao diện theo ý muốn.
- Ví dụ: Một giao diện có thể có nhiều panel sắp xếp theo một layout nhất định, mỗi panel lại có các component sắp xếp theo một layout riêng.
- Chú ý: Panel có bố cục mặc định là FlowLayout.



# LỚP PANEL (VÙNG CHỨA)



# **ĐỐI TƯỢNG QUẢN LÝ TRÌNH BÀY (LAYOUT)**

# Khái niệm

- Layout : Cách bố trí các components lên container.
- Không dễ dàng gì để tự quản lý vị trí của các components trên GUI  
-> LayoutManager là interface mô tả về các layout.
- Java cung cấp sẵn một số layout, các lớp layout này đều implement LayoutManager interface.

Có năm bộ quản lý trình bày:

- Flow Layout
- Border Layout
- Grid Layout
- Gridbag Layout
- Null Layout



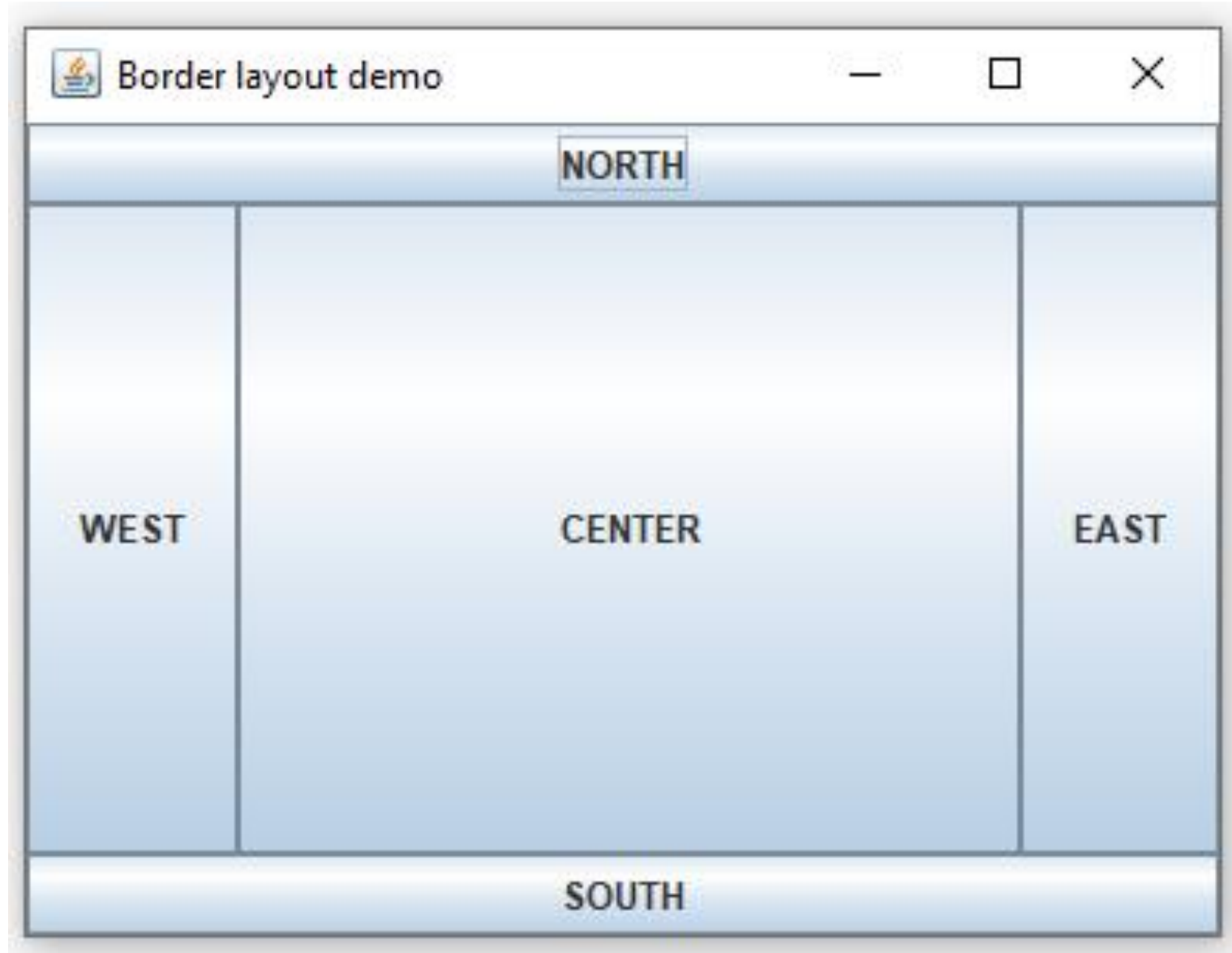
# BorderLayout

- Bố trí các component theo dạng ra biên của khung tạo ra 5 vị trí: EAST, WEST, SOUTH, NORTH, CENTER.
- Đây là layout MẶC ĐỊNH của Frame.
- Nếu container chỉ có 1 component và đặt nó ở vị trí CENTER thì component này phủ kín container.
- Cú pháp thêm 1 component vào container tại 1 vị trí:

```
Container.add("East", component);  
Container.add(BorderLayout.EAST, component);
```

- Tương tự cho “West”, “South”, “North”, “Center”

# BorderLayout



# BorderLayout

```
package container;
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;

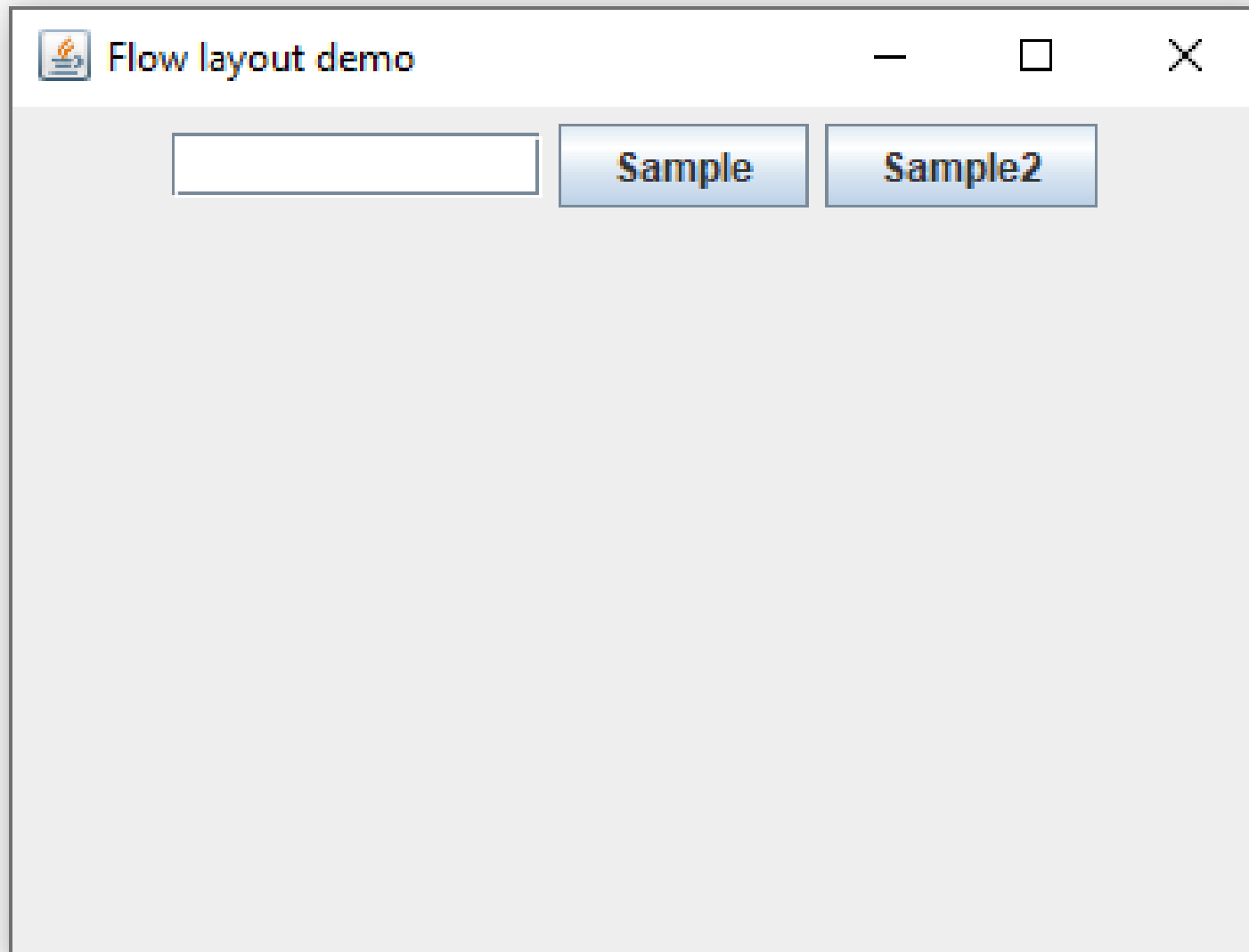
public class BorderLayoutDemoFrame extends JFrame{
    public BorderLayoutDemoFrame() {
        this.getContentPane().add(BorderLayout.NORTH, new JButton("NORTH"));
        this.getContentPane().add(BorderLayout.SOUTH, new JButton("SOUTH"));
        this.getContentPane().add(BorderLayout.CENTER, new
        JButton("CENTER"));
        this.getContentPane().add(BorderLayout.WEST, new JButton("WEST"));
        this.getContentPane().add(BorderLayout.EAST, new JButton("EAST"));
    }

    public static void main(String[] args) {
        BorderLayoutDemoFrame frame = new BorderLayoutDemoFrame();
        frame.setSize(400, 300);
        frame.setTitle("Border layout demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        jframe1.setLocationRelativeTo(null);
    }
}
```

# FlowLayout

- Bố trí các component theo dạng chảy xuôi theo thứ tự mà phần tử này được add vào container.
- Đây là layout mặc định của Panel.
- Nếu có nhiều component trên container -> Các component có thể được đặt trên nhiều dòng -> Vấn đề giống hàng (Align)
- Giữa các component, chúng hở nhau theo chiều dọc (vgap) bao nhiêu, theo chiều ngang (hgap) bao nhiêu?

# FlowLayout



# FlowLayout

```
package container;

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

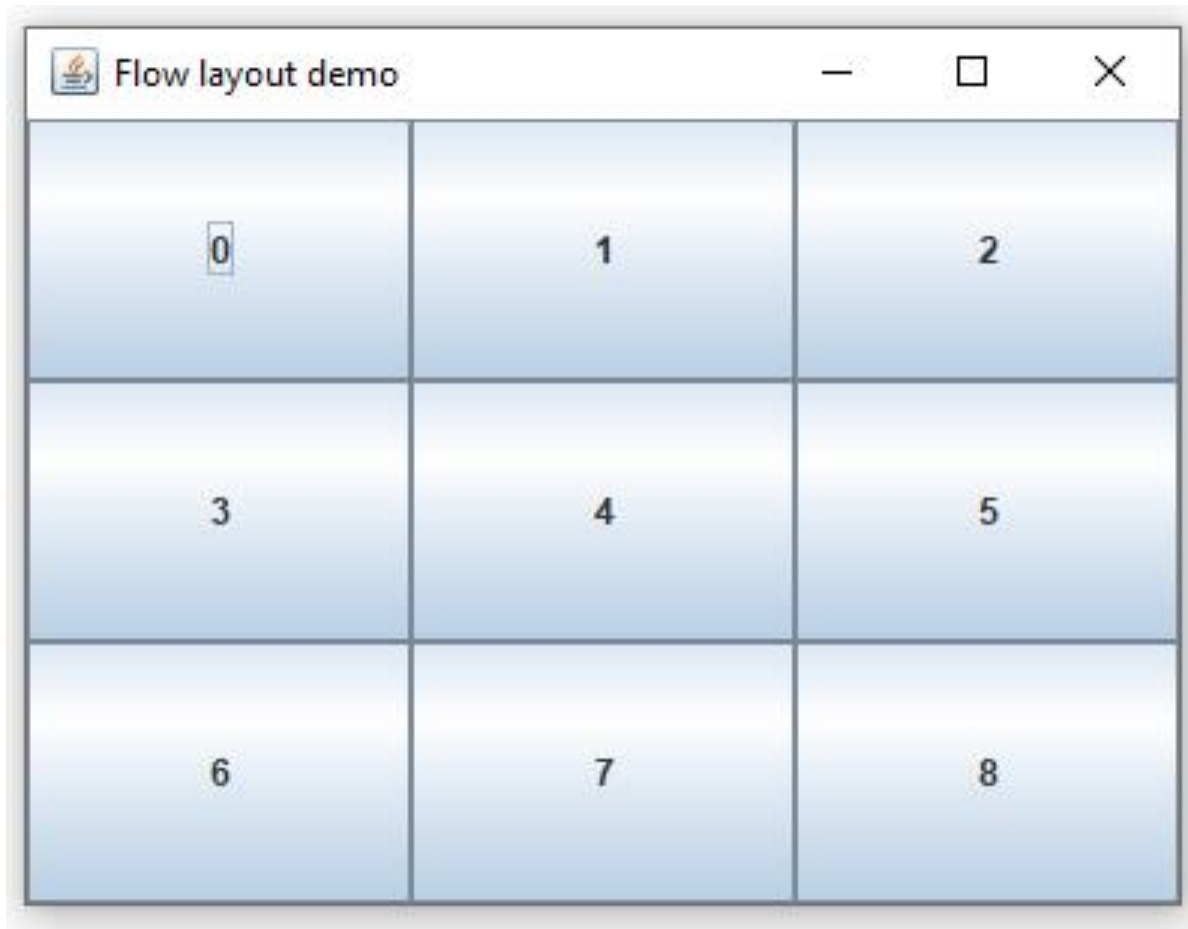
public class FlowLayoutDemo extends JFrame{
    public FlowLayoutDemo() {
        JPanel panel1 = new JPanel();
        panel1.add(new JTextField(10));
        panel1.add(new JButton("Add"));
        panel1.add(new JButton("Edit"));
        getContentPane().add(panel1);
    }

    public static void main(String[] args) {
        FlowLayoutDemo frame = new FlowLayoutDemo();

        frame.setSize(400, 300);
        frame.setTitle("Flow layout demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

# GridLayout

- Bố trí các component thành 1 lưới rows dòng, cols cột đều nhau.



# GridLayout

```
package container;

import java.awt.GridLayout;

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

public class GridLayoutDemo extends JFrame {
    public GridLayoutDemo() {
        JPanel panel1 = new JPanel();
        panel1.setLayout(new GridLayout(3, 3));

        for (int i = 0; i < 9; i++) {
            panel1.add(new JButton(String.valueOf(i)));
        }
        getContentPane().add(panel1);
    }

    public static void main(String[] args) {
        GridLayoutDemo frame = new GridLayoutDemo();

        frame.setSize(400, 300);
        frame.setTitle("Flow layout demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



# GridBagLayout

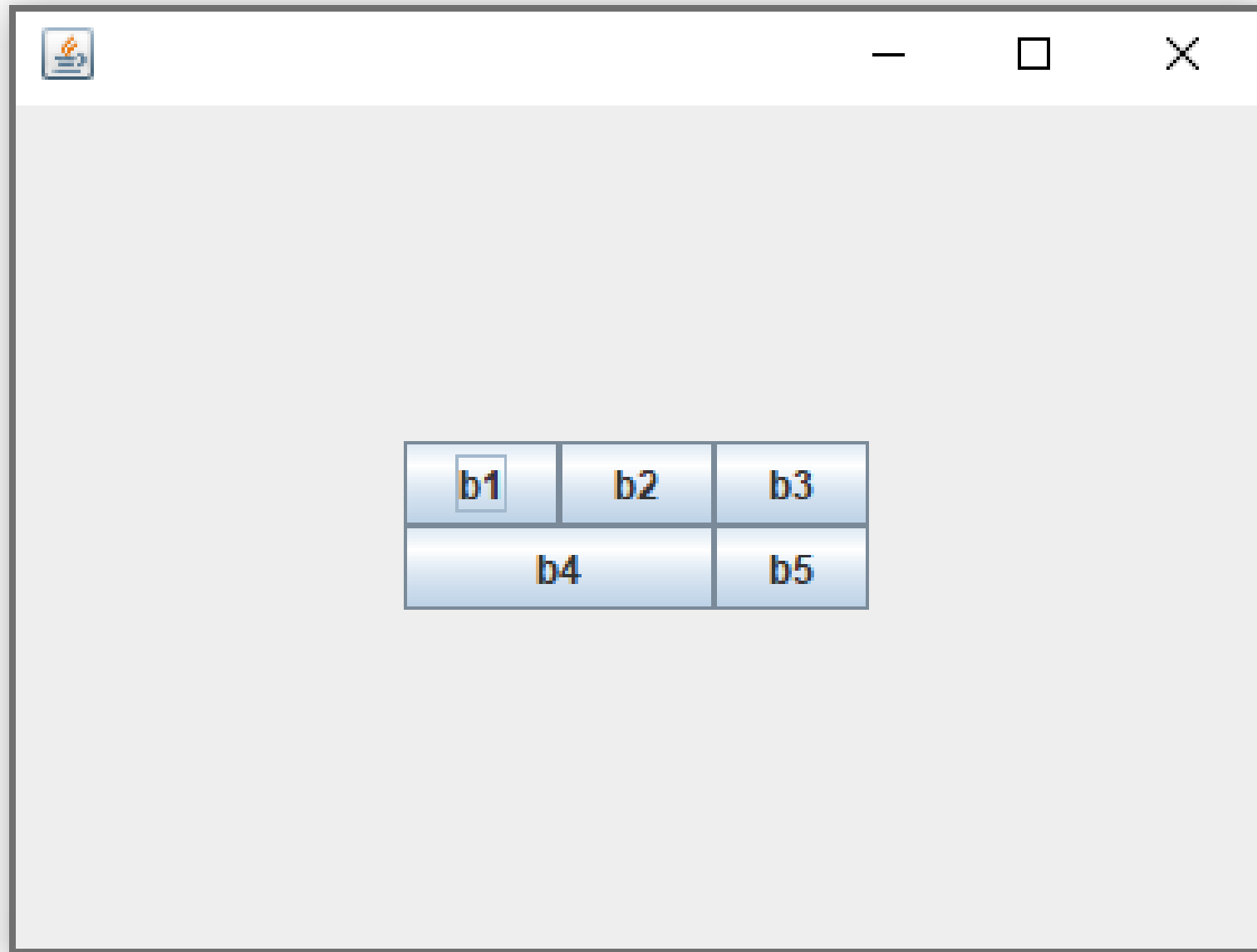
- Layout dạng lưới cho phép 1 component chiếm 1 số ô kề nhau theo cả 2 chiều.
- Cách để đưa component vào GridBagLayout:
  - 1 component vào 1 vị trí nhưng trải dài trên nhiều ô kề nhau là 1 sự “ràng buộc” 1 component vào các ô này.
  - Một đối tượng thuộc lớp GridBagConstraints sẽ đảm nhiệm việc này.

# GridBagLayout - GridBagConstraints

**Properties** – Đa số là static data

- **gridx, gridy** : ô sẽ đặt component vào
- **gridwidth, gridheight** : số ô sẽ phủ theo 2 chiều khi thêm 1 component vào ô <gridx,gridy>
- **weightx, weighty**: Khoảng hở của lưới, mặc định là 0.
- **anchor**: Vị trí đặt component, mặc định là CENTER, các hằng khai báo sẵn: GridBagConstraints.NORTH, EAST, WEST, SOUTH, NORTHEAST, SOUTHEAST, NORTHWEST, SOUTHWEST.
- **fill**: Xác định kiểu đặt khi component có kích thước lớn hơn ô sẽ được đặt vào. Các hằng khai báo sẵn: GridBagConstraints.NONE, HORIZONTAL, VERTICAL, BOTH.
- **insets**: Đặc tả khoảng hở <top, bottom, left, right> giữa các phần tử được đưa vào, mặc định là 0.
- **ipadx, ipady**: Khoảng đệm (số pixel trống) bên trong của phần tử theo 2 chiều. Mặc định là 0. Khi vẽ phần tử, sẽ thêm 2\*ipadx và 2\*ipady vào chiều rộng tối thiểu và chiều cao tối thiểu của phần tử.

# GridBagLayout



# GridBagLayout

```
package container;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

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

public class GridBagLayoutDemo extends JFrame{
    public GridBagLayoutDemo() {
        setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        c.gridx = 0;
        c.gridy = 0;
        add(new JButton("Button1"), c);
        c.gridx = 1;
        add(new JButton(" Button2"), c);
        c.gridx = 2;
```

# Null Layout

```
add(new JButton(" Button3"), c);
c.gridx = 0;
c.gridy = 1;
c.gridwidth = 2;
c.fill = GridBagConstraints.BOTH;
add(new JButton(" Button4"), c);
c.gridx = 2;
c.gridwidth = 1;
add(new JButton(" Button5"), c);
setSize(400, 300);
//hien thi giua man hinh
setLocationRelativeTo(null);
setDefaultCloseOperation(EXIT_ON_CLOSE);
}

public static void main(String[] args) {
    GridBagLayoutDemo frame1 = new GridBagLayoutDemo();
    frame1.setVisible(true);
}
}
```

# Null Layout

- Một khung chứa được trình bày theo kiểu Null Layout có nghĩa là người lập trình phải tự làm tất cả từ việc qui định kích thước của khung chứa, cũng như kích thước và vị trí của từng đối tượng component trong khung chứa.
- Để thiết lập cách trình bày là Null Layout cho một container ta chỉ việc gọi phương thức **setLayout(null)** với tham số là **null**.

# Null Layout

Một số phương thức của lớp trừu tượng Component dùng để định vị và qui định kích thước của component khi đưa chúng vào khung chứa trình bày theo kiểu kiểu tự do:

- public void setLocation(Point p)
- public void setSize(Dimension p)
- public void setBounds(Rectangle r)

**Ví dụ:**

- MyButton.setSize(new Dimension(20, 10));
- MyButton.setLocation(new Point(10, 10));
- MyButton.setBounds(10, 10, 20, 10);

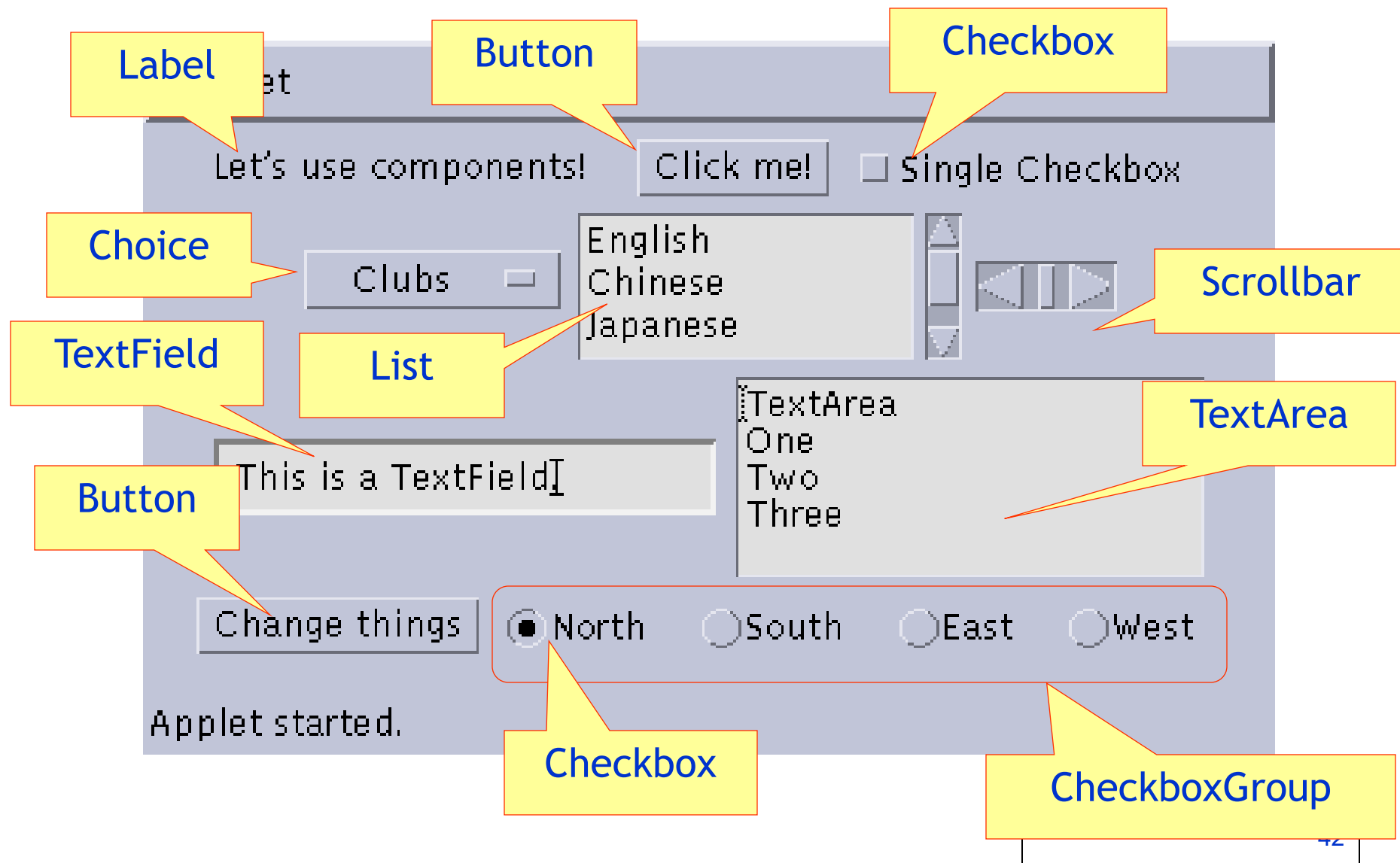
# **CÁC THÀNH PHẦN CƠ BẢN (COMPONENTS)**



# CÁC COMPONENTS CỦA GUI

- Tất cả các thành phần cấu tạo nên chương trình GUI được gọi là component.
- Ví dụ
  - Frame, Window, Dialog, Applet,...
  - TextFields, Labels, CheckBoxes, TextArea, Button, Choice, List, Scrollbars,...

# CÁC COMPONENTS CỦA GUI



# CÁC COMPONENTS CỦA GUI

**Khu vực thiết kế giao diện đồ hoạ**

**Thành phần đồ hoạ dùng để thêm vào màn hình**

**Thuộc tính của một thành phần trong giao diện**

**Ví dụ thiết lập tiêu đề cho màn hình là "Login Form"**

**Swing Containers**

- Panel
- Scroll Pane
- Internal Frame
- Tabbed Pane
- Tool Bar
- Layered Pane
- Split Pane
- Desktop Pane

**Swing Controls**

- Label
- Check Box
- Combo Box
- Text Area
- Progress Bar
- Spinner
- Editor Pane
- OK Button
- Radio Button
- List
- Scroll Bar
- Formatted Field
- Separator
- Tree
- Toggle Button
- Button Group
- Text Field
- Slider
- Password Field
- Text Pane
- Table

**Swing Menus**

**Swing Windows**

**[JFrame] - Properties**

Properties Binding Events Code

**Properties**

defaultCloseOperation	EXIT_ON_CLOSE
title	Login Form

**Other Properties**

background	[240,240,240]
bounds	<Not Set>
cursor	Default Cursor
enabled	<input checked="" type="checkbox"/>

# CÁC COMPONENTS CỦA GUI

The image shows a Java Swing window titled "Student Detail" with a light blue border. Inside, there are several input fields and controls:

- Name:** A single-line text field.
- Qualification:** A dropdown menu currently showing "Graduate".
- Address:** A multi-line text area.
- Hobby:** A group box containing three unchecked checkboxes: "Reading", "Singing", and "Dancing". This group is circled in red, and a callout box labeled "JPanel" points to it.
- Sex:** Two radio buttons labeled "Male" and "Female".
- Buttons:** "Validate" and "Reset" buttons at the bottom.

# CÁC METHOD CHUNG CỦA COMPONENTS

- Các component đều có các phương thức cơ bản sau:

TT	PHƯƠNG THỨC	Ý NGHĨA
	setSize(width, height)	cài đặt kích thước.
	setLocation(x, y)	cài đặt vị trí (lấy vị trí góc trên bên trái làm gốc).
	setBound(x, y, width, height)	là phương thức ghép chung cả setLocation và setSize.
	setBackground(color)	cài đặt màu nền. Có 2 cách truyền tham số màu: hoặc là dùng màu được quy ước sẵn trong lớp Color, ví dụ như "Color.white", hoặc tạo một đối tượng Color, ví dụ "new Color(255, 0, 0)".
	setForeground(color)	cài đặt màu chữ.
	setVisible(boolean)	cài đặt ẩn hay hiện. Thường thì chỉ Frame hay Window bắt buộc phải thiết lập "setVisible(true)", còn các component khác thì mặc định thiết lập này true rồi.

# CÁC METHOD CHUNG CỦA COMPONENTS

- Phương thức cơ bản của JFrame sau:

TT	PHƯƠNG THỨC	Ý NGHĨA
	<code>setTitle("Title")</code>	cài đặt tên tiêu đề
	<code>setLocationRelativeTo(null)</code>	cài đặt ko cho phép kéo thả thay đổi kích thước cửa sổ.
	<code>setDefaultCloseOperation(DO_NOTHING_ON_CLOSE)</code>	lựa chọn ko làm gì khi bạn nhấn nút đóng cửa sổ (nút chéo đỏ). Bạn có thể đặt giá trị “EXIT_ON_CLOSE” để thoát chương trình khi nhấn nút đóng, tuy nhiên cách này ko nên dùng vì ko phải lúc nào nó cũng thoát hoàn toàn
	<code>setLayout(layout)</code>	cài đặt cách bố trí các component trong container. Về các loại Layout mình sẽ trình bày sau
	<code>add(component)</code>	cài đặt màu chữ.
	<code>setVisible(boolean)</code>	sau khi khởi tạo component thì chúng ta thêm component đó vào container, ví dụ “add(mainPanel)”. Lưu ý phải thêm vào khung chứa thì component đó mới được hiển thị.

# CÁC METHOD CHUNG CỦA COMPONENTS

Phương thức cơ bản của JPanel tương tự JFrame

- `setLayout(layout)`
- `add(component)`

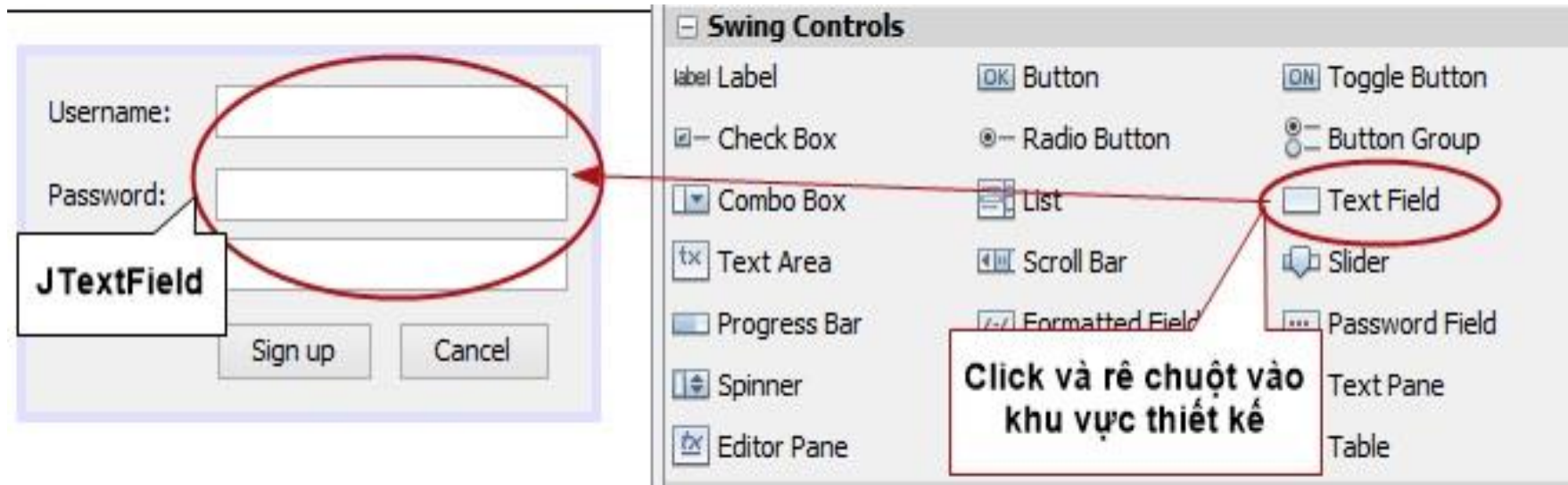
# Swing – JLabel

- **JLabel** là một thành phần để hiển thị văn bản tĩnh (static text). Một JLabel cũng có thể hiển thị icon hoặc cả hai.
- Hai phương thức quan trọng của JLabel là
  - **setText(String label)**: dùng để thiết lập nội dung cho JLabel
  - **String label = getText()** dùng để lấy nội dung của JLabel
  - **setFont(new Font("VNI", Font.PLAIN, 24))**
  - **setOpaque(true)**: mặc định màu nền của Label là trong suốt, đó là bạn phải cài đặt tính đục bằng true thì phương thức cài đặt màu nền setBackground mới có hiệu lực.
  - **setHorizontalAlignment(JLabel.CENTER)**: căn text vào giữa Label theo hàng ngang.
  - **setVerticalAlignment(JLabel.CENTER)**: căn text vào giữa Label theo hàng dọc.



# Swing – JTextField

- **JTextField** cho phép người dùng nhập và chỉnh sửa một dòng văn bản.
- Ba phương thức quan trọng của JTextField



# Swing – JTextField

- Phương thức:

Tên	Miêu tả
<b>String text = getText();</b>	Lấy nội dung của JTextField
<b>setText(String value);</b>	Thiết lập nội dung của JTextField
<b>setEditable(boolean editable)</b>	Thiết lập cho phép chỉnh sửa nội dung hay không. Nếu editable = true, được phép chỉnh sửa. Ngược lại, không được phép chỉnh sửa.
<b>setFont()</b>	Tương tự

# Swing – JTextArea

- JTextArea cho phép nhập, chỉnh sửa nhiều dòng văn bản.

The image shows a Swing window with a form. It contains the following elements:

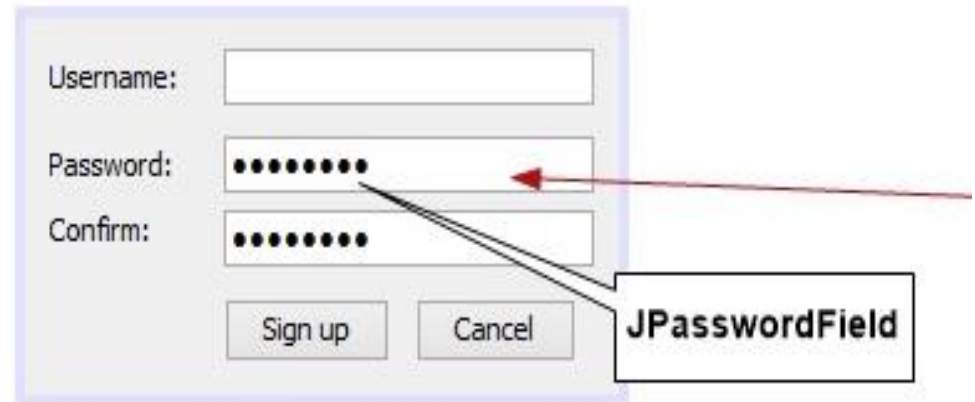
- Name:** A single-line text input field.
- Qualification:** A dropdown menu currently showing 'Graduate'.
- Address:** A multi-line text input field (JTextArea).
- Hobby:** A group box containing three checkboxes: 'Reading', 'Singing', and 'Dancing'. A red arrow points from this group to the 'Address' field.
- Sex:** A group box containing two radio buttons: 'Male' and 'Female'.
- Buttons:** 'Validate' and 'Reset' buttons at the bottom right.

A label 'JTextArea' with a pointer indicates the 'Address' field.

Tên	Miêu tả
<code>String text = getText();</code>	Lấy nội dung của JTextArea
<code>setText(String value);</code>	Thiết lập nội dung của JTextArea
<code>setEditable(boolean editable)</code>	Cho phép/không cho chỉnh sửa nội dung
<code>copy()</code> và <code>cut()</code>	Chuyển vùng văn bản được chọn từ JTextArea vào clipboard
<code>paste()</code>	Chuyển nội dung từ clipboard vào JTextArea
<code>setFont()</code>	

# Swing – JPasswordField

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



Tên	Miêu tả
<b>String text = getText();</b>	Lấy nội dung của JPasswordField
<b>setText(String value);</b>	Thiết lập nội dung của JPasswordField
<b>setEchoChar()</b>	Thiết lập ký tự hiển thị

# Swing – JCheckBox, JRadioButton

- Một thành phần có 2 trạng thái là chọn (checked) và không chọn (unchecked). Trạng thái mặc định của JCheckBox là unchecked. Người dùng có thể chọn cùng lúc nhiều lựa chọn
- JRadioButton tương tự JCheckBox ngoại trừ tại một thời điểm chỉ cho phép chọn một. Khi tạo nhiều JRadioButton, phải kết hợp với Button Group để ràng buộc thao tác chọn của người dùng

The image shows a Swing window with the following components:

- Name:** A text input field.
- Address:** A text area.
- Sex:** Two radio buttons labeled "Male" and "Female".
- Qualification:** A dropdown menu currently showing "Graduate".
- Hobby:** Three JCheckBoxes labeled "Reading", "Singing", and "Dancing". A red arrow points to the "Singing" checkbox, and a callout box labeled "JCheckBox" points to the "Dancing" checkbox.
- Buttons:** "Validate" and "Reset" buttons at the bottom right.

Tên	Miêu tả
<b>boolean isSelected()</b>	Trả về true nếu JCheckBox đang được chọn và false nếu ngược lại
<b>setSelected(boolean state)</b>	Thiết lập chọn hoặc bỏ chọn

# Swing – JList

- **JList** là một component hiển thị danh sách các đối tượng, cho phép người dùng chọn được item
- Bản thân JList chỉ là thành phần hiển thị. Để nạp dữ liệu cho JList hiển thị, cần có đối tượng model để chứa dữ liệu đó là DefaultListModel. Thông thường để truyền dữ liệu vào model cần có mảng dữ liệu, mảng đó là kết quả của các quá trình tìm kiếm, sắp xếp. Con đường dữ liệu được hiển thị ra JList như sau:
  - **ArrayList** → DefaultListModel → JList
- Ngoài ra JList thường phải đặt trong một loại component khung chứa là JScrollPane, vì JList ko hỗ trợ thanh cuộn, thanh cuộn là do JScrollPane cung cấp.

# Swing – JList

```
private DefaultListModel lstModelStudent;  
private JList lstStudent;  
private JScrollPane scroll;  
....  
lstStudent = new JList();  
scroll = new JScrollPane(lstStudent);  
add(scroll);  
updateDataListModelStudent();  
....  
private void updateDataListModelStudent() {  
    ArrayList listStudent = manager.getListStudent();  
    lstModelStudent = new DefaultListModel();  
    for (Student item : listStudent) {  
        lstModelStudent.addElement(item);  
    }  
    lstStudent.setModel(lstModelStudent);  
}
```

# Swing – JComboBox, JButton

- **JComboBox** là thành phần cho phép người dùng lựa chọn từ danh sách
- **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

The image shows a Java Swing window with a light gray background. It contains several input fields and controls:

- Name:** A single-line text field.
- Qualification:** A JComboBox with a dropdown arrow, currently showing "Graduate".
- Address:** A multi-line text area.
- Hobby:** A group box containing three JCheckBox components: "Reading", "Singing", and "Dancing". A red arrow points to the "Singing" checkbox, and a callout box labeled "JCheckBox" points to it.
- Sex:** A group box containing two radio buttons: "Male" and "Female".
- Buttons:** Two buttons at the bottom: "Validate" and "Reset".




# Swing – JComboBox

- Tương tự JList, chỉ khác là nó hiển thị danh sách dạng sổ xuống và có thuộc tính lựa chọn.
- **ArrayList** → **DefaultComboBoxModel** → **JComboBox**

```
private DefaultComboBoxModel cbbModelStudent;  
private JComboBox cbbStudent;  
....  
cbbStudent = new JComboBox();  
add(cbbStudent);  
updateDataComboBoxModelStudent();  
....  
private void updateDataComboBoxModelStudent() {  
    ArrayList listStudent = manager.getListStudent();  
    cbbModelStudent = new DefaultComboBoxModel();  
    for (Student item : listStudent) {  
        cbbModelStudent.addElement(item);  
    }  
    cbbStudent.setModel(cbbModelStudent);  
}
```

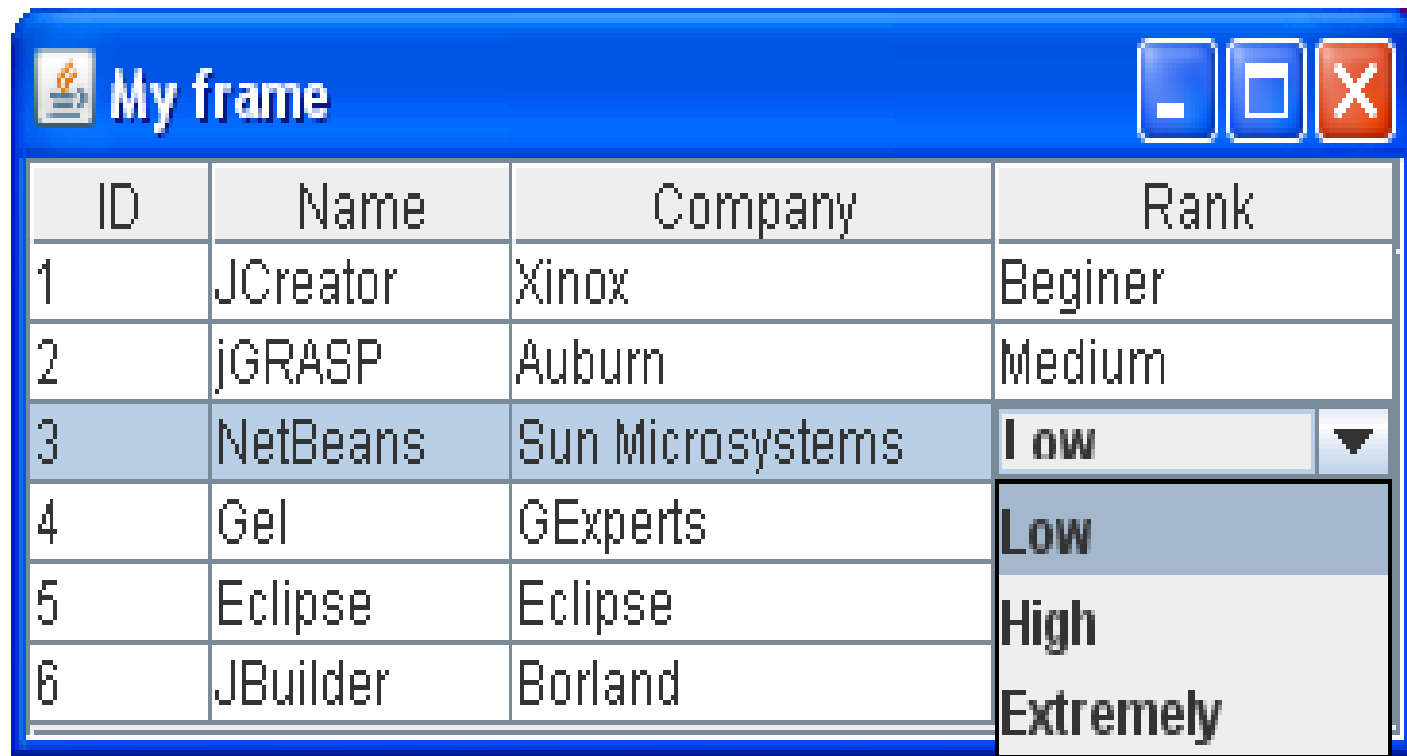
# JTable

```
JFrame f = new JFrame("My frame");  
String[ ][ ] dat={{ "1","JCreator","Xinox","Beginer"},  
                  {"2","jGRASP","Auburn","Medium"},  
                  {"3","NetBeans","Sun Microsystems","Expert"},  
                  {"4","Gel","GExperts","Beginer"},  
                  {"5","Eclipse","Eclipse","Expert"},  
                  {"6","JBuilder","Borland","Expert"}};  
String[ ] columnName = {"ID","Name","Company","Rank"};  
JTable t = new JTable(dat,columnName);  
JScrollPane s = new JScrollPane(t);  
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); f.add(s);  
f.setSize(500,150); f.setVisible(true);
```



ID	Name	Company	Rank
1	JCreator	Xinox	Beginer
2	jGRASP	Auburn	Medium
3	NetBeans	Sun Microsystems	Expert
4	Gel	GExperts	Beginer
5	Eclipse	Eclipse	Expert
6	JBuilder	Borland	Expert

# JTable



My frame

ID	Name	Company	Rank
1	JCreator	Xinox	Beginner
2	jGRASP	Auburn	Medium
3	NetBeans	Sun Microsystems	Low
4	Gel	GExperts	Low
5	Eclipse	Eclipse	High
6	JBuilder	Borland	Extremely

# Swing – JTable

- Tương tự JList, hiển thị dữ liệu ra màn hình như sau:
- **ArrayList** → **DefaultTableModel** → **JTable**

```
private static final String COLUMN_NAME = {"Mã HS", "Tên",  
"Tuổi"};  
private DefaultTableModel tbModelStudent;  
private JTable tbStudent;  
private JScrollPane scroll;  
....  
tbStudent = new JTable();  
scroll = new JScrollPane(tbStudent);  
add(scroll);  
updateDataTableModelStudent();  
....
```

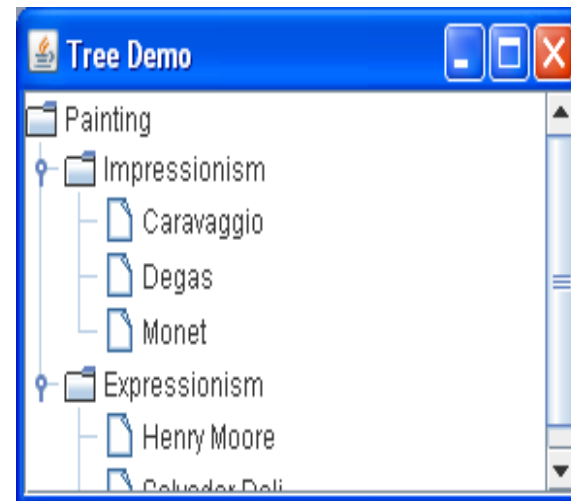
**JTable** chỉ có khả năng hiển thị dữ liệu dạng String, nên với dữ liệu kiểu số hay boolean thì cần chuyển đổi thành chuỗi

```
private void updateDataTableModelStudent() {  
    ArrayList listStudent = manager.getListStudent();  
    tbModelStudent = new  
    DefaultTableModel l(COLUMN_NAME, 0);  
    for (Student item : listStudent) {  
        String[] arr = new String[3];  
        arr[0] = item.getId();  
        arr[1] = item.getName();  
        arr[2] = item.getAge();  
        tbModelStudent.addRow(arr);  
    }  
    tbStudent.setModel(tbModelStudent);  
}
```

# JTree

- ❑ Windows Explorer has a tree like structure depicting files and folders.
- ❑ Windows Explorer structures can be created in Java using JTree.
- ❑ Every row in the hierarchy is termed as a node.

**When the nodes  
are clicked**



# Bài tập

- Thiết kế giao diện

The screenshot shows a Java Swing window titled "Student Form" with a blue title bar. Inside the window, the components are arranged in a simple vertical sequence: a "Name:" label followed by a text field, a "Birth:" label followed by a text field, an "Address:" label followed by a text field, and a "Sex:" label followed by two radio buttons labeled "Male" and "Female". At the bottom of the window, there are four buttons: "Add", "Find", "Delete", and "Update".

This screenshot is identical to the one above but includes red annotations to highlight specific layout managers used in the design. A red box outlines the top three input sections (Name, Birth, Address), with a red arrow pointing from a red callout box labeled "GridBagLayout". Another red box outlines the "Sex:" section with its two radio buttons, with a red arrow pointing from a red callout box labeled "GridLayout". A third red box outlines the bottom button bar, with a red arrow pointing from a red callout box labeled "BorderLayout.South".

GridBagLayout

GridLayout

BorderLayout.South

# Bài tập

- **Gợi ý:**
  - 2 Panel (Center + South của Frame)
  - Panel trên sử dụng GridBagLayout
    - Chú ý cần set **weightx** và **weighty** để chiếm hết cả frame
    - Hàng chứa radio button thêm 1 panel và set thành GridLayout

- **Bài chữa ->**



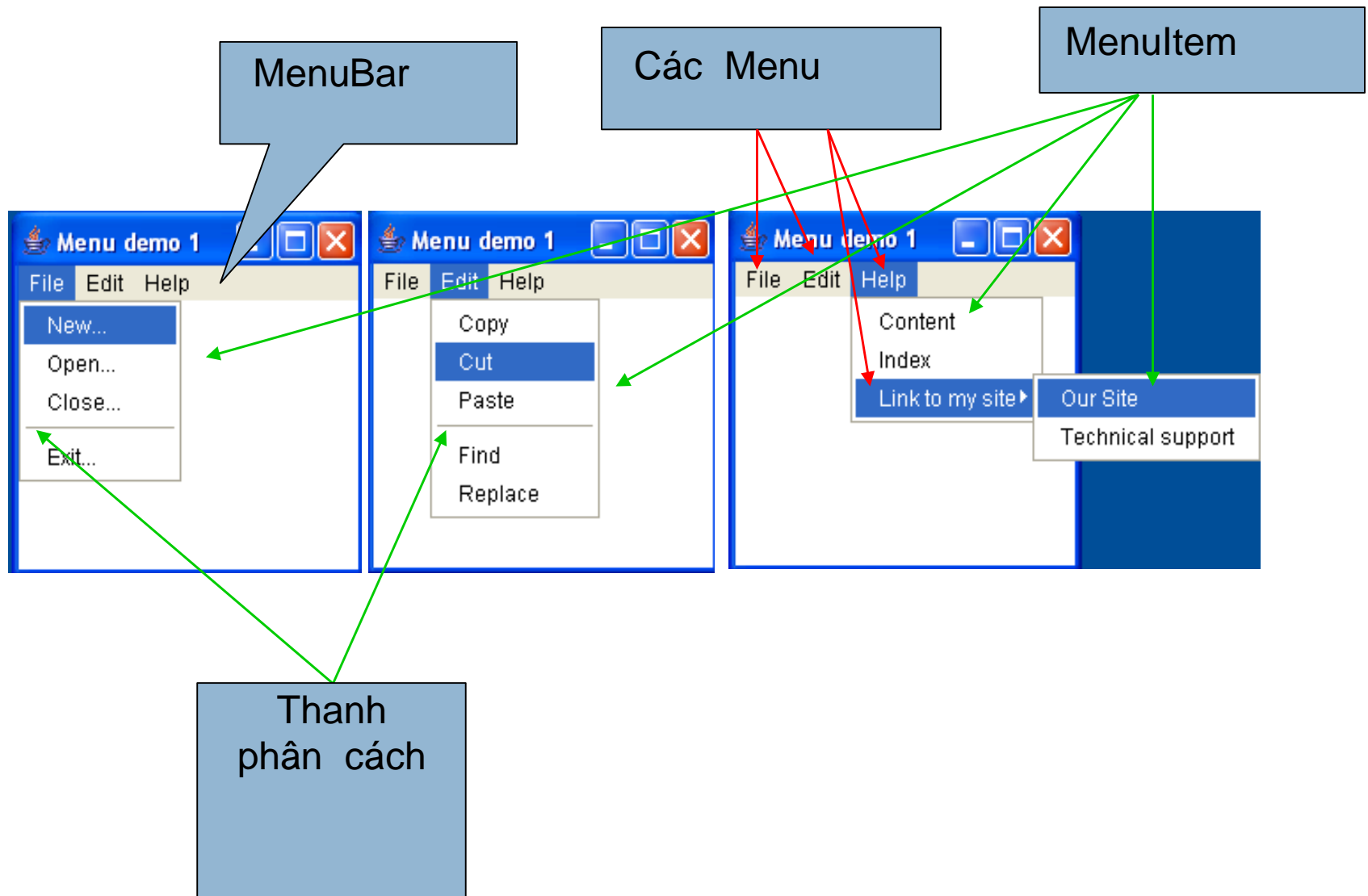
StudentFrame.java

# Menu

- **Hệ thống menu** (Menu System): Tập các mục chọn chức năng của ứng dụng được tổ chức phù hợp.
- Ngôn ngữ Java có một **tập hợp các lớp đối tượng để tạo các menu**.
- Có hai loại menu
  - **Pull-down menu**
  - **Pop-up menu**
- Chỉ có thể đặt các menubar vào trong các **Frame/JFrame**, Mỗi Frame chỉ chứa duy nhất một menubar



# Cấu trúc một hệ Menu



# Các lớp liên quan đến menu

## Hierarchy For Package java.awt

class java.awt.[MenuComponent](#) (implements java.io.[Serializable](#))

- o class java.awt.[MenuBar](#) (implements javax.accessibility.[Accessible](#), java.awt.[MenuContainer](#))

- o class java.awt.[MenuItem](#) (implements javax.accessibility.[Accessible](#))

  - o class java.awt.[CheckboxMenuItem](#) (implements javax.accessibility.[Accessible](#), java.awt.[ItemSelectable](#))

  - o class java.awt.[Menu](#) (implements javax.accessibility.[Accessible](#), java.awt.[MenuContainer](#))

    - o class java.awt.[PopupMenu](#)

class java.awt.[MenuShortcut](#) (implements java.io.[Serializable](#))

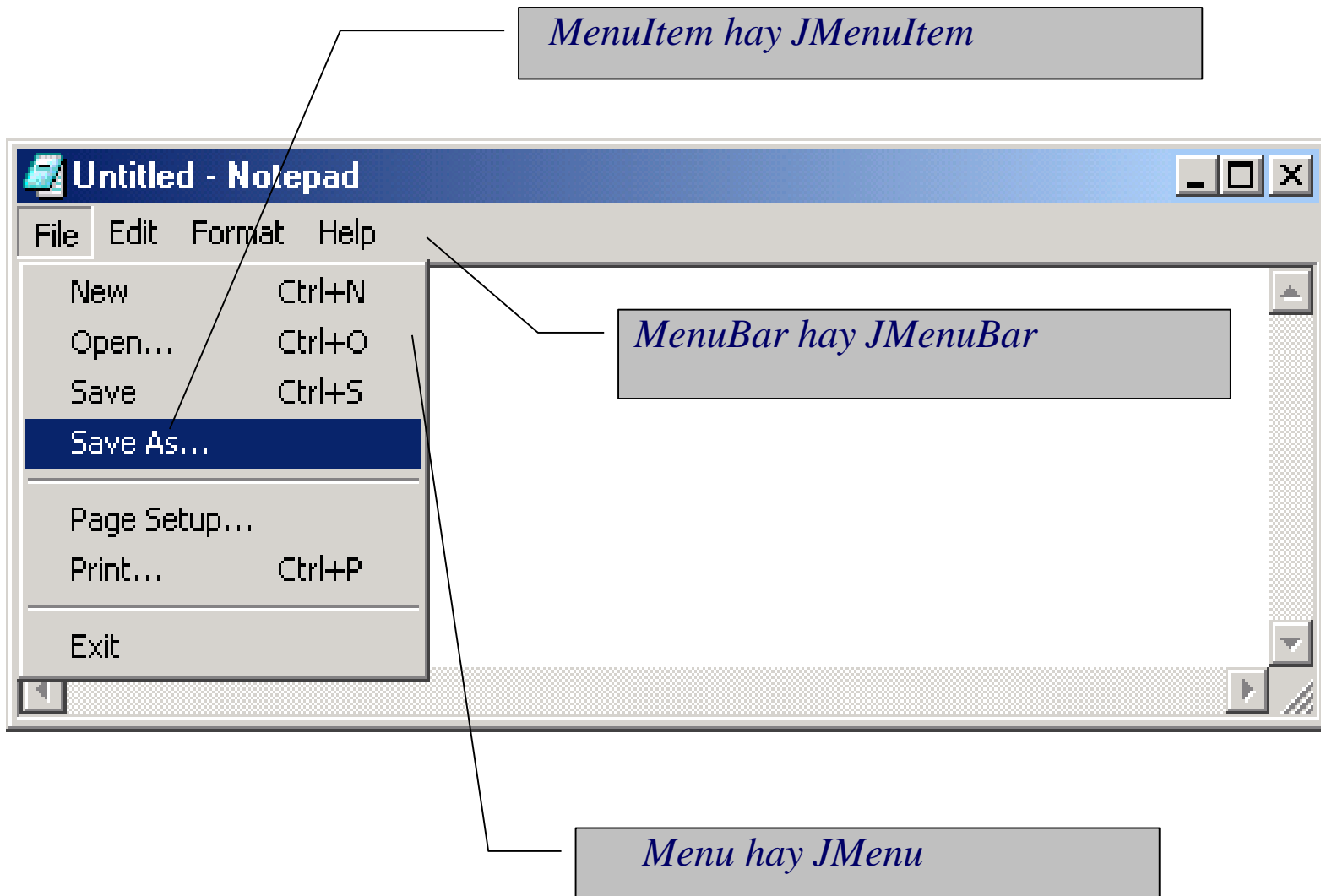
javax.swing

## Interface MenuElement

All Known Implementing Classes:

[JMenu](#), [JMenuBar](#), [JMenuItem](#), [JPopupMenu](#)

# Ví dụ



# Tạo hệ thống menu (1)

## □ MenuBar/JMenuBar

- MenuBar()/JMenuBar()

## □ MenuItem

- MenuItem()
- MenuItem(String label)
- MenuItem(String label, MenuShortcut s)

## □ JMenuItem

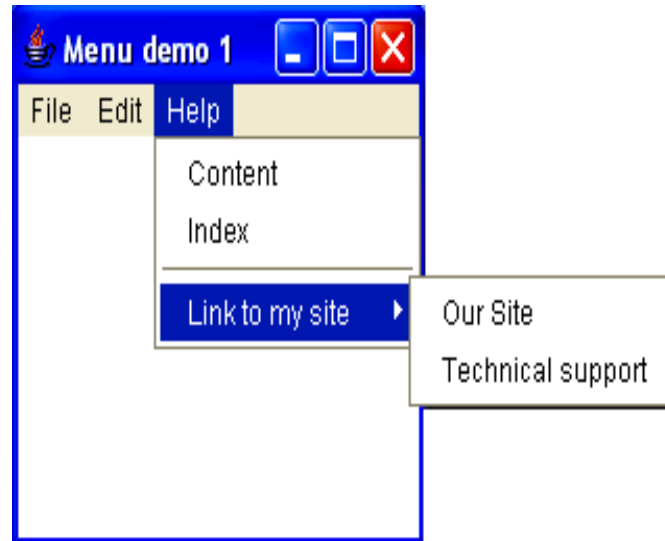
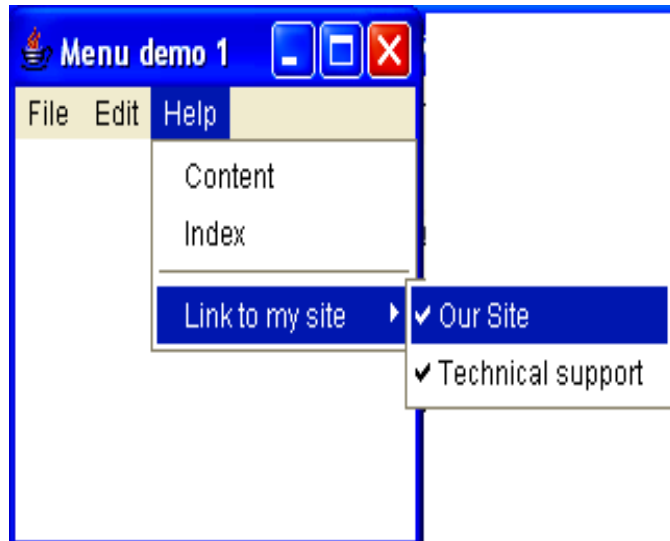
## □ Menu/JMenu

- Menu()
- Menu(String label)

# Tạo hệ thống menu (2)

## ☐ CheckboxMenuItem

- `CheckboxMenuItem()`
- `CheckboxMenuItem(String label)`
- `CheckboxMenuItem(String label, boolean state)`

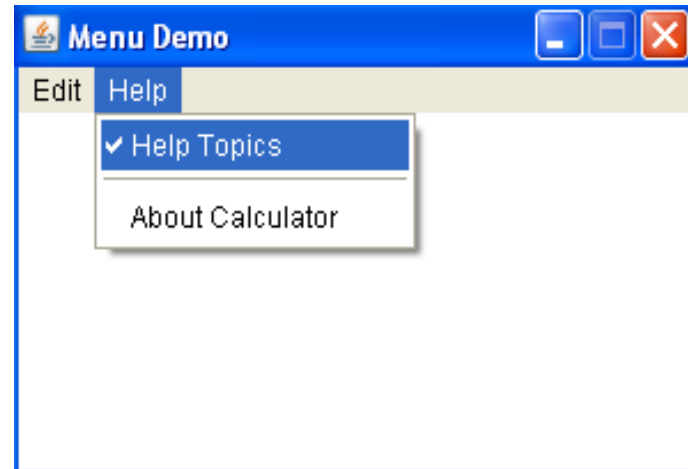
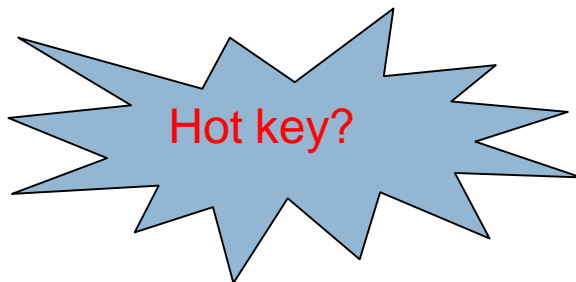
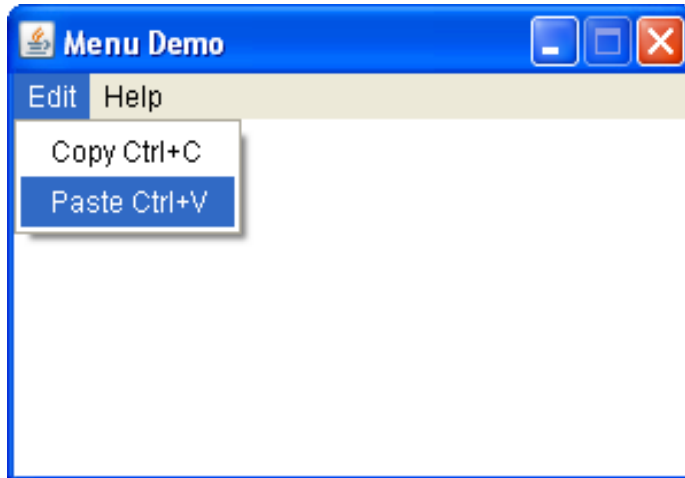


# Phím tắt – MenuShortcut

- **Shortcut Key**: Tổ hợp **Ctrl + Phím** sẽ tác động vào 1 mục chọn tương tự như kích chuột vào 1 mục menu.
- Lớp **java.awt.MenuShortcut** hỗ trợ thiết lập các phím nóng.
- Lớp **java.awt.event.KeyEvent** định nghĩa sẵn các phím
- Ví dụ thiết lập phím nóng cho MenuItem:
  - **MenuShortcut** CtrlN = **new MenuShortcut(KeyEvent.VK\_N)**;  
**mnuNew.setShortcut(CtrlN); //Ctrl+N**
  - **mnuOpen.setShortcut(new MenuShortcut(KeyEvent.VK\_O))**;

# Ví dụ

❖ Chương trình minh họa cách thiết lập menu cho ứng dụng:



**MenuDemo**

**MenuHotkeyDemo**

# LẬP TRÌNH HƯỚNG SỰ KIỆN

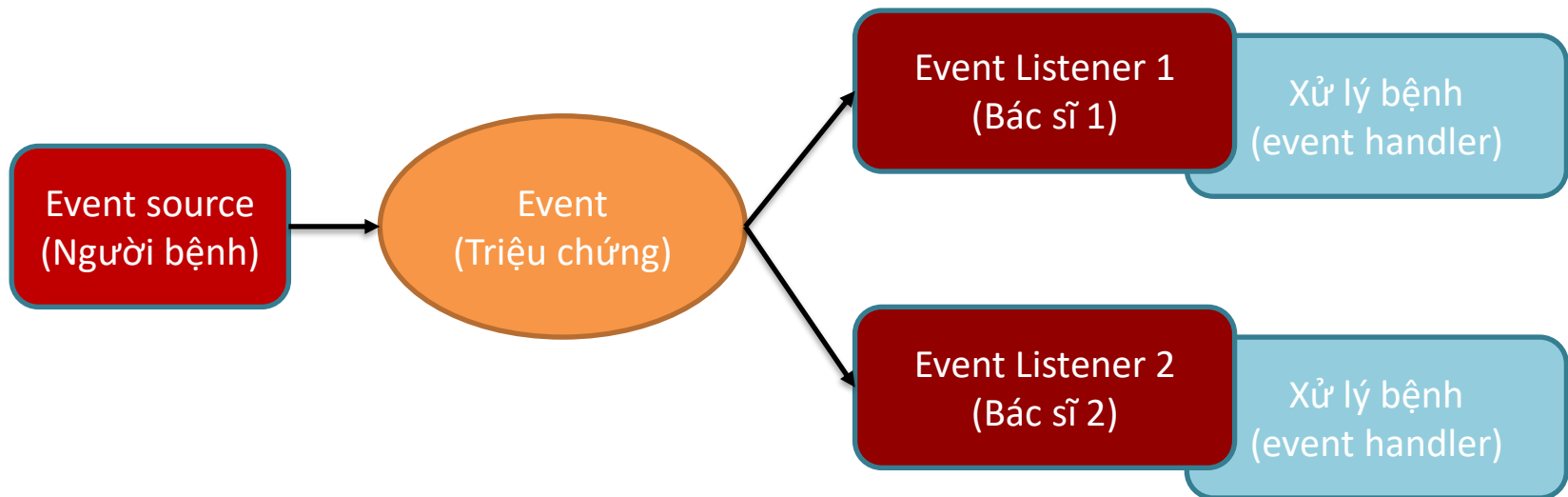


# Event

- Event : một tín hiệu mà ứng dụng nhận biết có sự thay đổi trạng thái của 1 đối tượng.
- 3 nguồn phát xuất event:
  - (1) User( gõ phím, kích chuột vào 1 phần tử,...)
  - (2) Do hệ thống (do định thời 1 tác vụ)
  - (3) Do 1 event khác ( các event kích hoạt nhau)
- Hiện nay, đa số các ngôn ngữ đều cung cấp mô hình này, VC++ cung cấp MFC (Microsoft Foundation Classes), Java cung cấp JFC (Java Foundation Classes).

# Ủy thác xử lý sự kiện

- Khi một sự kiện xảy ra, việc xử lý tùy thuộc vào người được ủy thác sự kiện. Có ba thành phần tham gia vào hệ thống event:
  - **Event nguồn:** là đối tượng tạo ra sự thay đổi
  - **Đối tượng event:** là chính bản thân cái event đó đã được mã hóa
  - **Đối tượng lắng nghe event:** làm công việc xử lý event đó.



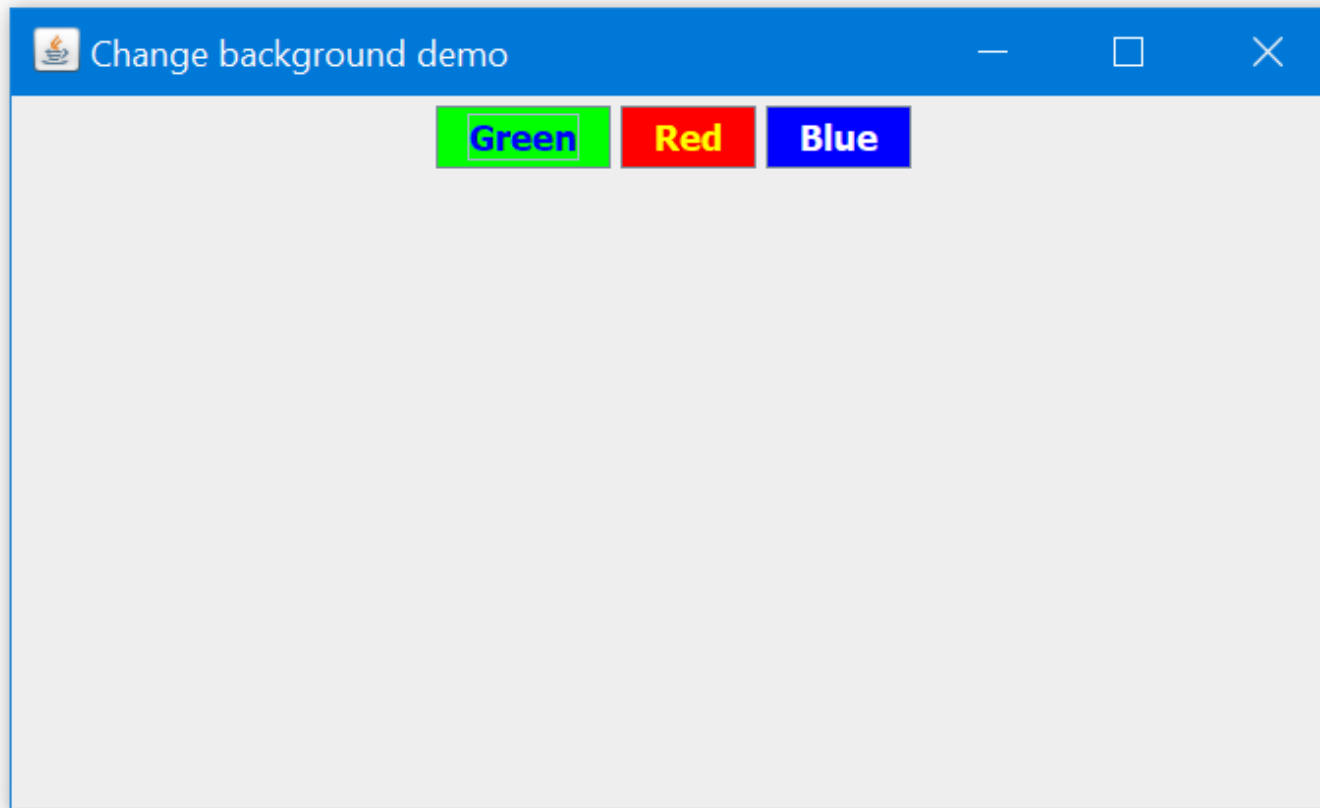
# Một số khái niệm

- **Event** : Là sự kiện phát sinh khi có 1 đối tượng đã thay đổi trạng thái (*VD khi người dùng nhấn chuột vào nút nhấn (JButton), chọn hoặc không chọn JCheckBox/JRadioButton,...*)
- **Event handler**: Là đoạn code để đạt phản ứng của ứng dụng khi gặp 1 event.
- **Event source**: Đối tượng kích hoạt event (thí dụ: nút lệnh bị user kích chuột).
- **Listener** : Đối tượng nhận sự ủy nhiệm xử lý sự kiện cho đối tượng khác
- Java định nghĩa sẵn các Listener Interface cho các tình huống khác nhau (mỗi Event object có Listener interface xử lý tương ứng).
- Một lớp có khả năng listener sẽ phải cụ thể hóa - viết code một số hành vi xử lý một event phù hợp.

# Swing – Xử lý sự kiện

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)
Window	WindowListener	void windowActivated(WindowEvent we) void windowClosed(WindowEvent we) void windowClosing(WindowEvent we) void windowDeactivated(WindowEvent we) void windowDeiconified(WindowEvent we) void windowIconified(WindowEvent we) void windowOpened(WindowEvent we)

# Ví dụ



Thay đổi màu nền frame khi kích vào các nút màu tương ứng

# Ví dụ

```
...
public class EventDemoFrame extends JFrame implements ActionListener {
    JButton buttonGreen = new JButton("Green");
    JButton buttonBlue = new JButton("Blue");
    JButton buttonRed = new JButton("Red");

    public EventDemoFrame() {
        setLayout(new FlowLayout());
        buttonGreen.addActionListener(this);
        add(buttonGreen);
        buttonRed.addActionListener(this);
        add(buttonRed);
        buttonBlue.addActionListener(this);
        add(buttonBlue);

        setSize(400, 300);
        setTitle("Change background demo");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

# Ví dụ

```
@Override
public void actionPerformed(ActionEvent e) {
    if(e.getSource().equals(buttonGreen)) {
        getContentPane().setBackground(Color.GREEN);
    }
    else if(e.getSource().equals(buttonBlue)) {
        getContentPane().setBackground(Color.BLUE);
    }
    else if(e.getSource().equals(buttonRed)) {
        getContentPane().setBackground(Color.RED);
    }
}

public static void main(String[] args) {
    EventDemoFrame frame = new EventDemoFrame();
    frame.setVisible(true);
}
}
```

# Các event

- Các component sẽ phát sinh ra các event khác nhau
- Mỗi event sẽ do 1 listener tương ứng xử lý
- Listener là interface
- Adapter cài đặt mặc định cho các phương thức của Listener.
- Nếu extends Adapter sẽ không bắt buộc phải cung cấp cài đặt cho tất cả các phương thức của các interface Listener -> giúp tiết kiệm code



# Action Event

- Một **ActionEvent** object được sinh ra khi: 1 nút lệnh bị kích, một mục chọn trong danh sách bị kích đôi, 1 mục menu bị kích.
- Các hằng kiểm tra có 1 phím bị nhấn khi kích chuột hay không: **ALT\_MASK** (phím Alt), **CTRL\_MASK** (phím Ctrl), **META\_MASK** (phím meta, ký tự mô tả về 1 ký tự khác -ký tự escape), **SHIFT\_MASK** (phím Shift).

# Adjustment Event

- Được sinh ra khi 1 thanh cuộn bị thao tác.
- Các hằng BLOCK\_DECREMENT, BLOCK\_INCREMENT: Độ giảm/tăng theo khối khi user kích chuột vào vùng giữa con trỏ và 1 biên của thanh cuộn, UNIT\_DECREMENT, UNIT\_INCREMENT: Đơn vị giảm/tăng khi user kích chuột vào mũi tên ở 2 đầu thanh cuộn. TRACK: Giá trị mô tả thanh cuộn khi bị user kéo

# Container Event

- Được sinh ra khi 1 component được thêm/xóa khỏi 1 container.
- Các hằng mô tả sự kiện: `COMPONENT_ADDED`, `COMPONENT_REMOVED`.

# Component Event

- Được sinh ra khi 1 componet bị ẩn đi, được hiển thị, bị di chuyển, bị thay đổi kích thước.
- Các hằng mô tả trạng thái gồm:
  - COMPONENT\_HIDDEN,
  - COMPONENT\_MOVED,
  - COMPONENT\_RESIZED,
  - COMPONENT\_SHOWN

# Input Event

- Là lớp cha của 2 lớp con: KeyEvent và MouseEvent.
- Các hằng khai báo trong lớp này mô tả các bit mặt nạ truy xuất phím đi kèm sự kiện hoặc nút chuột nào bị nhấn:  
ALT\_MASK, CTRL\_MASK, META\_MASK, SHIFT\_MASK,  
BUTTON1\_MASK, BUTTON2\_MASK, BUTTON3\_MASK.

# Key Event

- Được sinh ra khi user thao tác với bàn phím .
- Các hằng kiểu intKEY\_PRESSED, KEY\_RELEASED, KEY\_TYPED.  
Nếu phím chữ, phím số được gõ, cả 3 loại sự kiện được sinh ra (pressed, released, typed). Nếu phím đặc biệt được thao tác (phím Home, End, PageUp, PageDown- modifier key), chỉ có 2 sự kiện được sinh ra: pressed, released.

# Mouse Event

- Được sinh ra khi user thao tác chuột với 1 component.
- Các hằng int:
  - `MOUSE_CLICKED`,
  - `MOUSE_DRAGGED`,
  - `MOUSE_ENTERED`,
  - `MOUSE_EXITED`,
  - `MOUSE_MOVED`,
  - `MOUSE_PRESSED`,
  - `MOUSE_RELEASED`.

# Text Event

- Được sinh ra khi các ký tự trong 1 TextField hay 1 textArea bị đổi.
- Hằng int: `TEXT_VALUE_CHANGED`



# Ví dụ Event

Viết sự kiện đóng cửa sổ (đóng Frame) khi kích chọn nút “btthoat”

```
private void btnthoatActionPerformed(  
    java.awt.event.ActionEvent evt) {  
    int ret=JOptionPane.showConfirmDialog(null, "Thoát khỏi  
    chương trình ?", "Thoát", JOptionPane.YES_NO_OPTION);  
    if(ret==JOptionPane.YES_OPTION)  
        dispose(); // hoặc System.exit(0);
```

# Ví dụ Event

Viết sự kiện thay đổi màu nền khi kích chọn các nút tương ứng: buttonGreen, buttonBlue, buttonRed

```
public void actionPerformed(ActionEvent e) {  
    if(e.getSource().equals(buttonGreen)) {  
        getContentPane().setBackground(Color.GREEN);  
    }  
    else if(e.getSource().equals(buttonBlue)) {  
        getContentPane().setBackground(Color.BLUE);  
    }  
    else if(e.getSource().equals(buttonRed)) {  
        getContentPane().setBackground(Color.RED);  
    }  
}
```

# Ví dụ Event

Viết sự kiện chọn hoặc bỏ chọn cho checkBox

```
public void itemStateChanged(ItemEvent e) {  
    if (checkBox.isSelected()) {  
        lbCheck.setText("This CheckBox has checked");  
    } else {  
        lbCheck.setText("This CheckBox has unchecked");  
    }  
}  
});
```

# Tài liệu tham khảo

- <https://viettuts.vn/java-swing>
- <https://viettuts.vn/java-awt/cac-lop-adapter-trong-java-awt>
- [http://www.netfoo.net/oreilly/java/javanut/figs/jn2\\_2001.gif](http://www.netfoo.net/oreilly/java/javanut/figs/jn2_2001.gif)