

# **Session 08**

## Creating Graphical User Interface **(GUI & Swing)**

### **Java GUI and the swing package**

References:

[Java-Tutorials/tutorial-2015/uiswing/index.html](http://Java-Tutorials/tutorial-2015/uiswing/index.html)

**Trinh Thi Van Anh – PTIT**

# Nội dung

- Giới thiệu AWT và Swing
- Xây dựng Java GUI cơ bản
- Top-level container
- Layout Manager
- Common Control
- Event Listener
- Dialogbox
- Advanced Control

- Text component
- Choice component
- Menu
- Tabbed pane
- Scroll pane
- Dialog box
- Jlist
- Jtable

# JFC (Java Foundation Classes)

- Gồm 5 phần chính:
- **AWT** (Abstract Windows Toolkit): là thành phần công cụ thiết kế và lập trình giao diện cơ bản nhất trong Java
- **Swing**
- **Accessibility API**: Là bộ công cụ giúp người dùng kết nối với các thiết bị như bàn phím nổi, bộ đọc chữ tự động ... cho phép truy xuất trực tiếp tới các thành phần Swing.
- **2D API**: chứa các lớp hiện thực nhiều kiểu vẽ, các hình phức tạp, fonts, colors. 2D API không phải là 1 phần của Swing
- **Drag and Drop**: cho phép người dùng chọn giữ một đối tượng GUI rồi di chuyển qua các cửa sổ hoặc frame khác

# Giới thiệu về AWT

- AWT (Abstract Window Toolkit) (**java.awt.\***) cung cấp một tập hợp các lớp dùng để viết giao diện người dùng dạng đồ họa.
- Bộ khung (framework) GUI cũ cho Java (Java 1.1)
- Đặc điểm:
  - Bao gồm tập hợp các lớp ngang hàng, tức là giao diện lập trình ứng dụng cho các tính năng cửa sổ hiện có được cung cấp bởi hệ điều hành.
- Hạn chế:
  - Chiếm nhiều tài nguyên hệ thống (heavyweight object)
  - Khó mở rộng (không có các công nghệ hỗ trợ)
  - Một số dựa vào các bản sao mã bản ngữ (native code)
  - Gặp các vấn đề độc lập hệ nền
  - Phụ thuộc vào các thành phần GUI của hệ điều hành

# Giới thiệu về SWING

- Swing (**javax.swing.\***)
- Bộ khung GUI mới được giới thiệu đầu tiên trong java 1.2
- Bao gồm tất cả những đặc tính của AWT cộng với nhiều đặc tính tiên tiến khác.
- Thuần Java, các thành phần nhẹ (lightweight) (không dựa vào mã bản ngữ)
- Kiến trúc cảm quan (Look and feel)
- Các ưu điểm của Swing:
  - Các thành phần của Swing chiếm ít tài nguyên hệ thống hơn vì chúng không ngang hàng riêng trong hệ điều hành.
  - Hỗ trợ khái niệm “pluggable look-and-feel”, cung cấp thêm nhiều diện mạo để người dùng lựa chọn
  - Hỗ trợ các công nghệ nhập xuất mới: tiếng nói và thao tác không mouse
  - Dễ dàng mở rộng:
    - Button hỗ trợ cả văn bản và đồ họa
    - Sử dụng HTML trong Label
    - ...

# **Swing vs. AWT**

- AWT: tốt cho ứng dụng GUI đơn giản, ứng dụng phức tạp thì không.
- Từ Java 2, AWT được thay thế bởi Swing, mạnh, ổn định và mềm dẻo hơn.
- Swing: ít phụ thuộc vào nền tảng và ít sử dụng tài nguyên.
- Swing - lightweight components và AWT - heavyweight components.

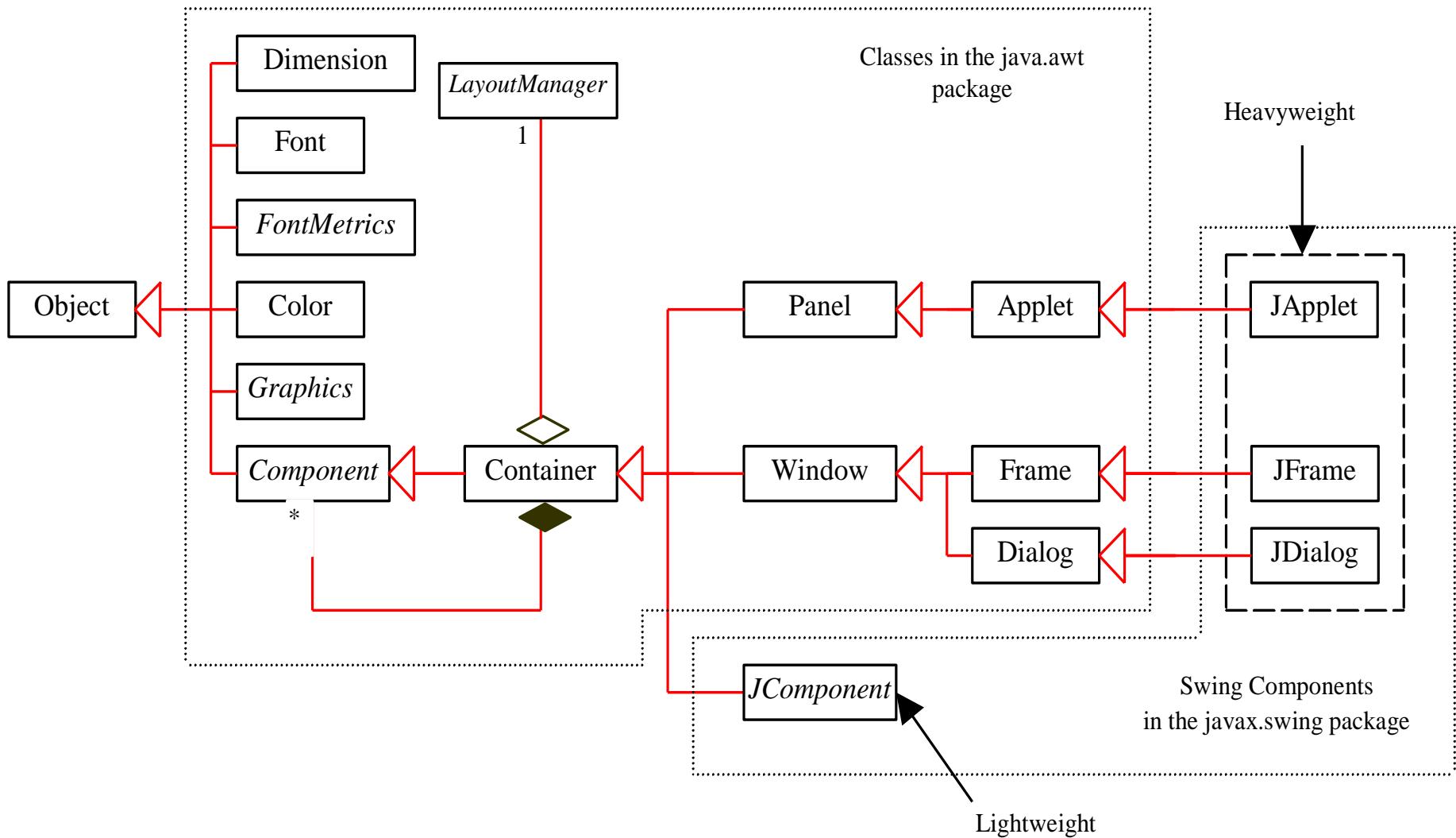
# Giới thiệu Java GUI

- AWT và Swing cung cấp tập hợp các lớp Java cho phép tạo các giao diện đồ họa (GUI)
- Cung cấp các thành phần để tạo hoạt động và hiệu ứng GUI như
  - Container (bộ chứa)
  - Component (thành phần GUI)
  - Layout manager (bộ quản lý bộ cục)
  - Graphic và drawing capabilitie (vẽ đồ họa)
  - Font
  - Event

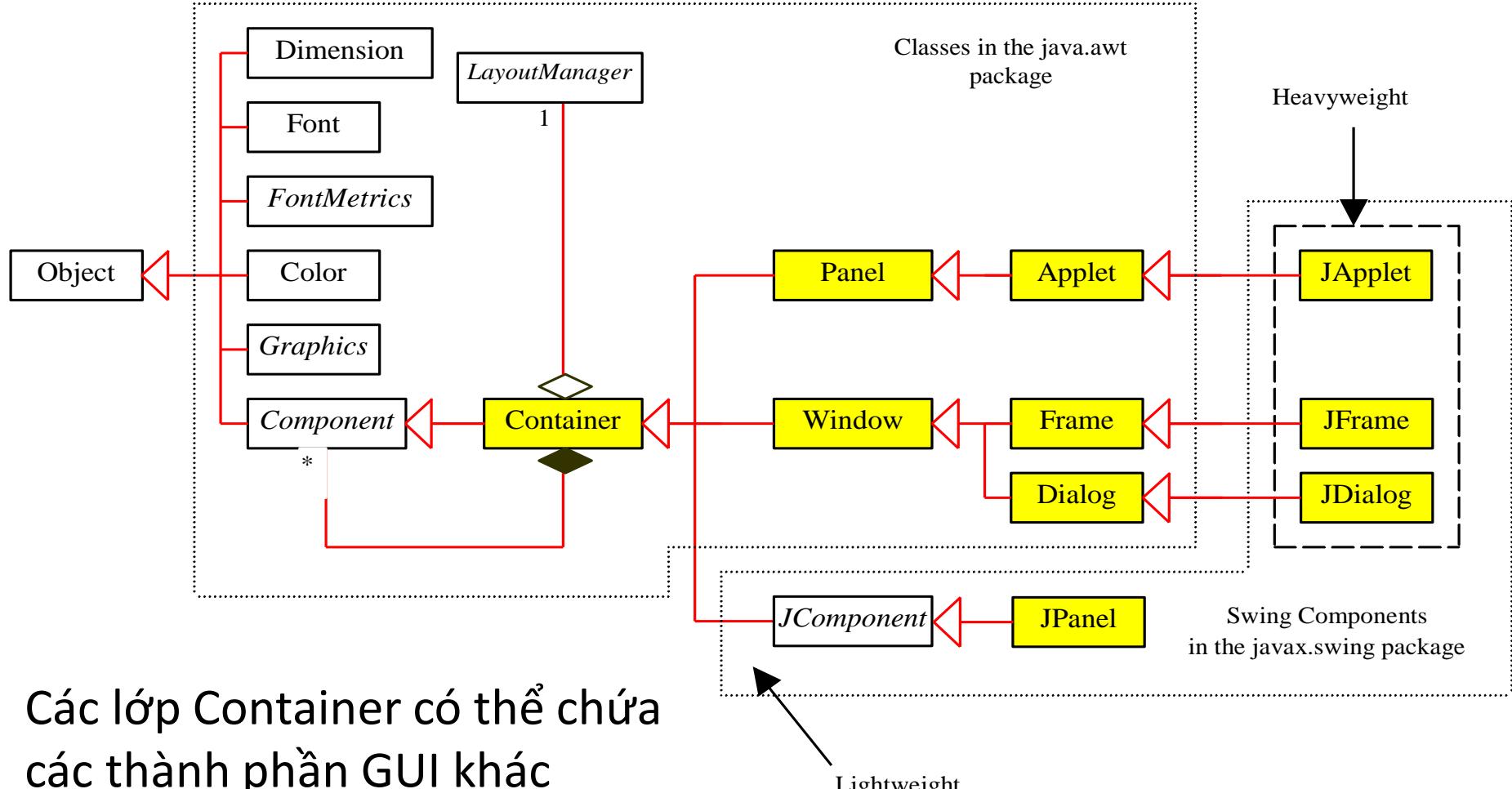
# GUI Class Hierarchy (AWT)



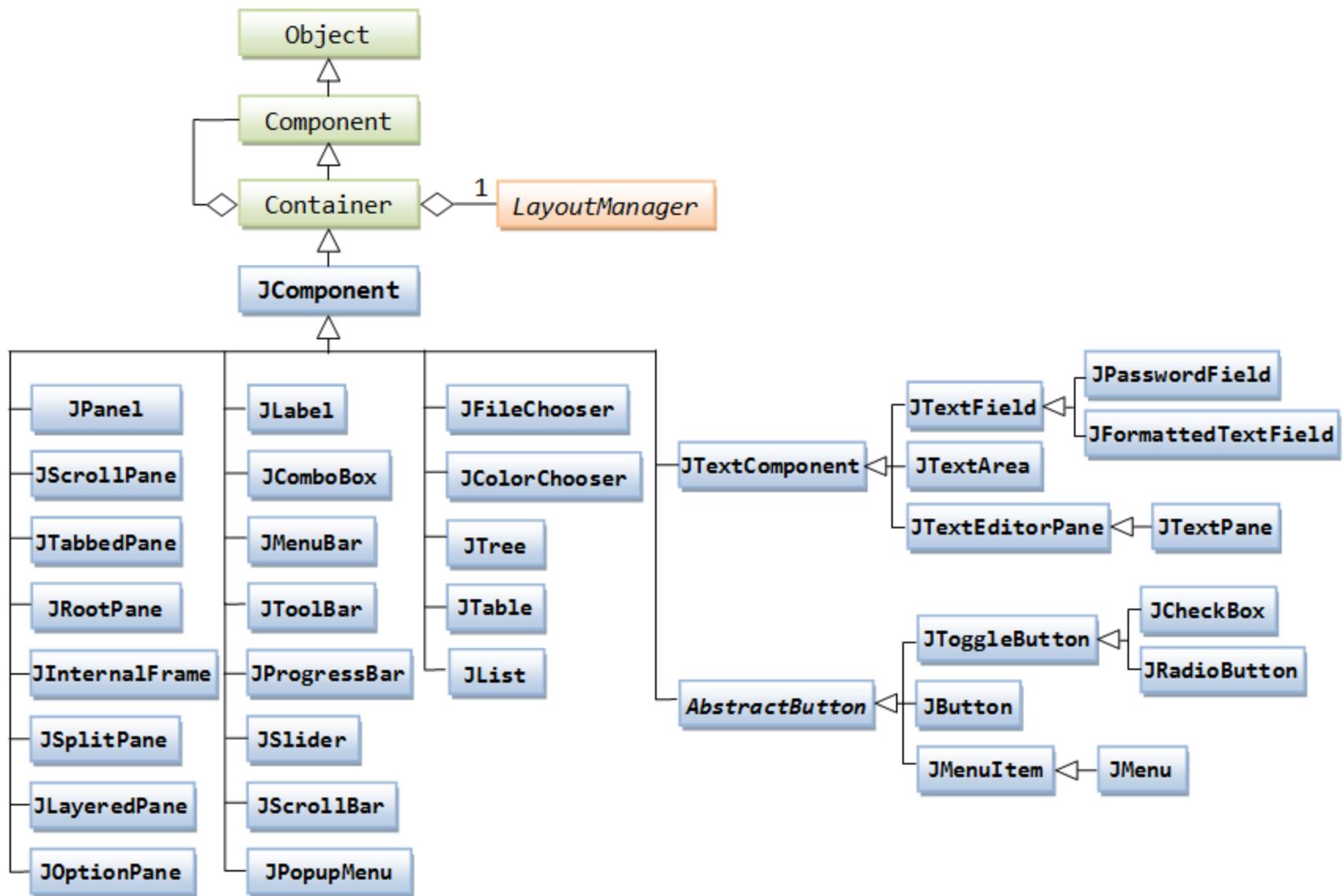
# Cấu trúc cây của (Swing)



# Các lớp Container



# Swing GUI Components



# Cơ bản về thiết kế GUI

- Khái niệm xây dựng GUI rất đơn giản. Những thành phần (**component**) được bố trí trong một bộ chứa (**container**) theo cách thức có tổ chức nào đó.
- Những **component** có thể là các đối tượng (như **Button**, **Menu**, **Label**, **TextField**, **Slider**, **Checkbox**, **Radio button**,...) hoặc có thể các bộ chứa lồng nhau,...
- Những thành phần được tổ chức trong những bộ chứa sử dụng bộ quản lý bố cục (**Layout Manager**)

# Component

- Là các đối tượng có biểu diễn đồ họa được hiển thị lên màn hình mà người dùng tương tác được
- Ví dụ: nút nhấn, checkbox, scrollbar

# Container

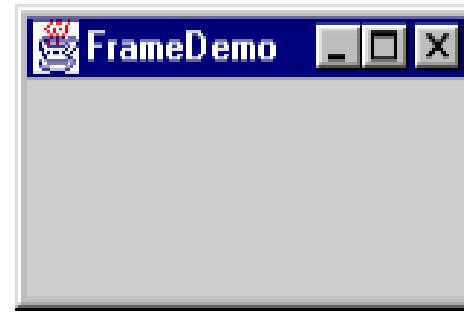
- **Frame, JFrame**
  - Là một cửa sổ Windows ở mức trên cùng gồm tiêu đề và đường biên như các ứng dụng Windows thông thường khác
  - Thường được dùng để tạo ra cửa sổ chính cho các ứng dụng khác
- **Panel**
  - Đối tượng khung chứa đơn giản nhất dùng để nhóm các đối tượng, thành phần con lại với nhau
  - Một Panel có thể chứa 1 Panel khác
- **Dialogs**
  - Là một cửa sổ dạng hộp thoại dùng để đưa ra các thông báo, lấy dữ liệu nhập từ người dùng.
- **ScrollPanes**
  - Tương tự Panel nhưng có thêm 2 thanh trượt giúp ta tổ chức và xem các đối tượng lớn
- **Applet:** Web Applet
- **JWindow:** Không có thanh tiêu đề hay các nút điều khiển.

# Top-level component

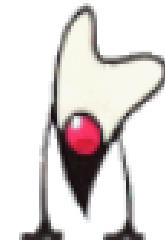
- Là thành phần trên cùng của bất kì Swing containment hierarchy nào



Dialog



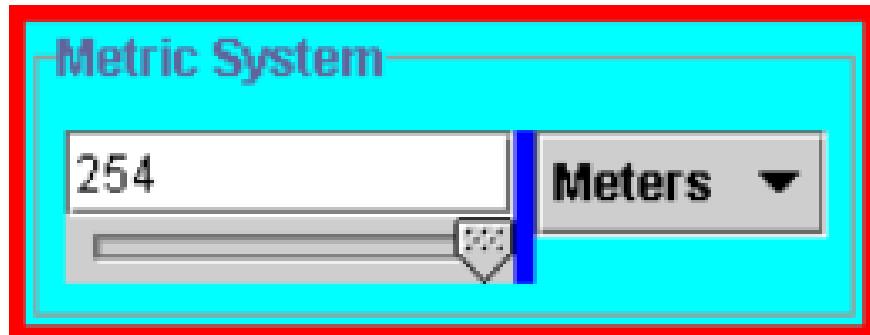
Frame



Applet

# Intermediate containers

- Là thành phần đơn thuần dùng để chứa các component khác



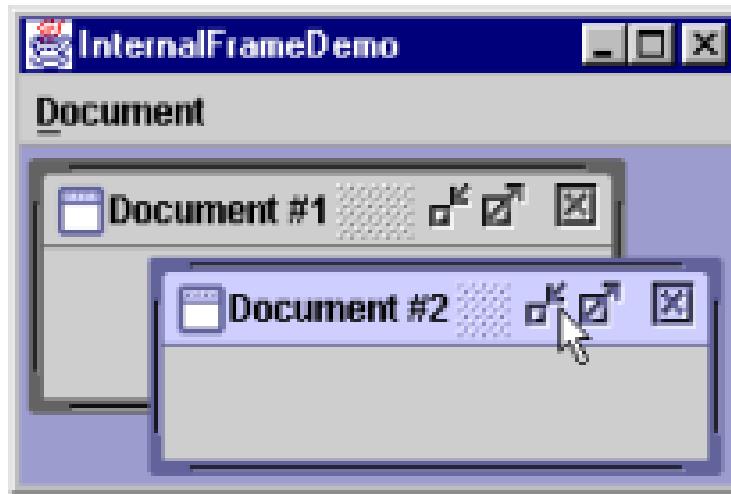
Panel



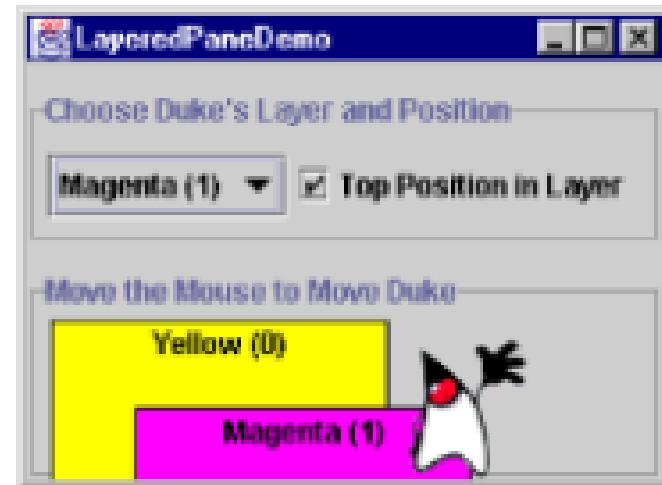
ScrollPane

# Special-Purpose Containers

- Là các thành phần chứa trung gian đặc biệt



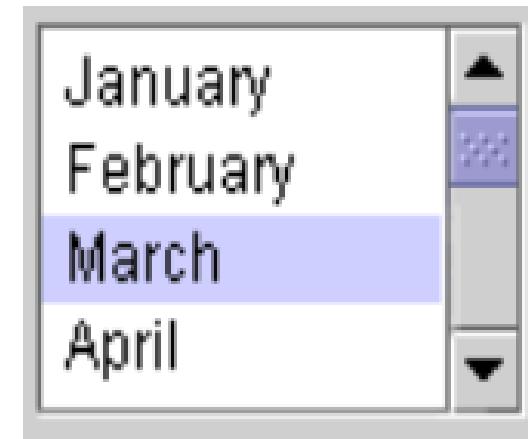
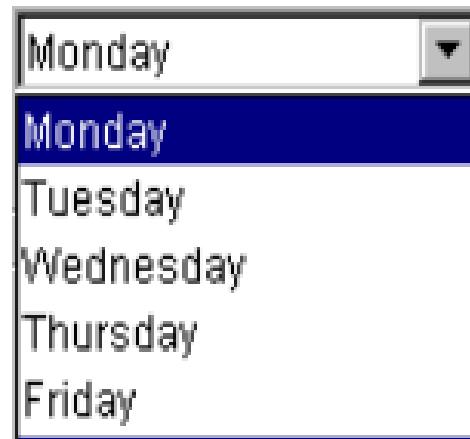
Internal Frame



Layered pane

# CÁC THÀNH PHẦN ĐIỀU KHIỂN CƠ BẢN

- Dùng để nhận dữ liệu từ người dùng



- Buttons

- **Combo Box**

- List

# CÁC THÀNH PHẦN ĐIỀU KHIỂN CƠ BẢN



Menu



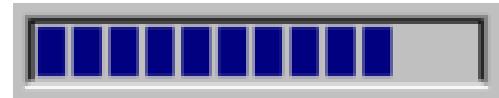
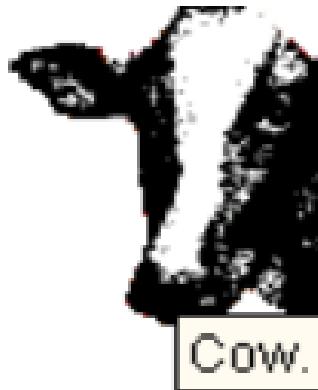
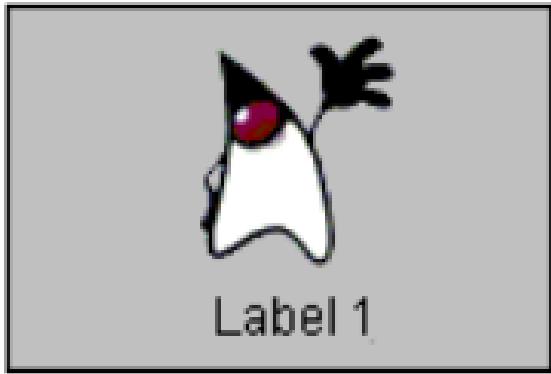
Text fields



Slide

# CÁC THÀNH PHẦN THUẦN HIỂN THỊ THÔNG TIN

- Dùng để hiển thị thông tin cho người sử dụng
- Không thể sửa đổi nội dung thông tin



Label

Tool tip

Progress Bar

# CÁC THÀNH PHẦN SỬA CHỮA ĐỊNH DẠNG

- Dùng để hiển thị các thông tin định dạng
- Cho phép người dùng lựa chọn định dạng



Color Chooser

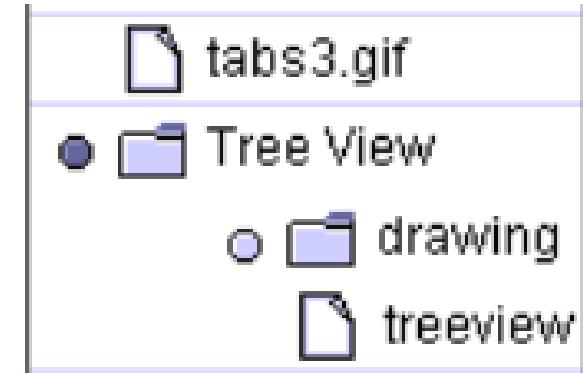


File Chooser

# CÁC THÀNH PHẦN HIỂN THỊ THÔNG TIN ĐÃ ĐỊNH DẠNG

First Name	Last Name
Mark	Andrews
Tom	Ball
Alan	Chung
Jeff	Dinkins

Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

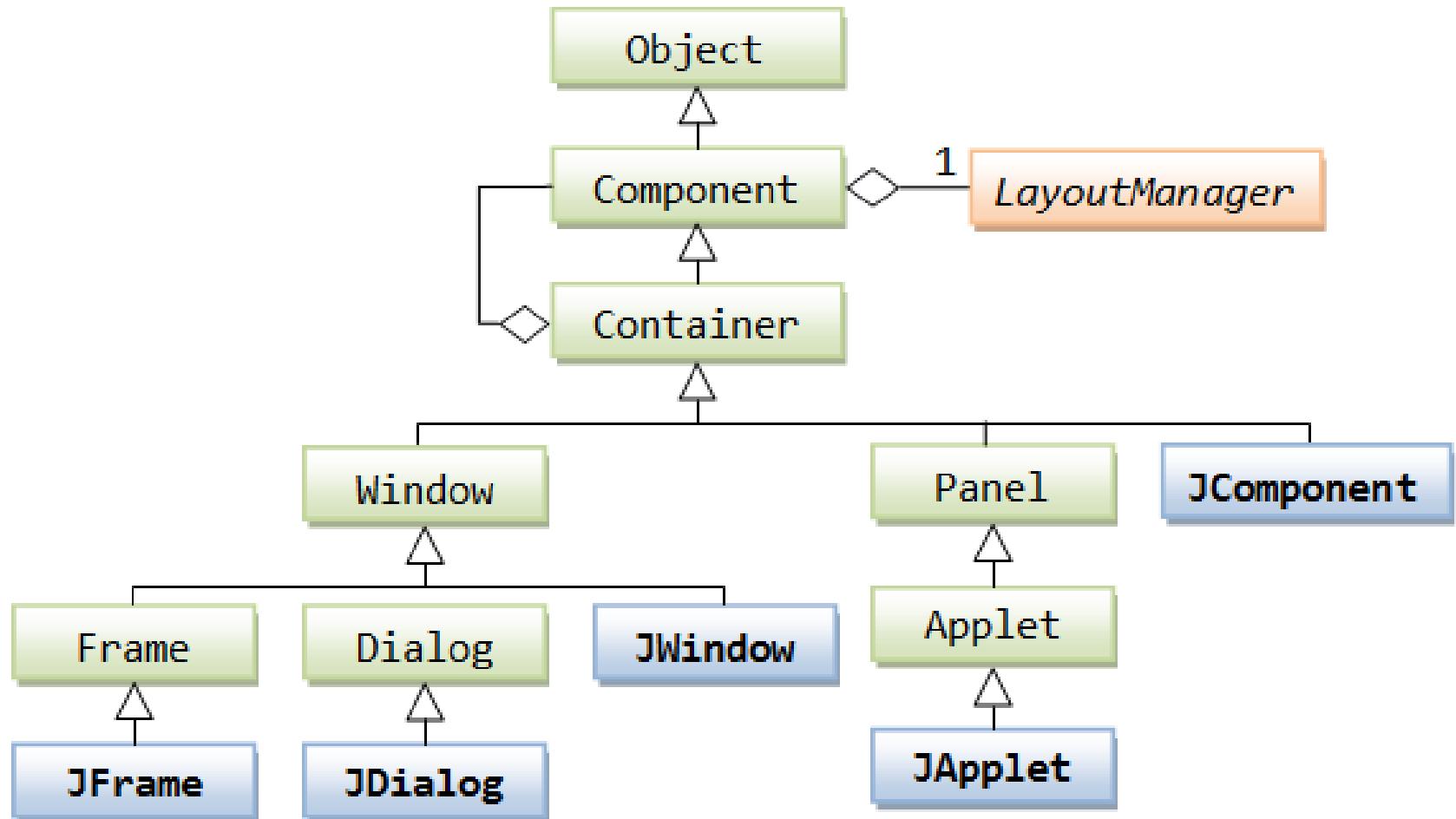


Table

Text

Tree

# Top-level container



# Jframe

```
javax.swing.JFrame

+JFrame()

+JFrame(title: String)

+void setSize(width: int, height: int)

+void setLocation(x: int, y: int)

+void setVisible(visible: boolean)

+void setDefaultCloseOperation(mode: int)

+void setLocationRelativeTo (c: Component)
```

# Ví dụ: Tạo cửa sổ với Swing

- Ứng dụng HelloWorld cơ bản
- Tạo một Cửa sổ với tiêu đề “Demo” trong đó chứa nhãn “Hello World!”
- Hiển thị cửa sổ
- Trong gói **demo**

```
1. public class HelloSwing{  
2.     public static void main(String[]  
    args){  
3.         JFrame win = new JFrame("Demo");  
4.         win.setDefaultCloseOperation(  
        JFrame.EXIT_ON_CLOSE);  
5.         win.setSize(300,200);  
6.         win.setLocationRelativeTo(null);  
7.         win.setResizable(false);  
8.         win.add(new JLabel("Hello  
world!"));  
9.         win.setVisible(true);  
10.    }  
11. }
```

```
1. public class DemoJFrame extends JFrame {  
2.     public      DemoJFrame ()      {  
3.         setTitle ("Demo JFrame") ;  
4.         setSize (300,200) ;  
5.  
       setDefaultCloseOperation (EXIT_ON_CLOSE) ;  
6.         setLocationRelativeTo (null) ;  
7.         setResizable (false) ;  
8.     }  
9.     public static void main (String [] args) {  
10.         new DemoJFrame () . setVisible (true) ;  
11.     }  
12. }
```

# Cách tạo JFrame sử dụng NetBeans

- Sử dụng project hiện có hoặc tạo mới project  
-> sử dụng dụng package hiện có hoặc tạo mới package -> chuột phải -> chọn **New** -> chọn **JFrame Form** -> nhập tên class tại **Class Name** -> chọn **Finish** để kết thúc.

Khu vực thiết kế giao diện đồ họa

Thành phần đồ họa 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"

To change layout manager of a container use Set Layout submenu from its context menu

Palette X

- Swing Containers
  - Panel
  - Tabbed Pane
  - Split Pane
  - ScrollPane
  - ToolBar
  - Desktop Pane
  - Internal Frame
  - Layered Pane
- Swing Controls
  - Label
  - Button
  - ToggleButton
  - Check Box
  - Radio Button
  - Button Group
  - ComboBox
  - List
  - TextField
  - TextArea
  - ScrollBar
  - Slider
  - ProgressBar
  - Formatted Field
  - Password Field
  - Spinner
  - Separator
  - TextPane
  - Editor Pane
  - Tree
  - Table
- Swing Menus
- Custom Windows

[JFrame] - Properties X

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"/>		

# Thuộc tính thường dùng với thành phần JFrame

## Thuộc tính

title

defaultCloseOperation

iconImage

resizable

size

## Miêu tả

Tiêu đề màn hình

Thiết lập xử lý khi người dùng chọn nút dấu X ở góc trên bên phải

Thiết lập icon ở góc trên bên trái của màn hình

Cho phép điều chỉnh kích thước màn hình hay không

Kích thước màn hình

# JDialog - demo

```
1. public class DemoJDialog extends JDialog{  
2.     public DemoJDialog() {  
3.         setTitle("Demo JDialog");  
4.         setDefaultCloseOperation(  
        DISPOSE_ON_CLOSE);  
5.         setSize(300,200);  
6.         setResizable(false);  
7.     }  
8.     public static void main(String[] args) {  
9.         new DemoJDialog().setVisible(true);  
10.    }  
11. }
```

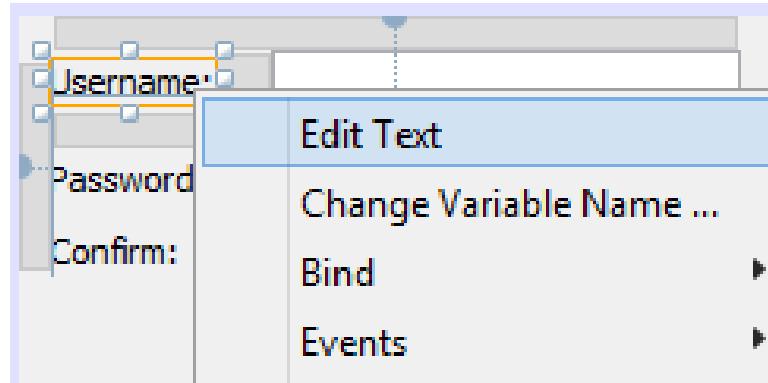
# JPanel

- 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.
- Constructors:
  - JPanel()
  - JPanel(LayoutManager lm)

```
1. public class DemoJPanel extends JFrame {  
2.     public DemoJPanel() {  
3.         setTitle("Demo JPanel");  
4.         setSize(300, 400);  
5.         setDefaultCloseOperation(  
        JFrame.EXIT_ON_CLOSE);  
6.         JPanel p=new JPanel();  
7.         p.setBorder(BorderFactory.  
createTitledBorder("Rong tin sach"));  
8.         p.add(new JLabel("Thu co nhan!"));  
9.         p.add(new JButton("An toi"));  
10.        this.add(p);  
11.    }  
12.    public static void main(String[] args) {  
13.        new DemoJPanel().setVisible(true);  
14.    }  
15.}
```

# JLabel (1)

- 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 và phương thức String label = **getText()** dùng để lấy nội dung của Jlabel
- Font **getFont()**
- void **setFont(Font font)**
  - Gets or sets font hiện thời của nhãn



# Jlabel (2)

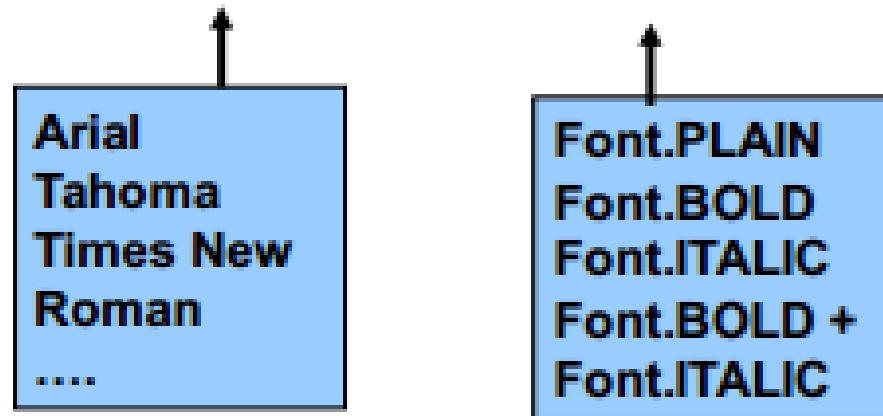
- Constructors:
- **JLabel()**: Creates an empty label
- **JLabel (String labelText)**: Creates a label with a given text
- **JLabel (String labelText, int alignment)**: Creates a label with given alignment where alignment can be LEFT, RIGHT, CENTER, LEADING or TRAILING.
- **JLabel (Icon img)**: Only Icon will be used for label
- **JLabel (String str, Icon img, int align)**

```
1. public class DemoJLabel extends JFrame{  
2.     public DemoJLabel () {  
3.         setLayout(new GridLayout(1,3,5,5)) ;  
4.         setDefaultCloseOperation(  
            JFrame.EXIT_ON_CLOSE) ;  
5.         Icon icon = new  
            ImageIcon("pic_8.jpg") ;  
6.         JLabel lbl = new  
            JLabel("Nhan la text") ;  
7.         JLabel lb2 = new JLabel(icon) ;  
8.         JLabel lb3 = new JLabel("icon va  
            text",icon, JLabel.CENTER) ;  
9.         lb3.setVerticalTextPosition(  
            JLabel.BOTTOM) ;
```

```
1.     lb3.setHorizontalAlignment(        
           JLabel.CENTER);  
2.     add(lb1);  
3.     add(lb2);  
4.     add(lb3);  
5.     pack();  
6.     setLocationRelativeTo(null);  
7. }  
8. public static void main(String[]  
    args) {  
9.     new  
        DemoJLabel().setVisible(true);  
10. }  
11. }
```

# fonts for text

- To draw characters in a font, you must first create an object of the class Font
- Constructor:
- `Font( String font_name, int font_style, int font_size )`

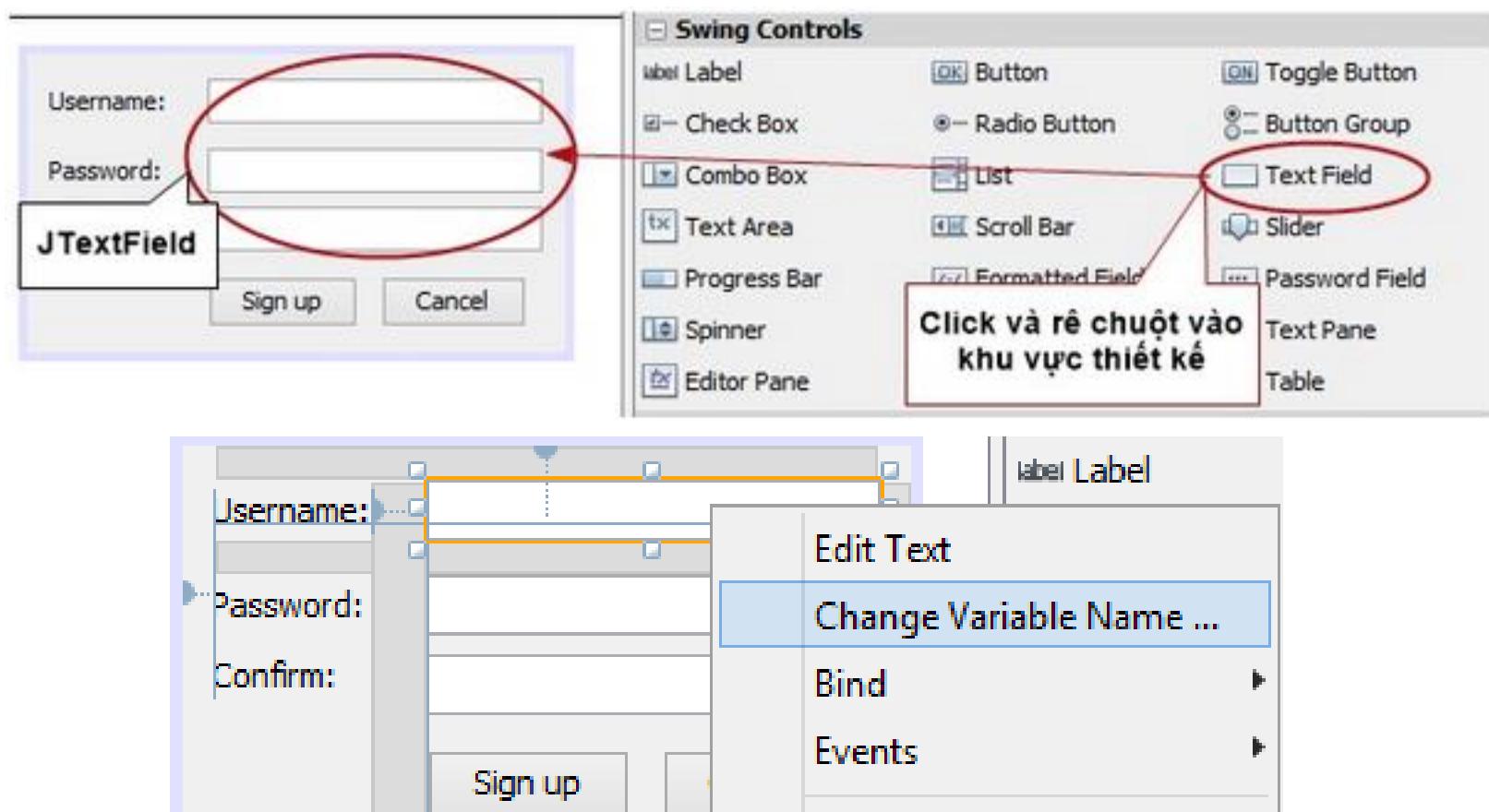


```
1. public class DemoJLabelwithColor extends JFrame{  
2.     public DemoJLabelwithColor() {  
3.         setLayout(new GridLayout(1, 2, 5, 5));  
4.         setDefaultCloseOperation(  
                JFrame.EXIT_ON_CLOSE);  
5.         setSize(400, 200);  
6.         JLabel lb;  
7.         lb = createJLabel("Vi du 1", Color.red,  
                            Color.green);  
8.         Font font=new Font("Arial",Font.BOLD,40);  
9.         lb.setFont(font);  
10.        add(lb);  
11.        lb = createJLabel("Vi du 1", Color.blue,  
                            Color.yellow);  
12.        lb.setFont(font);  
13.        add(lb);  
14.        setLocationRelativeTo(null); }
```

```
1. private JLabel createJLabel(String text,  
    Color textColor,Color backgroundColor) {  
2.     JLabel lb = new JLabel(text);  
3.     lb.setHorizontalAlignment(  
        JLabel.CENTER);  
1.     lb.setForeground(textColor);  
2.     lb.setOpaque(true);  
3.     lb.setBackground(backgroundColor);  
4.     return lb;  
5. }  
6. public static void main(String[] args) {  
7.     new  
        DemoJLabelwithColor().setVisible(true);  
8. }  
9. }
```

# JTextField

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



# JTextField - Constructors

- JTextField()
  - creates an empty textfield with 1 columns
- TextField(String s)
  - creates a new textfield with the given string
- JTextField(int cols)
  - creates an empty textfield with given number of columns
- JTextField(String text, int cols)
  - creates a new textfield with given string and given number of columns
- Example:
  - JTextField mmText = new JTextField(10);
  - JTextField txtName = new JTextField("To Lan", 20);

# JTextField - Methods

- `String getText()`
- `void setText(String t)`
  - gets or sets the text in text field
- `void setFont(Font font)`
  - sets the font for this text field
- `void setEditable(boolean b)`
  - determines whether the user can edit the content

```
1. public class DemoJTextField extends JFrame {  
2.     JTextField ten;  
3.     JPasswordField matkhau;  
4.     public DemoJTextField() {  
5.         super("Vi du nhap lieu");  
6.         JPanel p =new JPanel();  
7.         p.add(new JLabel("Ten: "));  
8.         p.add(new JTextField(15));  
9.         p.add(new JLabel("Mat khau:"));  
10.        p.add(new JPasswordField(15));  
11.        add(p);  
12.        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
13.    }  
14.    public static void main(String[] args) {  
15.        new DemoJTextField().setVisible(true);  
16.    }  
17.}
```

# JPasswordField

- JPasswordField tương tự như JTextField ngoại trừ nội dung trong JPasswordField sẽ chuyển thành dấu hoa thị (\*).
- setEchoChar(): Thiết lập ký tự hiển thị, mặc định là chấm

# JTextArea

- JTextArea cho phép nhập và chỉnh sửa nhiều dòng văn bản.
- String text = getText();
- setText(String value);
- setEditable(boolean editable)

# JButton

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

# Một số phương thức của lớp AbstractButton

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

```
1. public class DemoJButton extends JFrame {  
2.     JButton b1,b2;  
3.     public DemoJButton () {  
4.         super ("Vi du Button");  
5.         b1= new JButton ("Dung lai", new  
ImageIcon ("stop.png"));  
6.         b2= new JButton ("Thuc hien", new  
ImageIcon ("go.png"));  
7.         JPanel p = new JPanel();  
8.         p.add (b1);  
9.         p.add (b2);  
10.        add (p);  
11.        setSize (300,200);  
12.    }  
13.    public static void main (String[] args) {  
14.        new DemoJButton ().setVisible (true);  
15.    }  
16.}
```

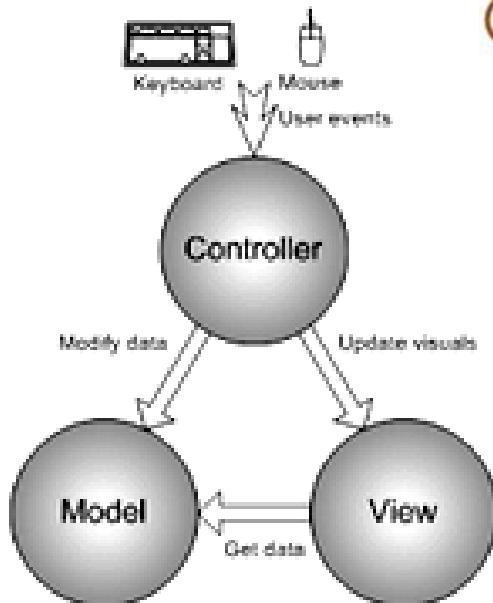
# Event (Sự kiện)

- Sự kiện có thể được định nghĩa như là một kiểu tín hiệu tới chương trình thông báo một việc gì đó đã xảy ra.
- Sự kiện được sinh ra bởi các hành động của người dùng, ví dụ như di chuyển, click chuột, hay nhấn phím....
- Ứng dụng cần đăng ký một hàm xử lý sự kiện với một đối tượng. Hàm xử lý sự kiện này sẽ được gọi bất cứ khi nào sự kiện tương ứng phát sinh.
- Trong quy trình này, ứng dụng cho phép bạn đăng ký các phương thức (handler), hay gọi là listener với các đối tượng. Những handler này tự động được gọi khi một sự kiện thích hợp phát sinh.
- Một Event Listener lắng nghe một sự kiện nào đó mà một đối tượng đã thiết lập. Mỗi event listener cung cấp các phương thức xử lý những sự kiện này.

# Two Kinds of Apps

Console App.  
Compute-centric  
Apps

Event-based App.  
User-centric  
Apps(GUI)

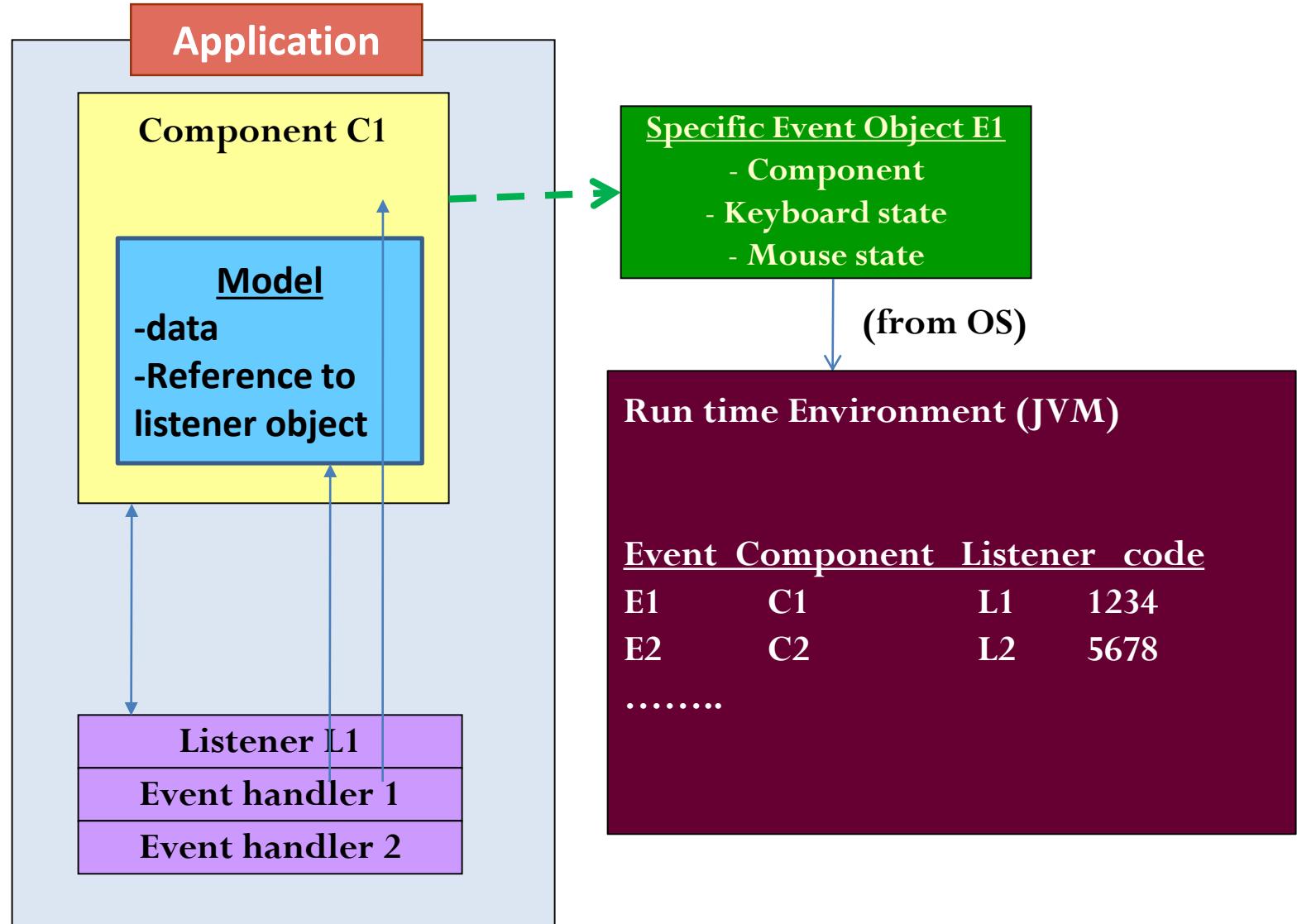


A screenshot of a Windows Command Prompt window titled "C:\WINDOWS\system32\cmd.exe". The command entered is "G:\GiangDay\FU\CoreJava\Chapter01\build\classes>java myPackage.Ex01.8.3". The output shows "G:\GiangDay\FU\CoreJava\Chapter01\build\classes>pause" followed by "Press any key to continue . . .".



**Model-View Controller Architecture for GUI component.**  
Model: Object contains data.  
View: Object users can see it on the screen  
Controller: Object manages events

# Java model for event management



# Cài đặt

- Các bước như sau:
- Cài đặt giao diện listener thích hợp. Cấu trúc như sau:

```
public class MyApp extends Frame  
implements ActionListener
```
- Xác định tất cả các thành phần tạo ra sự kiện. Các thành phần có thể là các button, label, menu item, hay window. Cho ví dụ, để đăng ký một thành phần với listener, ta có thể sử dụng:

```
JButton exitbtn=new JButton("Thoat");  
exitbtn.addActionListener(This);
```

# ActionListener interface

```
1. public interface ActionListener {  
2.     public void  
         actionPerformed(ActionEvent event);  
3. }
```

- ActionEvent có hai phương thức:
- **getSource()**: Để trả về nguồn của sự kiện.
- **toString()**: Để trả về chuỗi tương đương với sự kiện.

```
1. public class ButtonHandlingDemo
   implements ActionListener {
2. ....
3. btnResult.addActionListener(this);
1. ....
2. public void
   actionPerformed(ActionEvent ae) {
3. ...
4. if(ae.getSource() == btnResult)
5.     ....
6. }
7. }
```

# Ví dụ: gói event

```
public class EventExp extends JFrame  
    implements ActionListener {  
  
    JLabel lbr=new JLabel("Ban kinh ");  
    JTextField tfr=new JTextField(1);  
    JLabel lbrs=new JLabel("Dien tich ");  
    JTextField tfrs=new JTextField();  
    JButton btnCal=new JButton("Tinh");  
    JButton btnExit=new JButton("Thoat");  
  
    public EventExp(String title) {  
        super(title);  
        setLayout(new GridLayout(3, 2));  
    }  
}
```

```
        add(lbr);      add(tfr);
        add(lbtrs);    add(tfrs);
        btnCal.addActionListener(this);
        btnExit.addActionListener(this);
        add(btnCal);   add(btnExit);

    }

public static void main(String[] args) {
    EventExp f=new EventExp("Demo for
event");
    f.setDefaultCloseOperation(JFrame.EXIT_ON
_CLOSE);
    f.setSize(300, 200);
    f.setVisible(true);

}
```

```
@Override  
    public void  
actionPerformed(ActionEvent ae) {  
    if (ae.getSource() == btnCal) {  
        double  
r=Double.parseDouble(tfr.getText());  
        double s=r*r*Math.PI;  
        DecimalFormat f=new  
DecimalFormat("#.##");  
        tfrs.setText(f.format(s));  
    } else if (ae.getSource() == btnExit)  
        System.exit(0);  
}
```

# Some Common Events

<i>Object</i>	<i>Event</i>	<i>Interface</i>	<i>Method</i>
JButton	ActionEvent	ActionListener	actionPerformed()
JCheckBox	ActionEvent	ActionListener	actionPerformed()
	ItemEvent	ItemListener	itemStateChanged()
JRadioButton	ActionEvent	ActionListener	actionPerformed()
	ItemEvent	ItemListener	itemStateChanged()
JTextField JTextArea	ActionEvent FocusEvent	ActionListener FocusListener	actionPerformed() focusGained(), focusLost()
JPasswordField	ActionEvent	ActionListener	actionPerformed()



# Layout Manager (1)

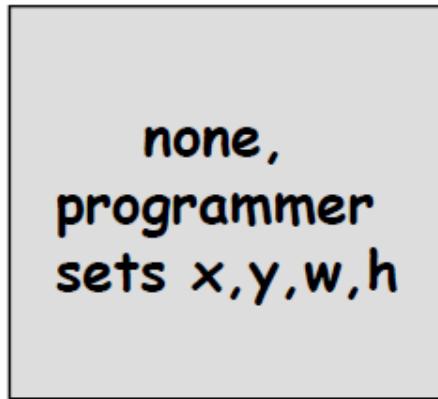
- ✓ Flow Layout
- ✓ Border Layout
- ✓ Card Layout
- ✓ GridLayout
- ✓ GridBagConstraints Layout
- ✓ Box Layout
- ✓ Overlay Layout

Defined in the AWT

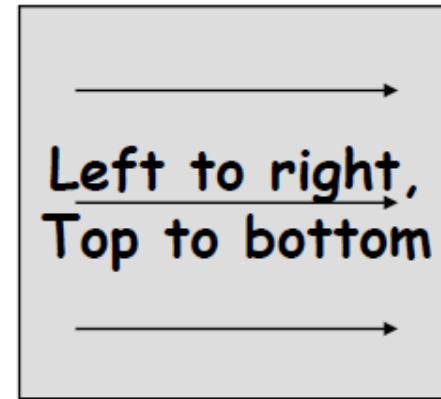
Defined in Swing

# Layout Manager (2)

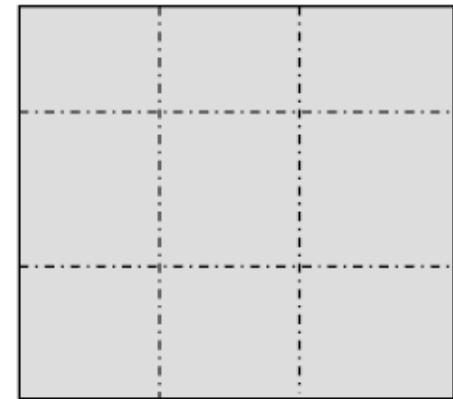
null



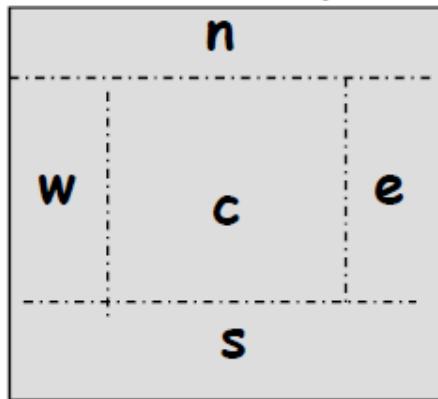
FlowLayout



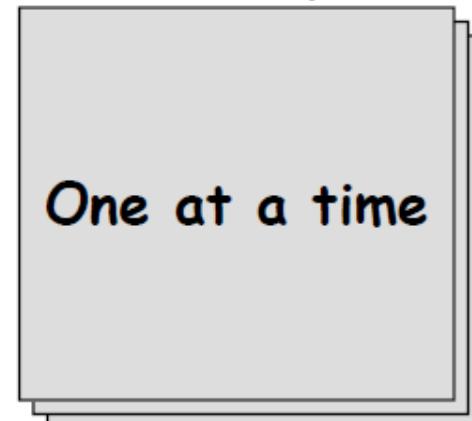
GridLayout



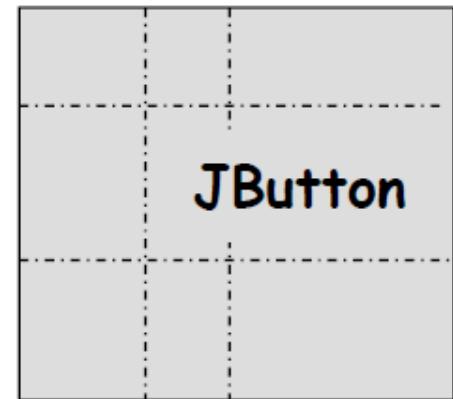
BorderLayout



CardLayout



GridBagLayout



# **FlowLayout**

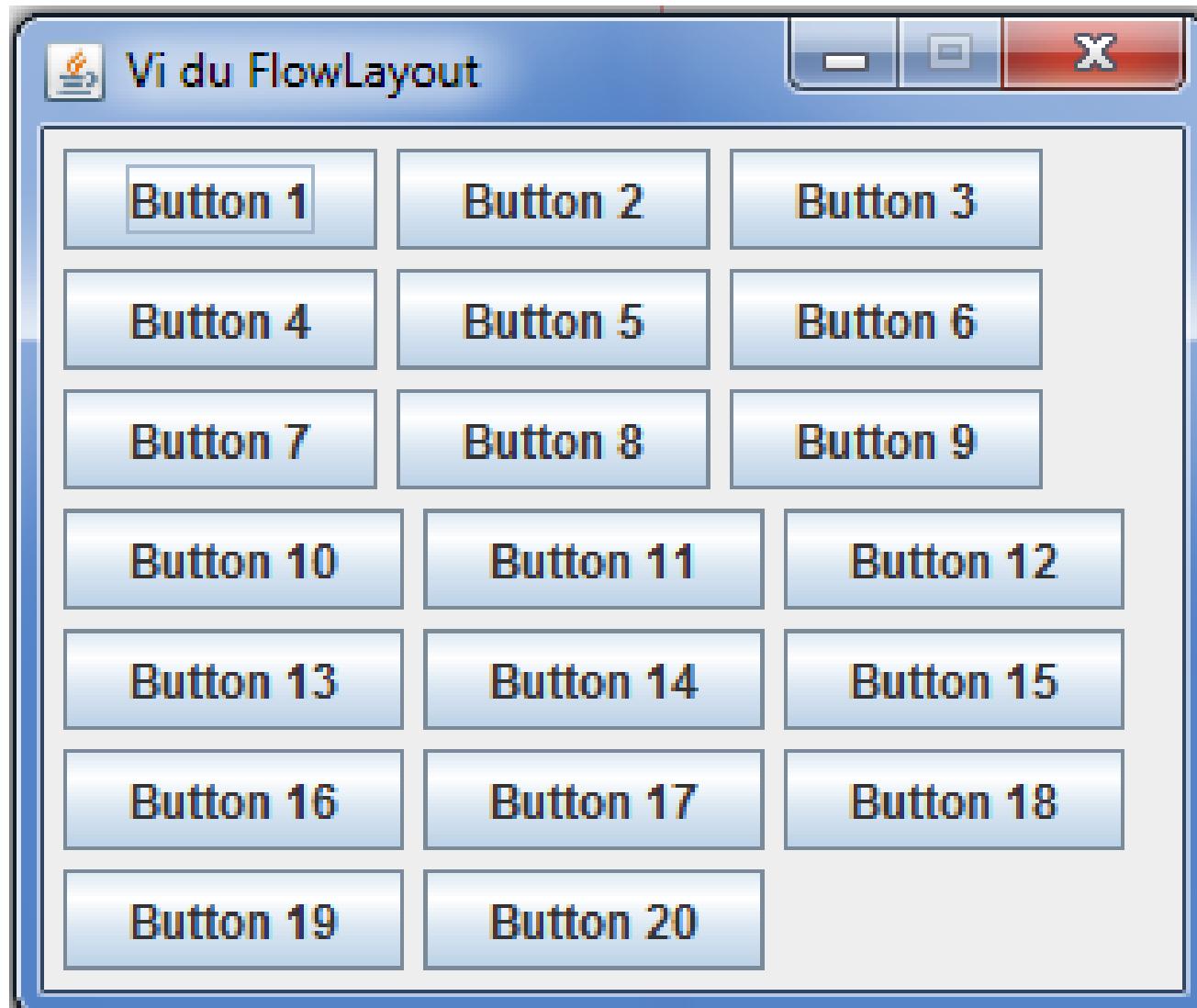
- Mặc định khi một JPanel được khởi tạo thì bản thân lớp chứa này sẽ có kiểu Layout là FlowLayout.

# FlowLayout – Constructors

- `public FlowLayout()`
  - Centers each row and keeps 5 pixels between entries in a row and between rows
- `public FlowLayout(int align)`
  - Same 5 pixels spacing, but changes the alignment of the rows to `FlowLayout.LEFT`, `FlowLayout.RIGHT`, `FlowLayout.CENTER`
- `public FlowLayout(int align, int hgap, int vgap)`
  - Specify the alignment as well as the horizontal and vertical spacing between components (in pixel)

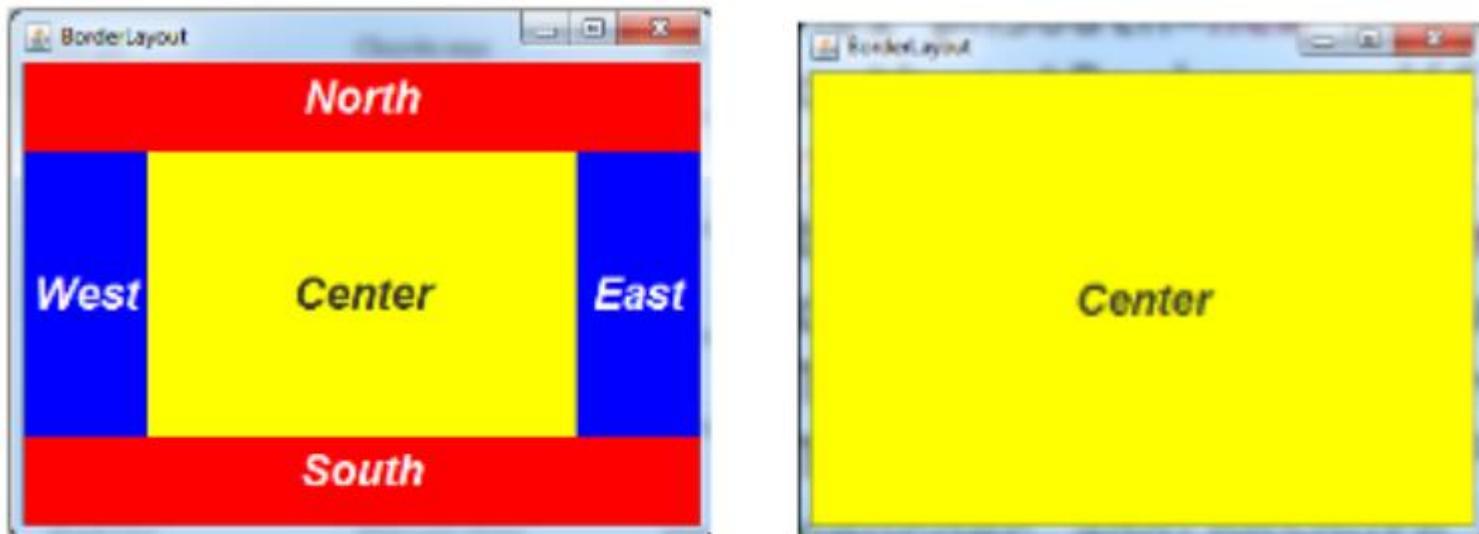
```
1. public class DemoFlowLayout extends JFrame{  
2.     public DemoFlowLayout() {  
3.         setTitle("Vi du FlowLayout");  
4.         setSize(300,250);  
5.         setDefaultCloseOperation(EXIT_ON_CLOSE);  
6.         setLocationRelativeTo(null);  
7.         setLayout(new FlowLayout(FlowLayout.LEFT));  
8.         for(int i=1;i<=20;i++){  
9.             add(new JButton("Button "+i));  
10.        }  
11.    }  
12.    public static void main(String[] args) {  
13.        new DemoFlowLayout().setVisible(true);  
14.    }  
15.}
```

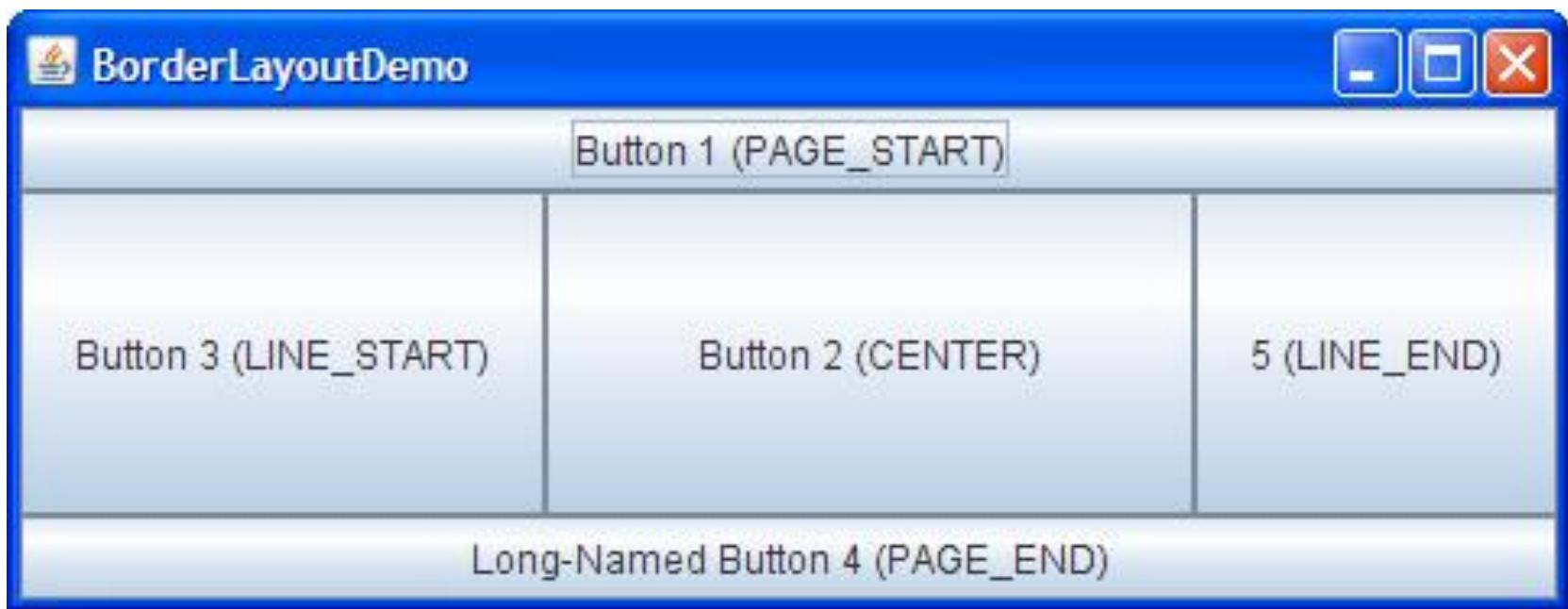
# Kết quả



# Border Layout

- 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.

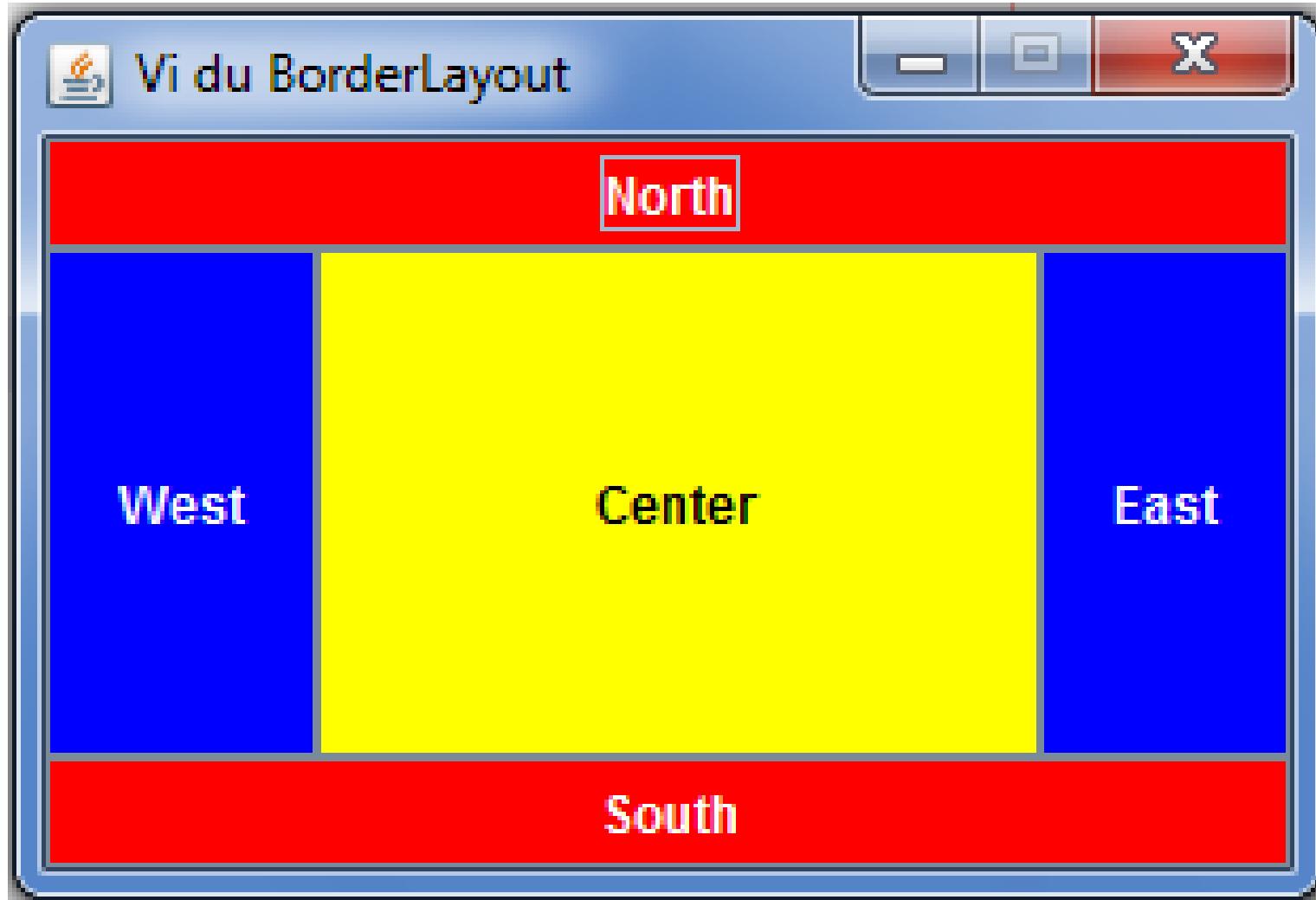




```
1. public class DemoBorderLayout extends JFrame{  
2.     private JButton  
3.     bn=new JButton("North"),  
4.     bs=new JButton("South"),  
5.     be=new JButton("East"),  
6.     bw=new JButton("West"),  
7.     bc=new JButton("Center");  
8.     public DemoBorderLayout() {  
9.         setTitle("BorderLayout");  
10.        setSize(300,200);  
11.        setDefaultCloseOperation(EXIT_ON_CLOSE);  
12.        setLocationRelativeTo(null);  
13.        setResizable(false);  
14.        add(BorderLayout.NORTH, bn);  
15.        add(BorderLayout.SOUTH, bs);  
16.        add(BorderLayout.EAST, be);  
17.        add(BorderLayout.WEST, bw);  
18.        add(BorderLayout.CENTER, bc);  
19.    }  
20.}
```

```
1. bn.setBackground(Color.red);
2.         bs.setBackground(Color.red);
3.         bc.setBackground(Color.YELLOW);
4.         be.setBackground(Color.BLUE);
5.         bw.setBackground(Color.BLUE);
6.         bn.setForeground(Color.WHITE);
7.         bs.setForeground(Color.WHITE);
8.         bc.setForeground(Color.black);
9.         be.setForeground(Color.WHITE);
10.        bw.setForeground(Color.WHITE);
11.    }
12. public static void main(String[] args) {
13. new DemoBorderLayout().setVisible(true);
14.    }
15. }
```

# Kết quả



# Grid Layout

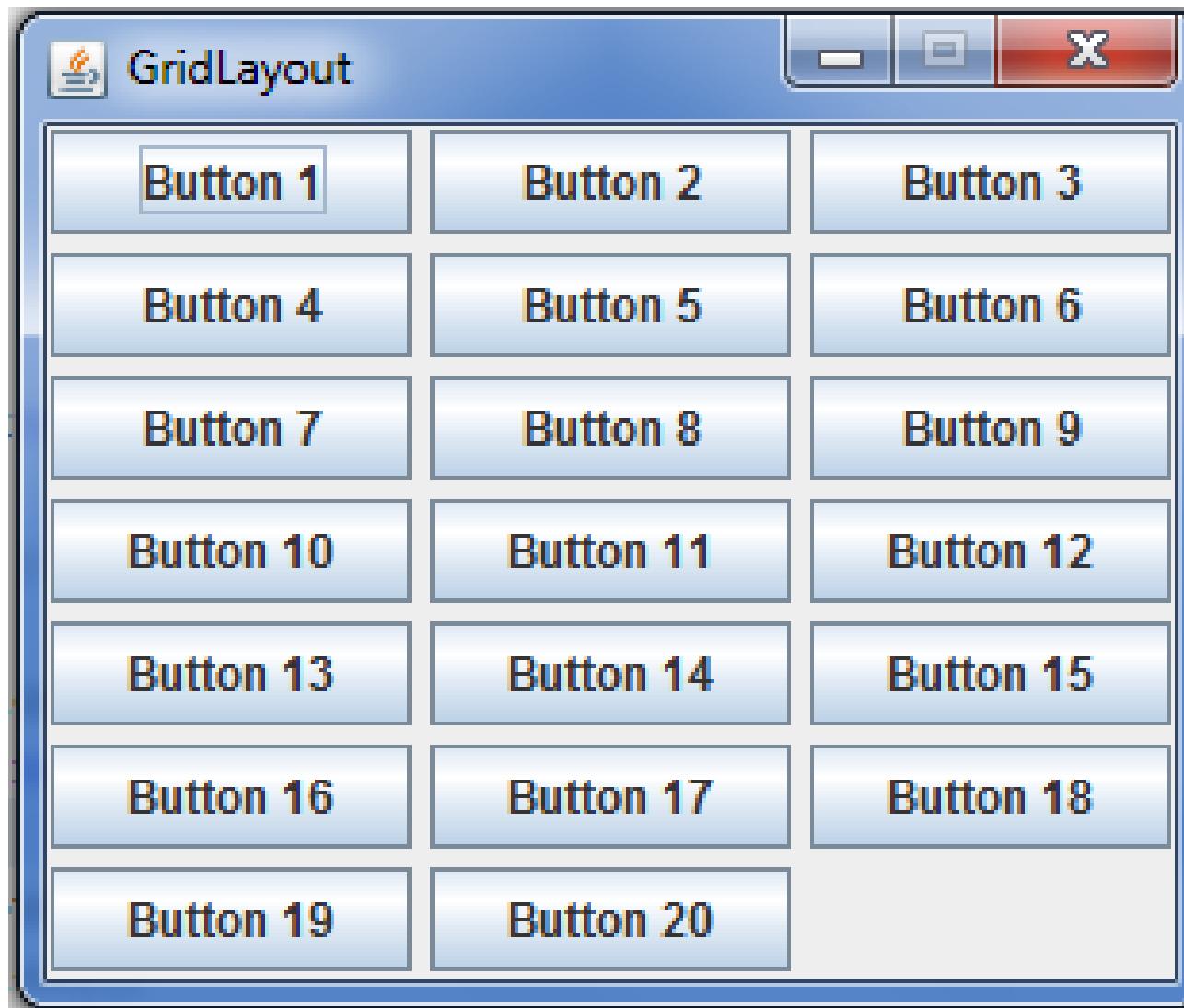
- Hỗ trợ việc chia container thành một lưới
- Các thành phần được bố trí trong các dòng và cột
- Một ô lưới nên chứa ít nhất một thành phần
- Kiểu layout này được sử dụng khi tất cả các thành phần có cùng kích thước

# GridLayout – Constructors

- `public GridLayout()`
  - Creates a single row with one column allocated per component
- `public GridLayout(int rows, int cols)`
  - Divides the window into the specified number of rows and columns
  - Either rows or cols (but not both) can be zero
- `public GridLayout(int rows, int cols, int hgap, int vgap)`
  - Uses the specified gaps between cells

```
1. public class DemoGridLayout extends  
   JFrame{  
2.   public DemoGridLayout() {  
3.     setTitle("GridLayout");  
4.     setSize(300, 250);  
5.  
6.     setDefaultCloseOperation(EXIT_ON_CLOSE);  
7.     setLocationRelativeTo(null);  
8.     setLayout(new GridLayout(7,3,5,5));  
9.     for(int i = 1;i <=20;i++) {  
10.       add(new JButton("Button "+i));  
11.     }  
12.   }  
13.   public static void main(String[] args) {  
14.     new DemoGridLayout().setVisible(true);  
15.   }  
16. }
```

# Kết quả

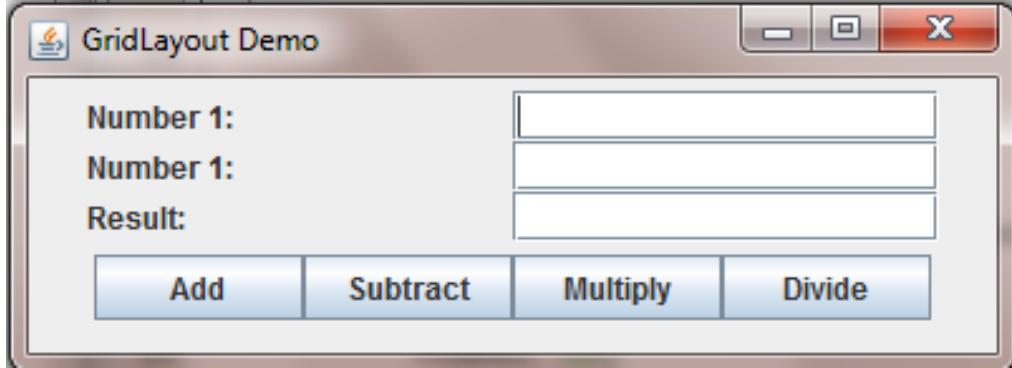


```
1. public class DemoGridLayout1 extends JFrame {  
2.     public DemoGridLayout1() {  
3.         initUI(); }  
4.     public final void initUI() {  
5.         JPanel panel = new JPanel();  
6.  
    panel.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));  
7.         panel.setLayout(new GridLayout(5, 4, 5,  
5));  
8.         String[] buttons = {  
9.             "Cls", "Bck", "", "Close",  
10.            "7", "8", "9", "/",  
11.            "4", "5", "6", "*",  
12.            "1", "2", "3", "-",  
13.            "0", ".", "=", "+"  
14.        };  
}
```

```
1. for (int i = 0; i <
       buttons.length; i++) {
2.   if (i == 2)
3.     panel.add(new
       JLabel(buttons[i]));
4.   else
5.     panel.add(new
       JButton(buttons[i]));
6.   }
7. add(panel);
8. setTitle("Vi du GridLayout");
9. setSize(350, 300);
```

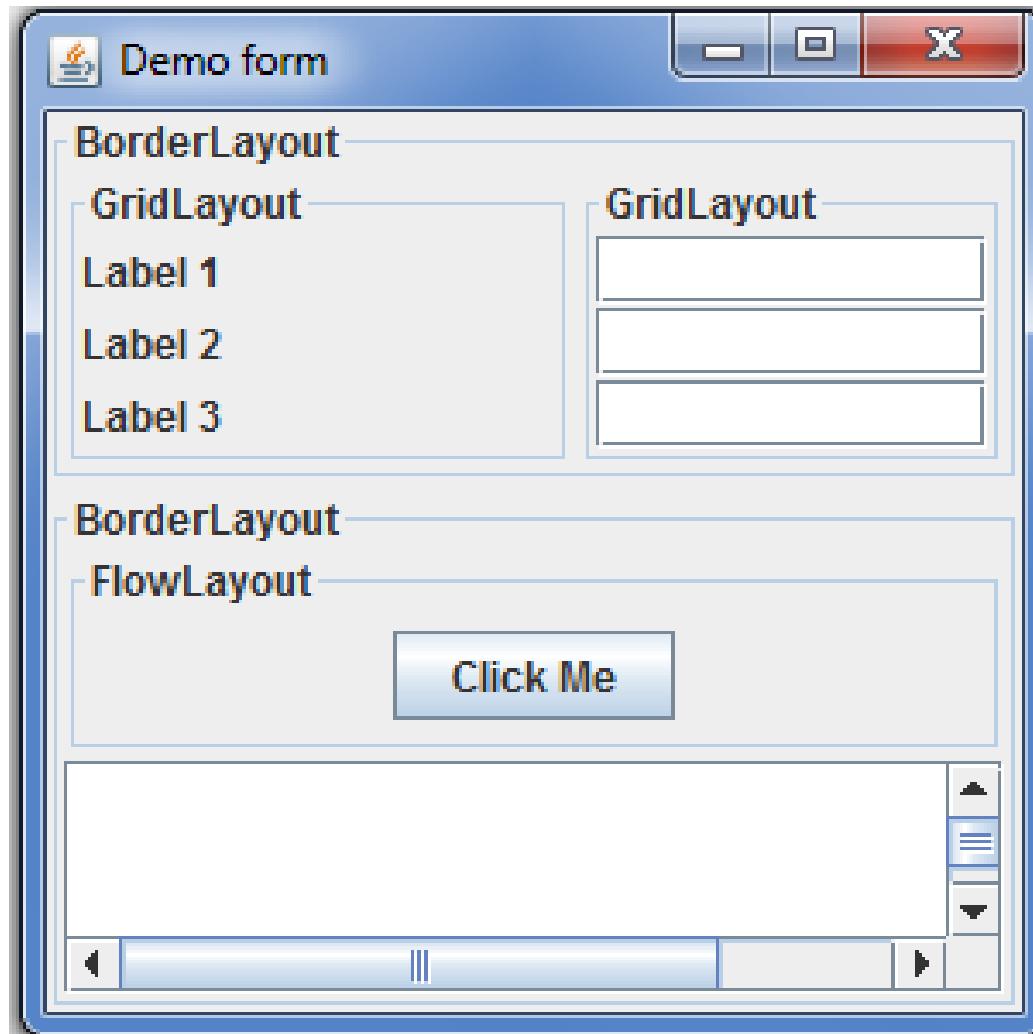
```
1.     setDefaultCloseOperation(JFrame.EXIT_ON  
      _CLOSE);  
2.     setLocationRelativeTo(null);  
3. }  
4. public static void main(String[] args)  
{  
5.     DemoGridLayout1 ex = new  
    DemoGridLayout1();  
6.     ex.setVisible(true);  
7. }  
8. }
```

# Combinations (GridLayout)



- JPanel **p1**=new JPanel();  
**p1.setLayout(new GridLayout(3,2));**
- JPanel **p2**=new JPanel();  
**p2.setLayout(new GridLayout(1,4));**
- JPanel **main**=new JPanel();  
    **main.add(p1, BorderLayout.NORTH);**  
    **main.add(p2, BorderLayout.SOUTH);**  
    **this.setContentPane(main); //add(main);**

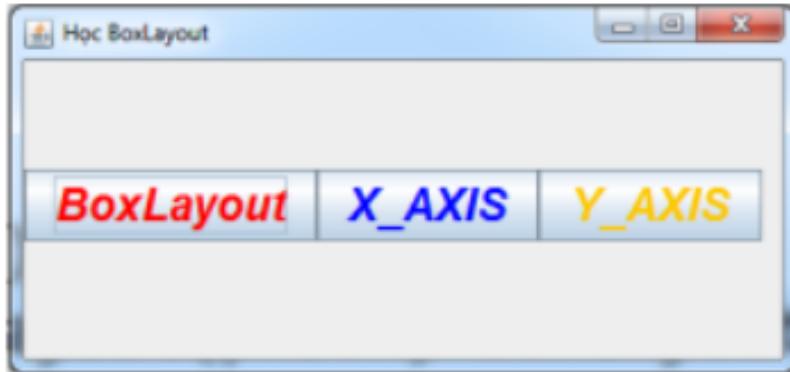
# Ví dụ: DemoBorder.java



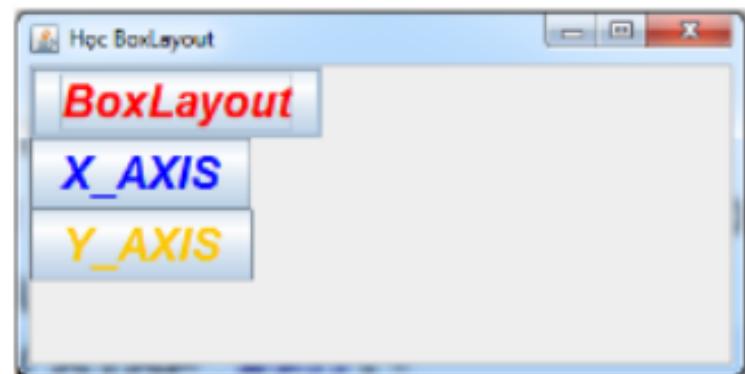
# Box Layout

- BoxLayout cho phép add các control theo dòng hoặc cột, tại mỗi vị trí add nó chỉ chấp nhận 1 control, do đó muốn xuất hiện nhiều control tại một vị trí thì bạn nên add vị trí đó là 1 JPanel rồi sau đó add các control khác vào JPanel này .
- BoxLayout.X\_AXIS cho phép add các control theo hướng từ trái qua phải.
  - Box box = new Box(BoxLayout.X\_AXIS);
  - Box box = Box.createHorizontalBox();
- BoxLayout.Y\_AXIS cho phép add các control theo hướng từ trên xuống dưới.
  - Box box = new Box(BoxLayout.Y\_AXIS);
  - Box box = Box.createVerticalBox();
- BoxLayout sẽ không tự động xuống dòng khi hết chỗ chứa, tức là các control sẽ bị che khuất nếu như thiếu không gian chứa nó.

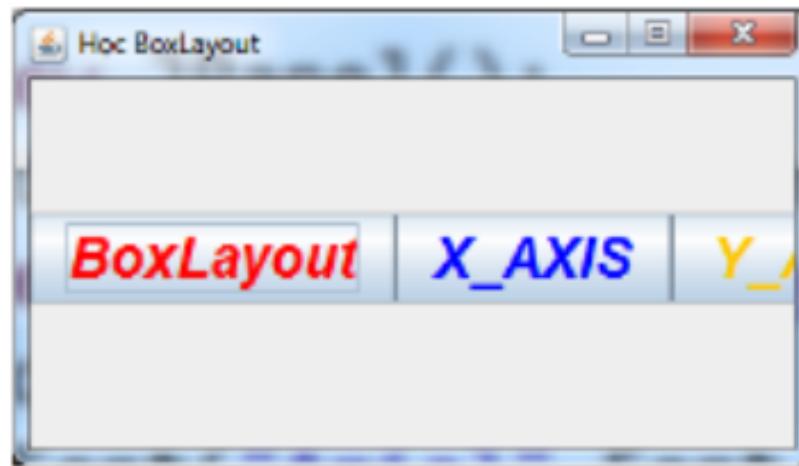
# Ví dụ Box Layout



BoxLayout.X\_AXIS



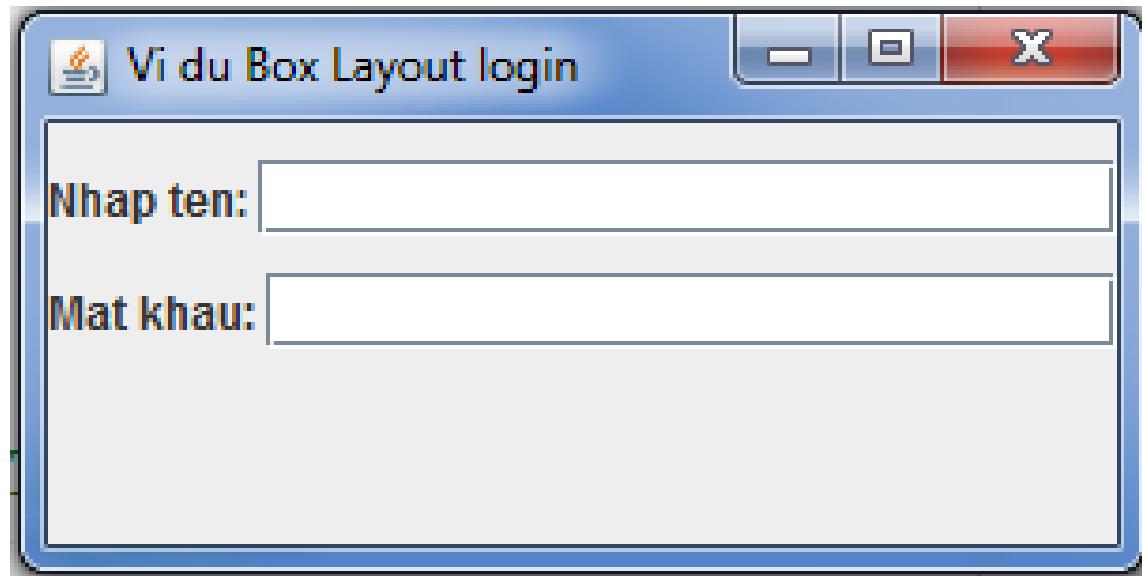
BoxLayout.Y\_AXIS



No wrap row when resize dimension

```
1. public class BoxLayout1 extends JFrame{  
2.     public BoxLayout1() {  
3.         setTitle("Vi du Box Layout login");  
4.         setSize(300,150);  
5.         setDefaultCloseOperation(EXIT_ON_CLOSE);  
6.         JPanel p = new JPanel();  
7.         JPanel p1 = new JPanel();  
8.         JPanel p2 = new JPanel();  
9.         p.setLayout(new BoxLayout(p,  
        BoxLayout.Y_AXIS));  
10.        p1.setLayout(new BoxLayout(p1,  
        BoxLayout.X_AXIS));  
11.        p2.setLayout(new BoxLayout(p2,  
        BoxLayout.X_AXIS));  
12.        p1.add(new JLabel("Nhap ten: "));  
13.        p1.add(new JTextField(15));  
14.        p2.add(new JLabel("Mat khau: "));  
15.        p2.add(new JPasswordField(15));
```

```
1. p.add(Box.createRigidArea(new Dimension(10, 10))) ;  
2. p.add(p1) ;  
3. p.add(Box.createRigidArea(new Dimension(10, 10))) ;  
4. p.add(p2) ;  
5. this.add(p, BorderLayout.NORTH) ;  
6. }  
7. public static void main(String[] args) {  
8.     new BoxLayout1().setVisible(true) ;  
9. }  
10. }
```



```
1. public class BoxLayout2 extends JFrame {  
2.     public BoxLayout2 () {  
3.         setTitle("Vi du Box Layout login");  
4.         setSize(300,150);  
5.  
6.         setDefaultCloseOperation(EXIT_ON_CLOSE);  
7.         Box b=Box.createVerticalBox();  
8.         Box p1=Box.createHorizontalBox();  
9.         Box p2=Box.createHorizontalBox();  
10.        p1.add(new JLabel("Nhap ten: "));  
11.        p1.add(new JTextField(15));  
12.        p2.add(new JLabel("Mat khau: "));  
13.        p2.add(new JPasswordField(15));  
14.    }  
15.    public static void main(String[] args) {  
16.        BoxLayout2 f=new BoxLayout2();  
17.        f.setVisible(true);  
18.    }  
19.}
```

```
1. b.add(Box.createRigidArea(new  
    Dimension(10, 10)));  
2. b.add(p1);  
3. b.add(Box.createRigidArea(new  
Dimension(10, 10)));  
4. b.add(p2);  
5. this.add(b, BorderLayout.NORTH);  
6. }  
7. public static void main(String[]  
    args) {  
8.     new BoxLayout2().setVisible(true);  
9. }  
10. }
```

# Ví dụ đăng nhập (gói event)

```
1. public class DemoLogin extends JFrame  
    implements ActionListener {  
2.     private JButton bLogon;  
3.     private JButton bExit;  
4.     private JTextField username;  
5.     private JPasswordField pass;  
6.     public DemoLogin() {  
7.         setTitle("Logon program");  
8.         setSize(500, 350);  
9.         setDefaultCloseOperation(EXIT_ON_CLOSE);  
10.        setLocationRelativeTo(null);  
11.        buildGUI();  
12.    }
```

```
1. private void buildGUI() {  
2.     JPanel p1=new JPanel();  
3.  
4.     p1.setBorder(BorderFactory.createLineBorder(Color.red));  
5.     JLabel tieude;  
6.     p1.add(tieude = new JLabel("DANG NHAP"));  
7.     tieude.setFont(new Font("Arial", Font.BOLD, 30));  
8.     tieude.setForeground(Color.red);  
9.     add(p1, BorderLayout.NORTH);  
10.    JPanel p2 = new JPanel();  
11.    p2.setBorder(BorderFactory.createLineBorder(Color.red));  
12.    p2.add(bLogon=new JButton("Dang Nhap"));  
13.    p2.add(bExit=new JButton("Exit"));  
14.    add(p2, BorderLayout.SOUTH);  
15.    JPanel p3=new JPanel();  
16.    p3.setBorder(BorderFactory.createLineBorder(Color.red));  
17.    Box b=Box.createVerticalBox();  
18.    Box b1=Box.createHorizontalBox();  
19.    Box b2=Box.createHorizontalBox();
```

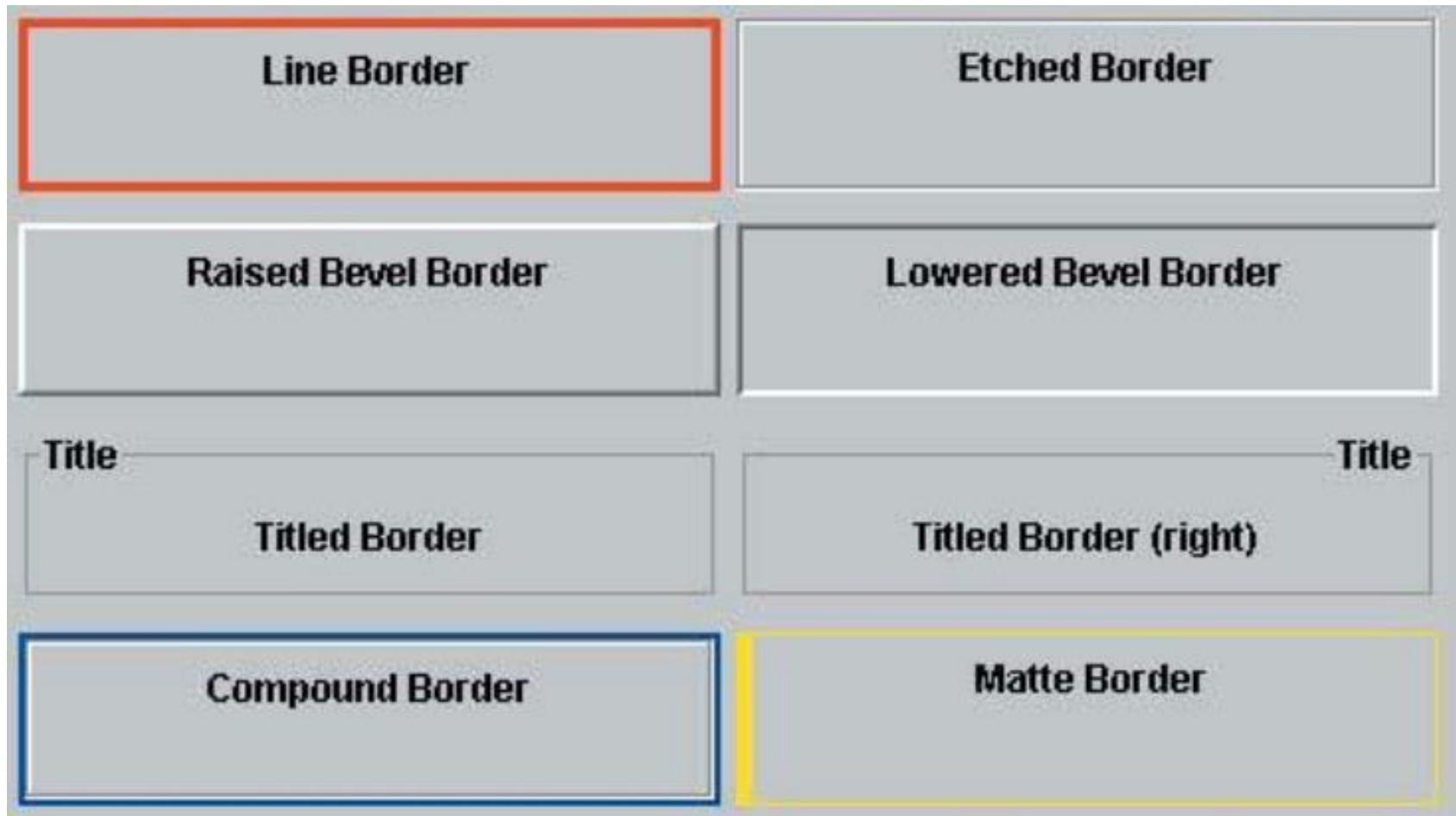
```
1. JLabel      lblUser, lblPass;
2.         b1.add(lblUser=new JLabel("Tên:   "));
3.         lblUser.setFont(new Font("Arial",
Font.PLAIN, 15));
4.         b1.add(username= new JTextField(20));
5.         b2.add(lblPass=new JLabel("Mat khau:   "));
6.         lblPass.setFont(new Font("Arial",
Font.PLAIN, 15));
7.         b2.add(pass=new JPasswordField(20));
8.
lblUser.setPreferredSize(lblPass.getPreferredSize());
9.         b.add(Box.createVerticalStrut(50));
10.        b.add(b1);
11.        b.add(Box.createVerticalStrut(10));
12.        b.add(b2);
13.        p3.add(b);
14.        add(p3,BorderLayout.CENTER);
15.        username.addActionListener(this);
16.        pass.addActionListener(this);
17.        bLogon.addActionListener(this);
18.        bExit.addActionListener(this);
19.    }
```

```
1. @Override
2.     public void actionPerformed(ActionEvent e) {
3.         if(e.getSource() == bLogon) {
4.
5.             if(username.getText().equalsIgnoreCase("anh") &&
6.                 pass.getText().equalsIgnoreCase("anh")) {
7.                 dispose();
8.
9.                 JOptionPane.showMessageDialog(null, "Dang nhap thanh
cong!!!!");
10.            }
11.        }
12.    }
13.    else if(e.getSource() == bExit) {
14.        System.exit(0);
15.    }
16. }
17. }
```

# Borders

- Tạo đường viền cho đối tượng, extends Jcomponent
- Tạo đường viền cho đối tượng thì dùng BorderFactory.create**Xxx**Border(...)
- Ép đường viền cho đối tượng dùng phương thức setBorder()
- Ví dụ
- `label.setBorder(BorderFactory.createLineBorder(Color.red));`

# Ví dụ



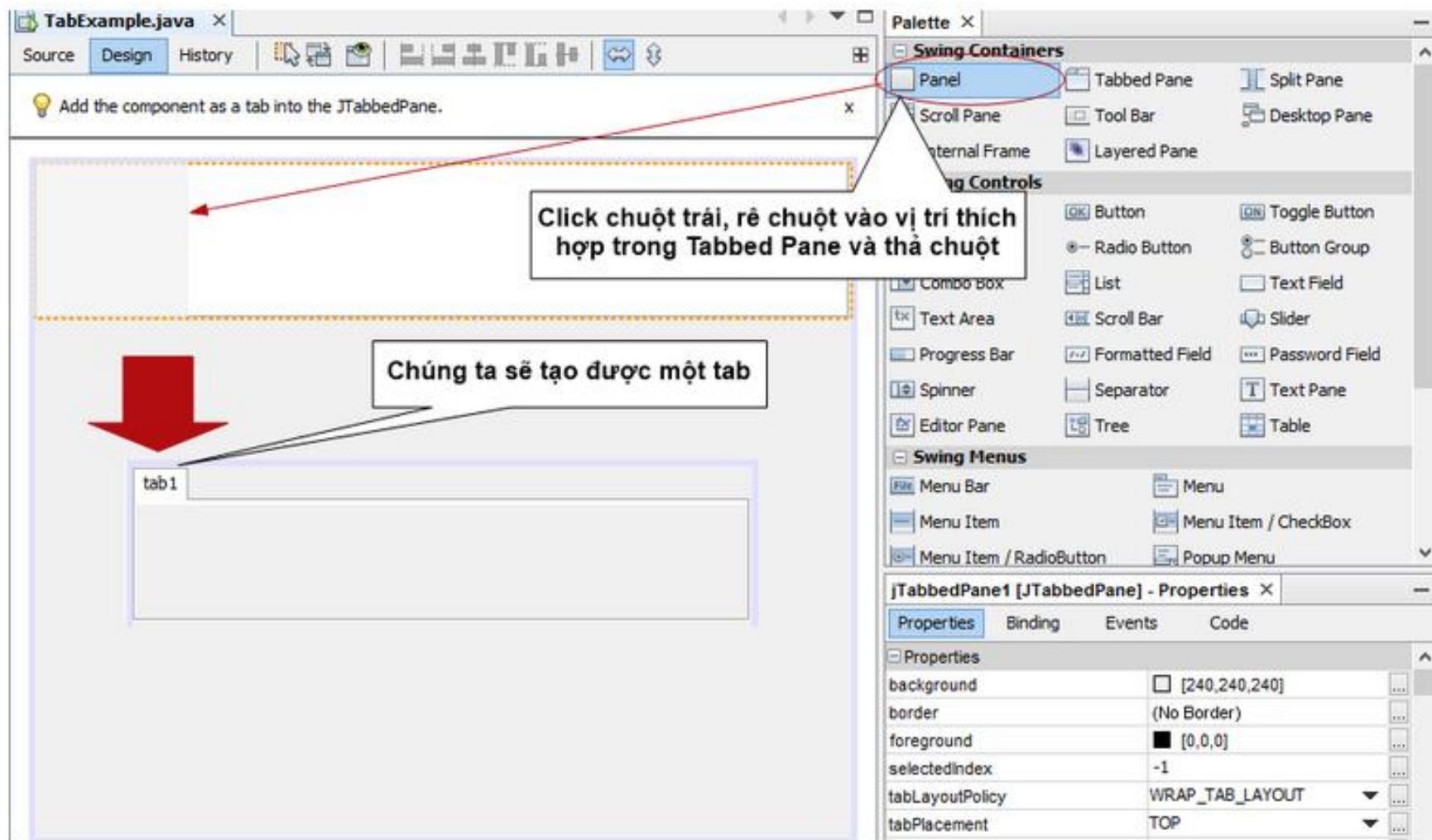
# JTabbedPane

- JTabbedPane là một thành phần cho phép người lập trình thêm một số container như JPanel trong một cửa sổ duy nhất. Mỗi thành phần được thêm vào JTabbedPane tương ứng với một tab.
- Tạo Tabbed Pane với NetBeans
  - Chọn công cụ Tabbed Pane để đưa vào màn hình
  - Thêm Panel vào Tabbed Pane
  - Sau khi thêm Panel vào, chúng ta tiến hành thiết kế giao diện bằng cách chọn các thành phần trong Swing Controls và đặt vào Panel. Nếu bạn muốn thêm một Panel khác vào Tabbed Pane thì cách thực hiện tương tự như lúc thêm Panel đầu tiên.

# Ví dụ (gói tab)

```
public class TabbedPane {  
    private static void showGUI() {  
        final JFrame m = new JFrame("Tabbed Pane  
Example");  
        m.setSize(400, 400);    m.setVisible(true);  
        m.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        m.setLayout(new GridLayout(1, 1));  
        JTabbedPane tab = new  
        JTabbedPane(JTabbedPane.TOP);  
        tab.addTab("Tab1", addPanel("This is tab1"));  
        tab.addTab("Tab2", addPanel("This is tab2"));  
        tab.addTab("Tab3", addPanel("This is tab3"));  
        tab.addTab("Tab4", addPanel("This is tab4"));  
        m.add(tab);  
    }  
}
```

```
int selectedIndex = tab.getSelectedIndex();
System.out.println("Default Index:" +
selectedIndex);
tab.setSelectedIndex(tab.getTabCount() - 1);
selectedIndex = tab.getSelectedIndex();
System.out.println("Index:" + selectedIndex);
}
private static JPanel addPanel(String text) {
JPanel p = new JPanel();
p.add(new JLabel(text));
p.setLayout(new GridLayout(1, 1));
return p;
}
public static void main(String[] args) {
    showGUI();
}
}
```



# JCheckBox - JRadioButton

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

# JCheckBox – Constructor

- JCheckBox()
- JCheckBox(String text)
- JCheckBox(String text, boolean selected)
- JCheckBox(Icon icon)
- JCheckBox(String text, Icon icon)
- JCheckBox(String text, Icon icon, boolean selected)

# JCheckBox – Methods

- `boolean isSelected()`
  - returns the state of the checkbox
- `void setSelected(boolean state)`
  - sets the checkbox to a new state
- `String getText()`
- `void setText(String text)`
  - gets or sets the button's text
- `addItemListener`
  - Add an ItemListener to process ItemEvent in itemStateChanged

# JCheckBox Demo

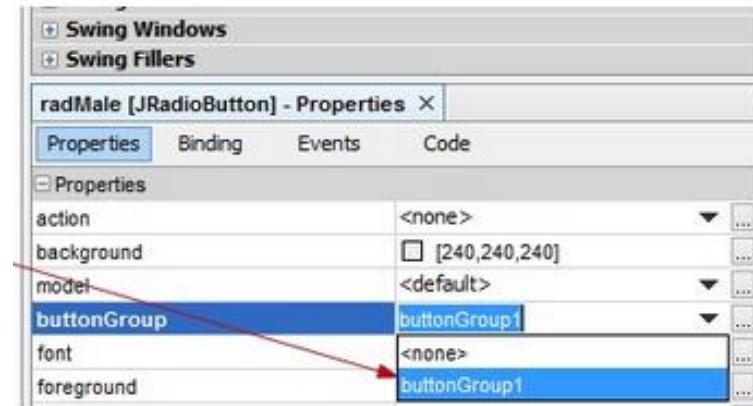
- JCheckBox ga, bo;
- JPanel p2 = new JPanel();
- p2.add(ga = new JCheckBox("Đùi gà"));
- p2.add(bo = new JCheckBox("Bò tái chanh"));
- add(p2);
- public void **itemStateChanged**(ItemEvent e) {
- if (**e.getItem() == ga**)  
JOptionPane.showMessageDialog(null, "Bạn chọn đùi gà");
- if (**e.getItem() == bo**)  
JOptionPane.showMessageDialog(null, "Bạn chọn bò tái chanh");

# JRadioButton – Constructor

- JRadioButton()
- JRadioButton(String text)
- JRadioButton(String text, boolean selected)
- JRadioButton(Icon icon)
- JRadioButton(String text, Icon icon)
- JRadioButton(String text, Icon icon, boolean selected)

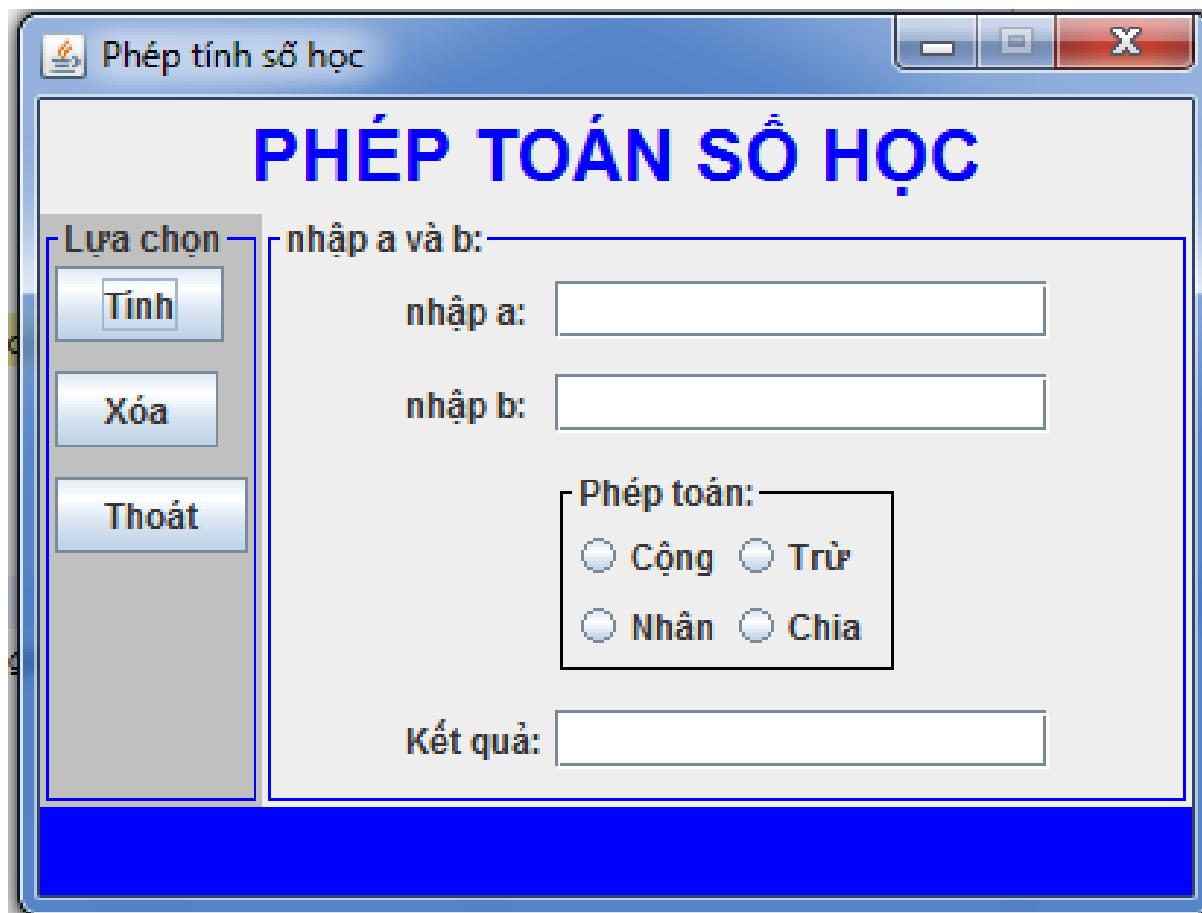
# Gắn JRadioButton vào Button Group

- ButtonGroup pt=new  
ButtonGroup();
- Final JRadioButton cong=new  
JRadioButton("Cộng");
- Final JRadioButton tru=new  
JRadioButton("Trừ");
- Final JRadioButton nhan=new  
JRadioButton("Nhân");
- Final JRadioButton chia=new  
JRadioButton("Chia");
- pt.add(cong);pt.add(tru);
- pt.add(nhan);pt.add(chia);



# Ví dụ JRadioButton

## ■ Gói **jradio**



# JComboBox (1)

- JComboBox là thành phần cho phép người dùng lựa chọn từ danh sách thả xuống (dropdown list).



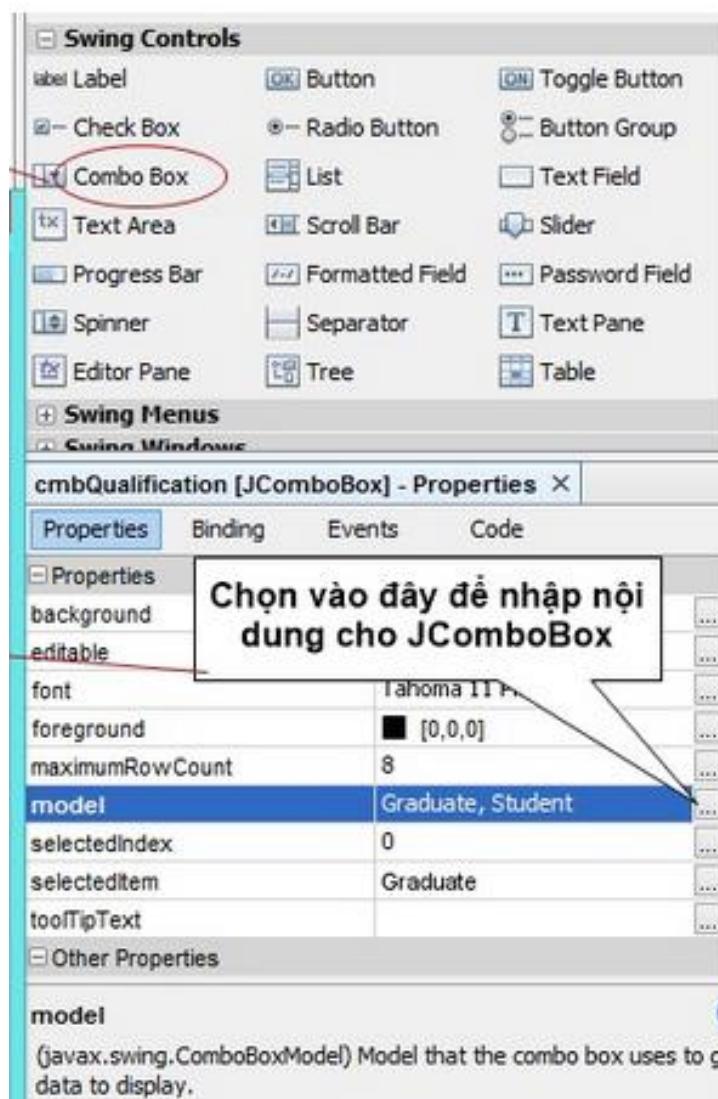
## Constructors

- public JComboBox()
- public JComboBox(Object[] items)
- public JComboBox(Vector items)

### Common used methods

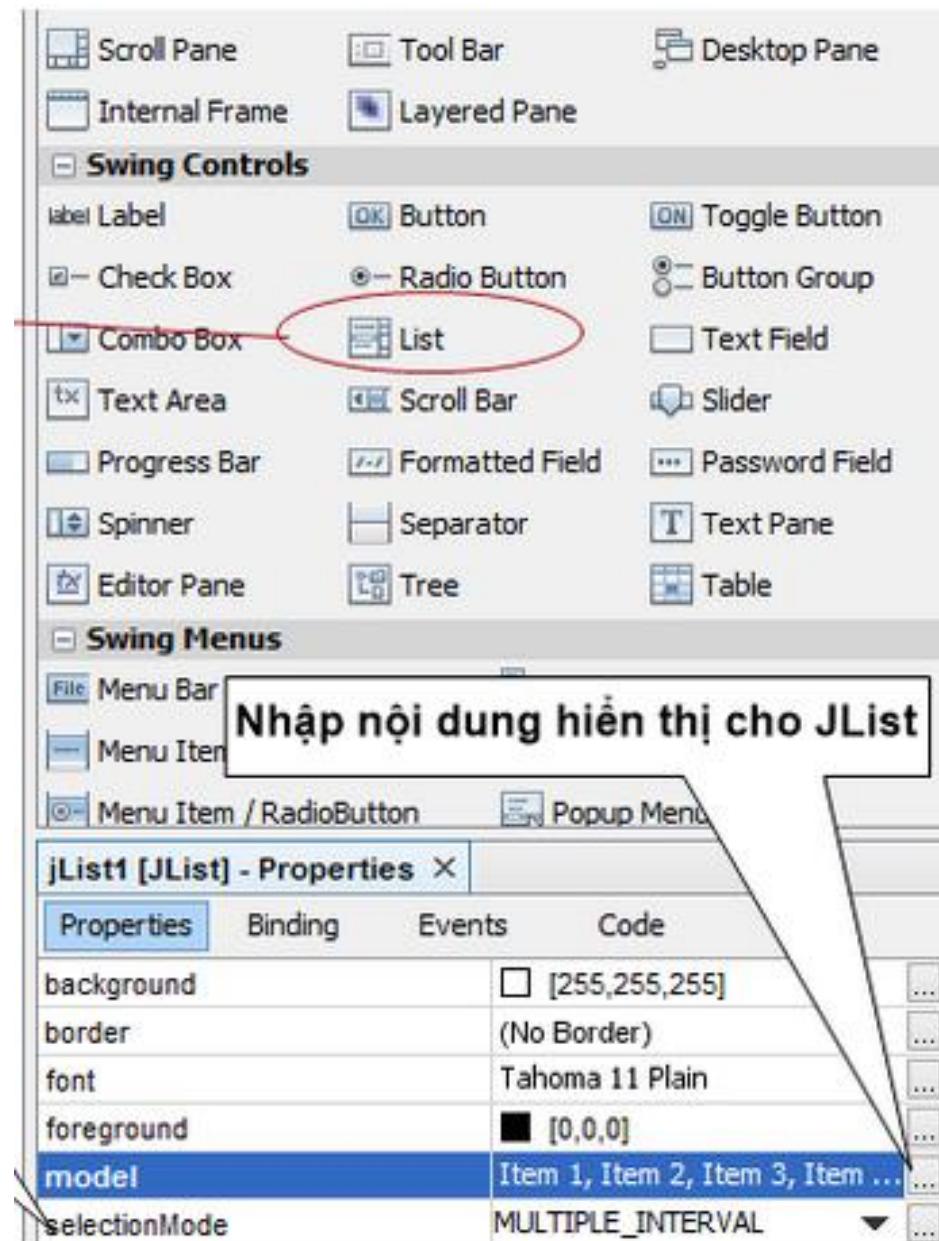
- public void addItem(Object item)
- public int getItemCount()
- public int getSelectedIndex()
- public Object getSelectedItem()
- public void setEditable(boolean editable)
- public void setSelectedIndex(int index)
- public void setSelectedItem(Object item)

# JComboBox (2)



# JList (1)

- JList được sử dụng để hiển thị một nhóm các item. Và nó cho phép chọn một hoặc nhiều item. Các item trong JList được hiển thị trong một hoặc nhiều dòng.



# Jlist (2)

## ■ Constructors

- JList()
- JList ( Object[] dataItems)
- JList ( Vector vectorItems)
- JList được khởi tạo với 1 model là DefaultListModel,  
đó là 1 model mà Java xây dựng sẵn

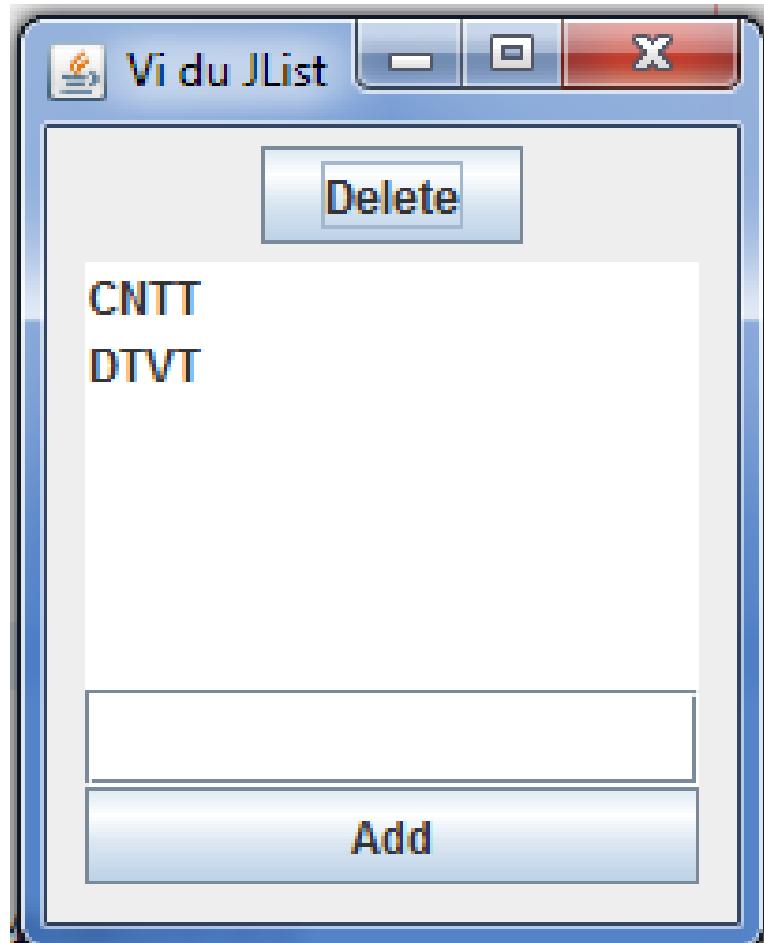
## ■ Event handling:

- ListSelectionEvent
- ListSelectionListener
- public void valueChanged(ListSelectionEvent e)

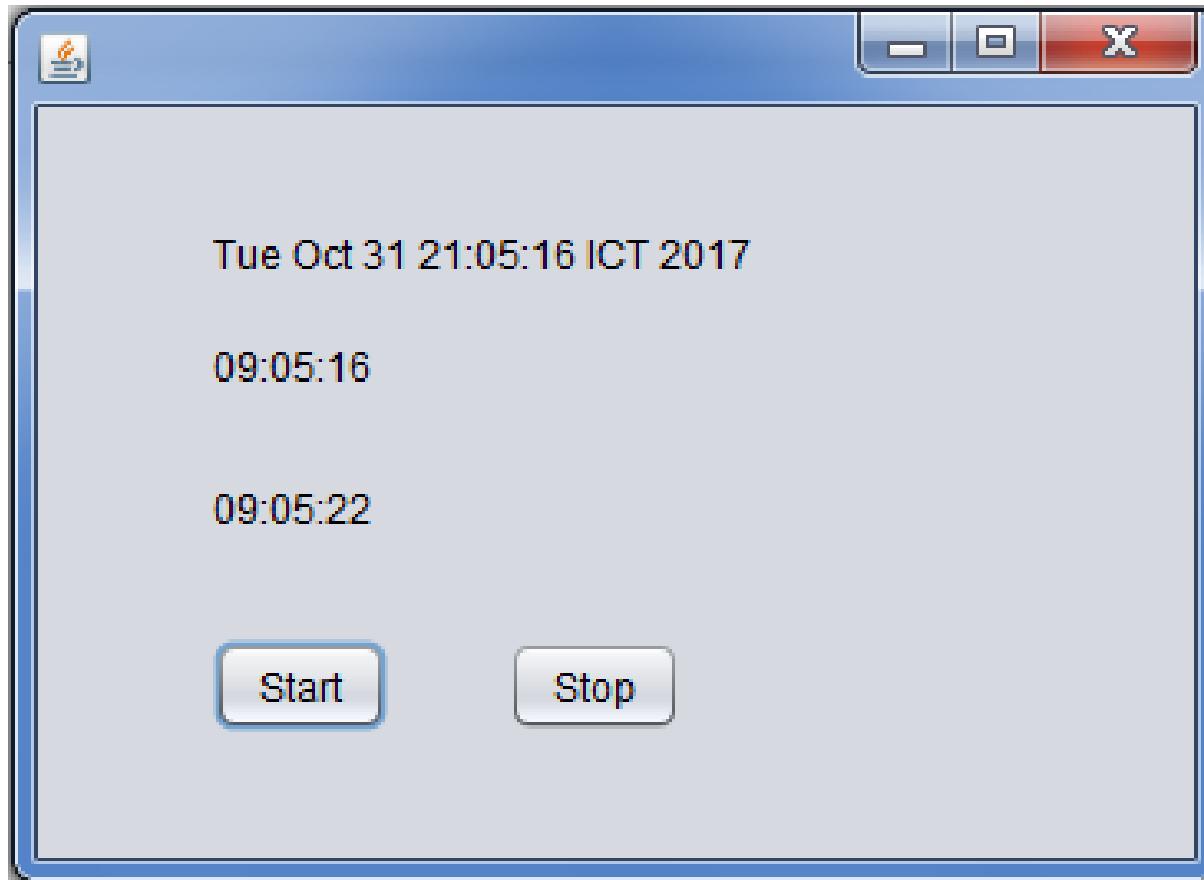
```
public void clearSelection()
public int getSelectedIndex()
public int[] getSelectedIndices()
public boolean isEmpty()
public void setListData(Object[] items)
public void setSelectedIndex(int index)
public void setSelectedIndices(int[] indices)
```

# Ví dụ

- Nằm trong gói: **jlist**



# Ví dụ (clock)



# JSlider

- Lớp **JSlider** được sử dụng để tạo con trượt slider. Bởi sử dụng JSlider, một người dùng có thể lựa chọn một giá trị từ một dãy cụ thể.
- **JSlider()**: tạo một slider với giá trị khởi tạo là 50 và dãy giá trị là từ 0 tới 100.
- **JSlider(int orientation)**: tạo một slider với orientation đã cho được thiết lập bởi hoặc JSlider.HORIZONTAL hoặc JSlider.VERTICAL với dãy từ 0 tới 100 và giá trị khởi tạo là 50.
- **JSlider(int min, int max)**: tạo một thanh slider ngang bởi sử dụng giá trị min và max đã cho.
- **JSlider(int min, int max, int value)**: tạo một thanh slider ngang bởi sử dụng giá trị min, max và value đã cho.
- **JSlider(int orientation, int min, int max, int value)**: tạo một slider bởi sử dụng orientation, min, max và value đã cho.

# Gói slider

```
JLabel jLabel1;
public Main() {
    initComponents();
    setLayout(new BorderLayout());
    jLabel1=new JLabel("Java is
cool",JLabel.CENTER);
    jLabel1.setFont(new Font("Times New Roman",
Font.BOLD, 32));
    add(jLabel1,BorderLayout.NORTH);
    add(jSlider1,BorderLayout.CENTER);
    jSlider1.setMinimum(200);
    jSlider1.setMaximum(1000);
    jSlider1.setMinorTickSpacing(20);
    jSlider1.setMajorTickSpacing(100);
    jSlider1.setPaintLabels(true);
    jSlider1.setPaintTicks(true);
```

# JFileChooser

- Lớp **JFileChooser** là một thành phần cung cấp một kỹ thuật đơn giản cho người dùng để lựa chọn một file.
- **constructor**
- **JFileChooser()**
- **JFileChooser(File currentDirectory)**
- **int showDialog(Component parent, String approveButtonText)** Popup một hộp thoại file chooser tùy biến với một nút xác nhận tùy biến
- **int showSaveDialog(Component parent)**  
Hiển thị một hộp thoại "Save File"
- **int showOpenDialog(Component parent)**  
Hiển thị một hộp thoại "Open File"

# Ví dụ



```
private void  
jButton1ActionPerformed(java.awt.  
event.ActionEvent evt) {  
txtfile.setText("'" + displayChosenF  
ile());  
}  
}
```

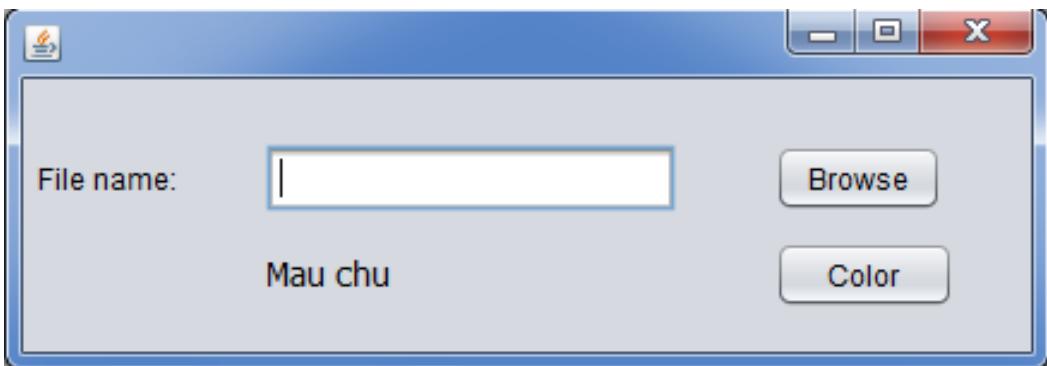
```
private String displayChosenFile() {  
    String filestr=null;  
    JFileChooser file=new JFileChooser(".");  
    int select=file.showOpenDialog(null);  
    if(select==JFileChooser.APPROVE_OPTION) {  
        File  
        selectedFile=file.getSelectedFile();  
        System.out.println(selectedFile.getParent());  
        System.out.println(selectedFile.getName());  
        try{  
            filestr=selectedFile.getCanonicalPath();  
        }catch(Exception e){  
            e.printStackTrace();  
        }  
    }  
    return filestr;  
}
```

đối số **JFileChooser.APPROVE\_OPTION** thể hiện là bạn đã chấp nhận “Open” hoặc “Save” file.

# JColorChooser

- Lớp JColorChooser cung cấp một pane cho các control được thiết kế để cho phép người dùng thao tác và lựa chọn một màu.
- **JColorChooser():** Tạo một bảng chọn màu với một màu ban đầu là màu trắng.
- **JColorChooser(Color initialColor):** Tạo một bảng chọn màu với màu khởi tạo đã cho.

```
Color c =  
    JColorChooser.showDialog(this,  
        "Choose foreground  
color", Color.BLACK);  
  
    if(c != null) {  
jTextArea1.setForeground(c);  
jButton1.setForeground(c);  
}
```



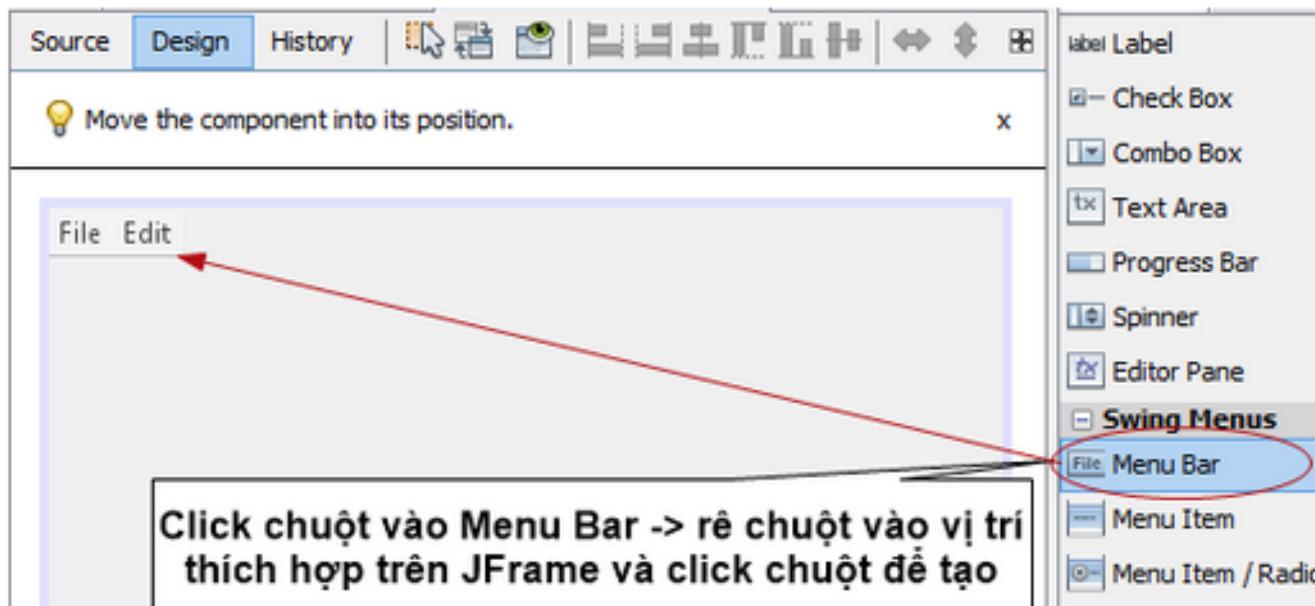
# Swing Menu Components

## Objectives



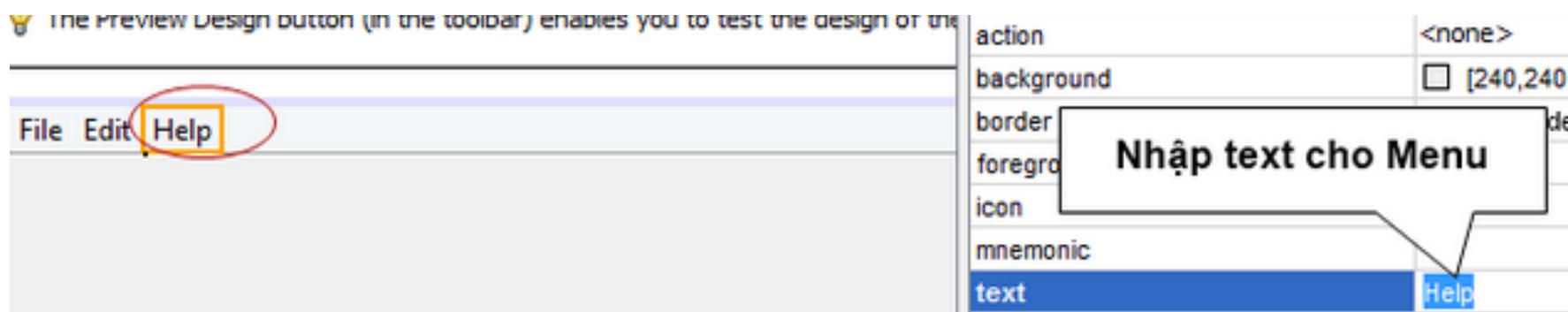
# JMenuBar class

- JMenuBar là một class và được sử dụng để tạo một thanh menu. Một thanh menu chứa một hoặc nhiều item, ví dụ File, Edit, View, ...
- JFrame f= new JFrame ("Menu demo");
- JMenuBar bar = new JMenuBar();
- f.setJMenuBar(bar);



# JMenu class

- **JMenu** là một class được sử dụng để tạo một menu trên thanh menu.
- Constructors:
  - JMenu ()
  - JMenu (String label)
  - JMenu mfile= new JMenu ("File");
  - bar.add(mfile);
  - mfile.addSeparator();
  - mfile.setMnemonic (KyEvent.VK\_F);



# JMenuItem

## ■ Constructors:

- JMenuItem()
- JMenuItem(Action a)
- JMenuItem(Icon icon)
- JMenuItem(String text)
- JMenuItem(String text, Icon icon)
- JMenuItem(String text, int mnemonic)

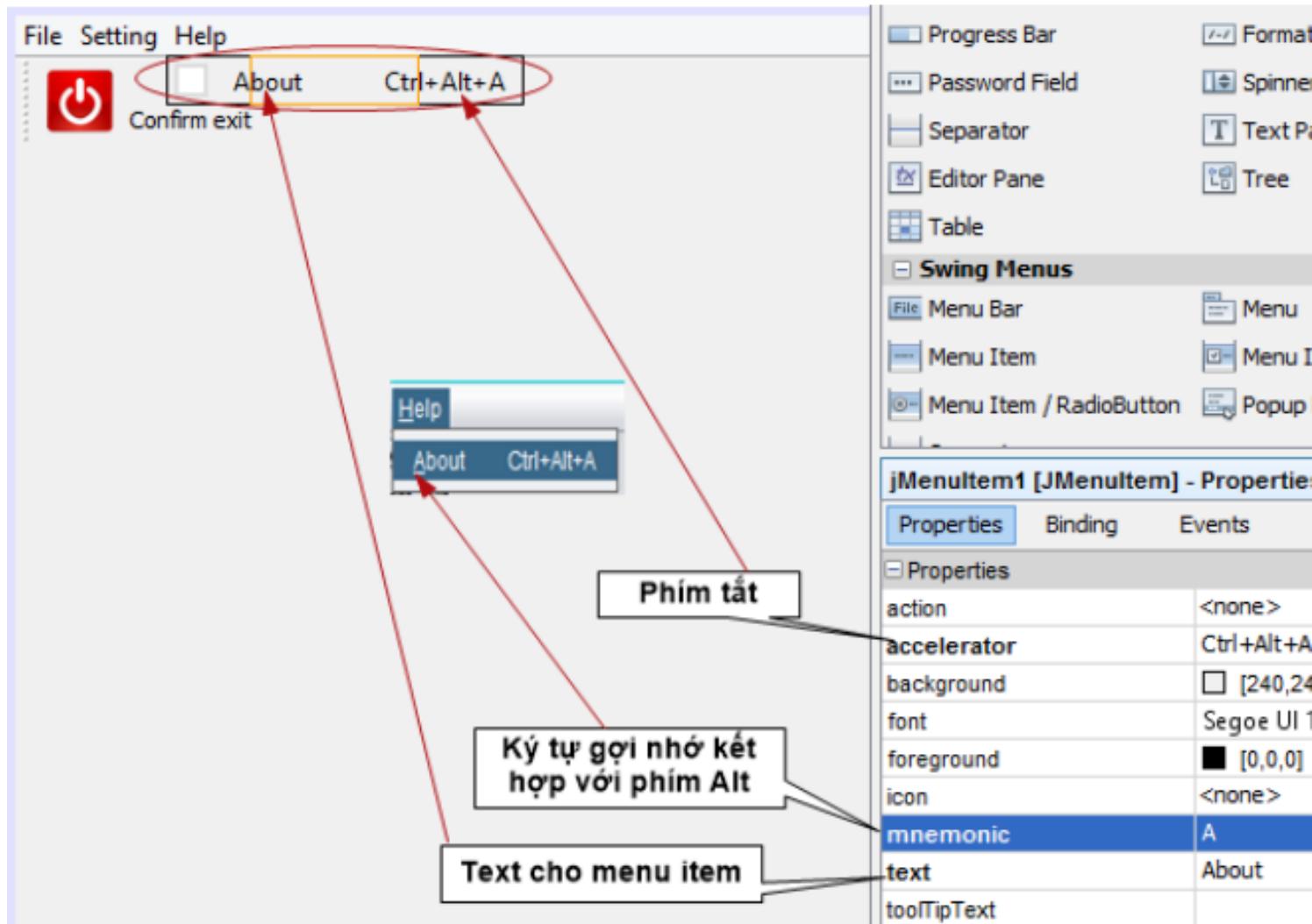
## ■ Important methods

- setEnable(boolean enable)
- setMnemonic(int mnemonic)
- setAccelerator(KeyStroke keyStroke)

## ■ Add menu item to menu

- menuObject.add(menuItemObject)

# Tùy chỉnh menu item



# JCheckBoxMenuItem

## ■ Constructors:

- JCheckBoxMenuItem()
- JCheckBoxMenuItem(Action a)
- JCheckBoxMenuItem(String text)
- JCheckBoxMenuItem(Icon icon)
- JCheckBoxMenuItem(String text, Icon icon)
- JCheckBoxMenuItem(String text, boolean b)
- JCheckBoxMenuItem(String text, Icon icon, boolean b)

## ■ Important methods

- boolean isSelected()
- get/ setSelected (boolean)
- get/setState(boolean)

## ■ Add menu item to menu

```
menuObject.add(checkBoxMenuItemObject)
```

# JCheckBoxMenuItem

## ■ Constructors:

- JCheckBoxMenuItem()
- JCheckBoxMenuItem(Action a)
- JCheckBoxMenuItem(String text)
- JCheckBoxMenuItem(Icon icon)
- JCheckBoxMenuItem(String text, Icon icon)
- JCheckBoxMenuItem(String text, boolean b)
- JCheckBoxMenuItem(String text, Icon icon, boolean b)

## ■ Important methods

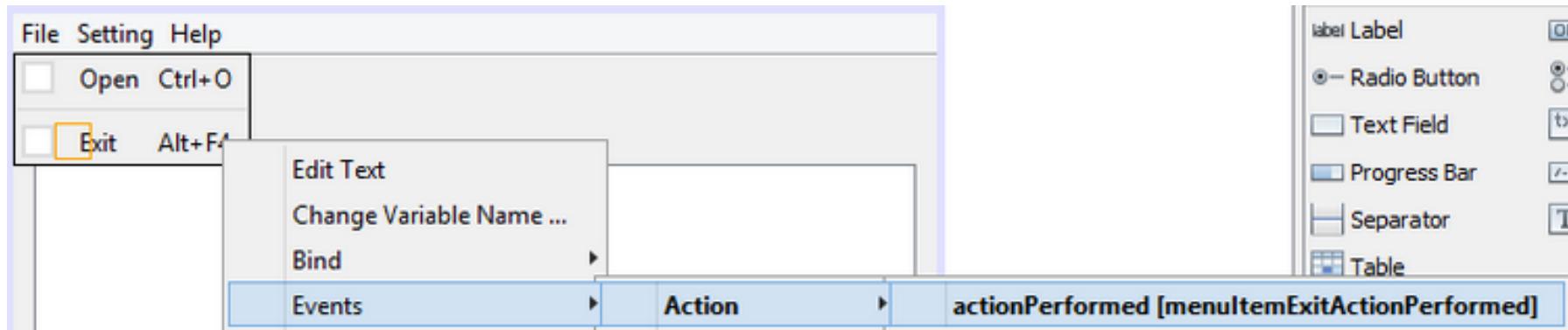
- boolean isSelected()
- get/ setSelected (boolean)
- get/setState(boolean)

## ■ Add menu item to menu

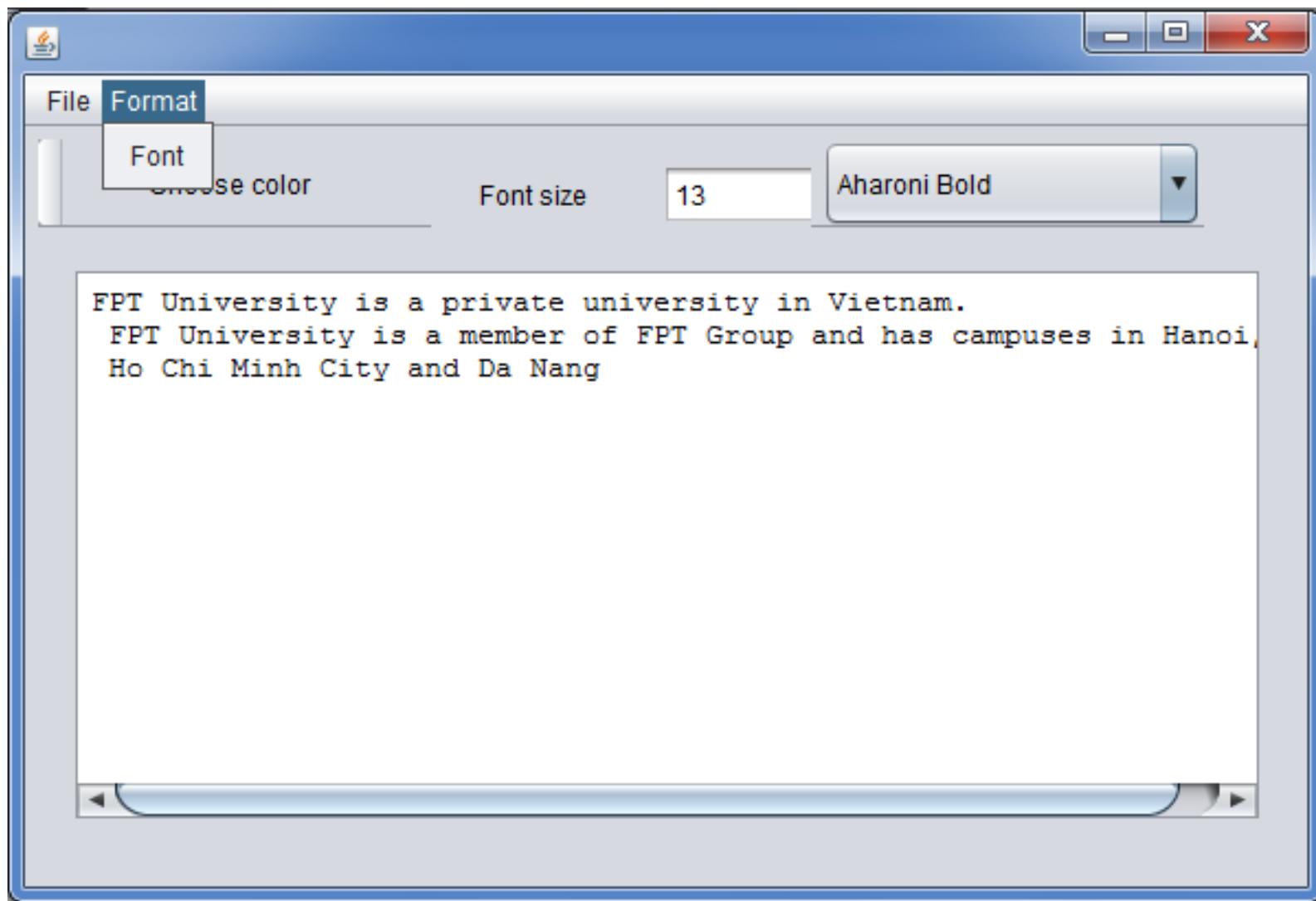
```
menuObject.add(checkBoxMenuItemObject)
```

# Đăng ký sự kiện cho Menu item

- Cách đăng ký được thực hiện tương tự như Button



# Ví dụ (editor)



# JTable

- JTable là một thành phần cho phép hiển thị dữ liệu theo dòng (row) và cột (column). Giao nhau giữa dòng và cột được gọi là ô (cell) và đây là nơi hiển thị dữ liệu. Một JTable có 2 phần là **Column Header** (tiêu đề) và **Data** (dữ liệu).

# Constructors - Methods of JTable

- `JTable( Object[][][] entries, Object[] columnNames )`
  - constructs a table with a default table model
- `JTable( TableModel model )`
  - displays the elements in the specified, non-null table model
- `int getSelectedRow()`
  - returns the index of the first selected row, -1 if no row is selected
- `Object getValueAt( int row, int column )`
- `void setValueAt( Object value, int row, int column )`
  - gets or sets the value at the given row and column
- `int getRowCount()`
  - returns the number of row in the table

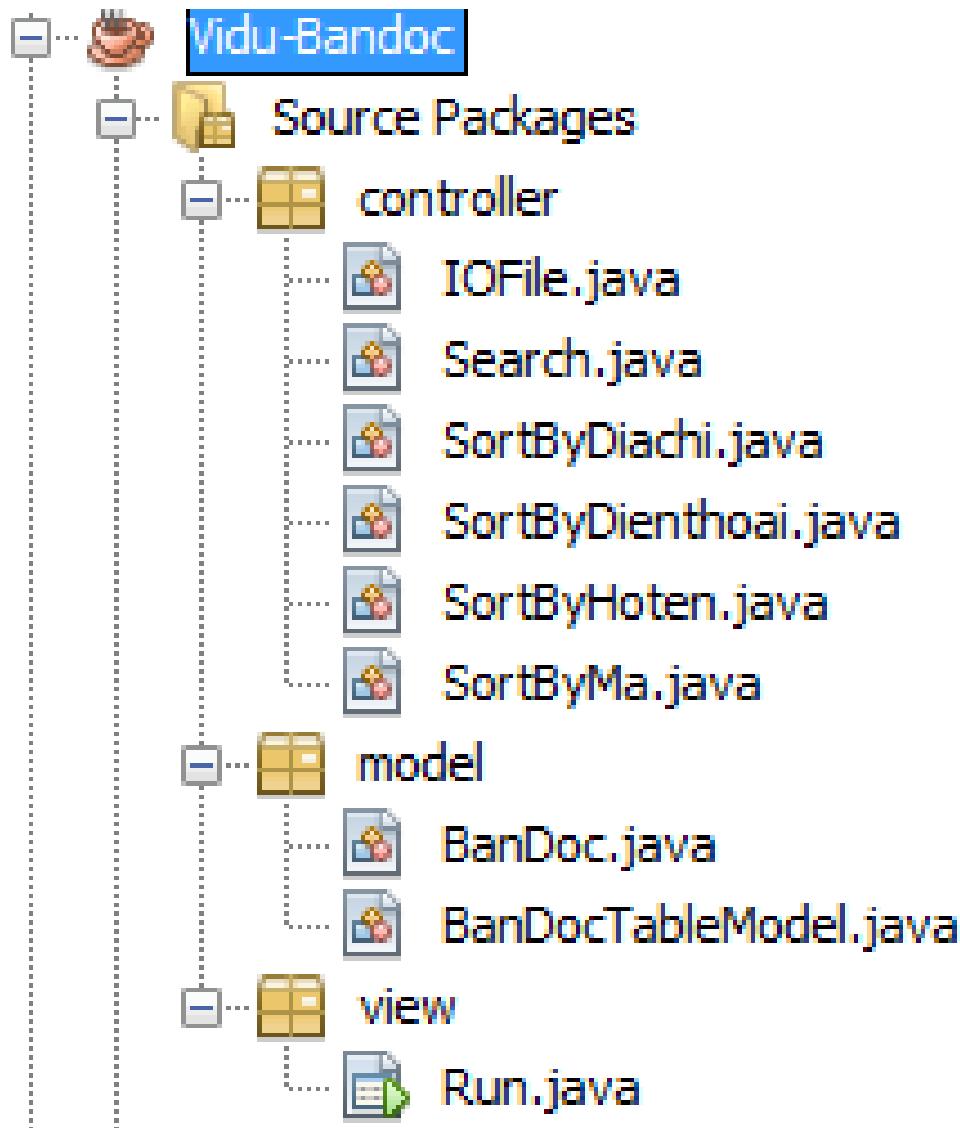
# JTable with changeable choices

- Tạo JTable:
- Tạo 1 mảng tên cột, tạo một DefaultTableModel
- String[] cols= {"Ma mon", "Ten mon", "So tin chi"};
- DefaultTableModel **model**=new DefaultTableModel(cols,0);
- JTable table = new JTable(model);
- JScrollPane pane = new JScrollPane(table);
- Add/remove elements
- Use the **model**, not the JTable directly

# Methods in DefaultTableModel

- void **addRow**( Object[] rowData )
  - Thêm 1 dòng dữ liệu vào cuối của table model
- void **insertRow**( int row, Object[] rowData )
  - Thêm 1 dòng dữ liệu vào dòng thứ index
- void **removeRow**( int row )
  - Xóa 1 dòng ở dòng thứ row ở model
- void **setValueAt**( Object value, int row, int column )
  - Đặt giá trị ở ô có tọa độ row và column
- Void **fireTableDataChanged()**

# Ví dụ (gói jTable)



# BanDoc.java

```
public class BanDoc implements  
Serializable{  
  
    private String maBD;  
    private String tenBD;  
    private String diachi;  
    private String dienthoai;  
  
    public BanDoc() {  
    }  
  
    //getter and setter  
}
```

# BanDocTableModel.java

```
1. public class BanDocTableModel extends AbstractTableModel {  
2.     private String[] columns = {"Ma ban  
doc", "Ho ten", "Dia chi", "Dien thoai"};  
3.     ArrayList<BanDoc> bandoc;  
  
4.     public ArrayList<BanDoc> getBanDoc() {  
5.         return bandoc;    }  
6.     public void setBanDoc(ArrayList<BanDoc>  
bandoc) {  
7.         this.bandoc = bandoc;    }  
8.     public int getRowCount() {  
9.         return bandoc.size();    }  
10.    public int getColumnCount() {  
11.        return columns.length;    }
```

```
1. @Override
2. public String getColumnName(int column) {
3.     return columns[column];
4. }
5. public Object getValueAt(int rowIndex, int
columnIndex) {
6.     BanDoc bd = bandoc.get(rowIndex);
7.     if(columns[columnIndex].equals("Ma ban doc")) {
8.         return bd.getMaBD();
9.     }
10.    else if(columns[columnIndex].equals("Ho ten")) {
11.        return bd.getTenBD();
12.    }
13.    else if(columns[columnIndex].equals("Dia chi")) {
14.        return bd.getDiachi();
15.    }
16.    else if(columns[columnIndex].equals("Dien thoai")) {
17.        return bd.getDienthoaai();
18.    }
19.    return "";
20. }
21. }
```

**Ban doc****Tim kiem va sap xep**

Ma ban doc	Ho ten	Dia chi	Dien thoai
a1	ha anh	so 123	1234
c342	Vu thanh	so 22bn	345678
b123	cao Lam	chua boc	23455

**Them moi****Sua****Xoa**

Ma ban doc:

**Cap nhat**

Ho ten ban doc:

Dia chi:

**Bo qua**

Dien thoai:

# Ví dụ (Vidu-De1)

Mã bạn đọc	Tên bạn đọc	Địa chỉ	Số điện thoại
10000	Cao Ha	so 34	123
10001	Vu Xuan	so 564 chua boc	765
10002	Tran Ian	so 87 Nguyen khuyen	876

Thêm mới      Sửa      Xóa

Mã bạn đọc:

Họ và tên:  Cập nhật

Địa chỉ:

Điện thoại:  Bỏ qua