

Lập trình Java

BÀI 5 GUI

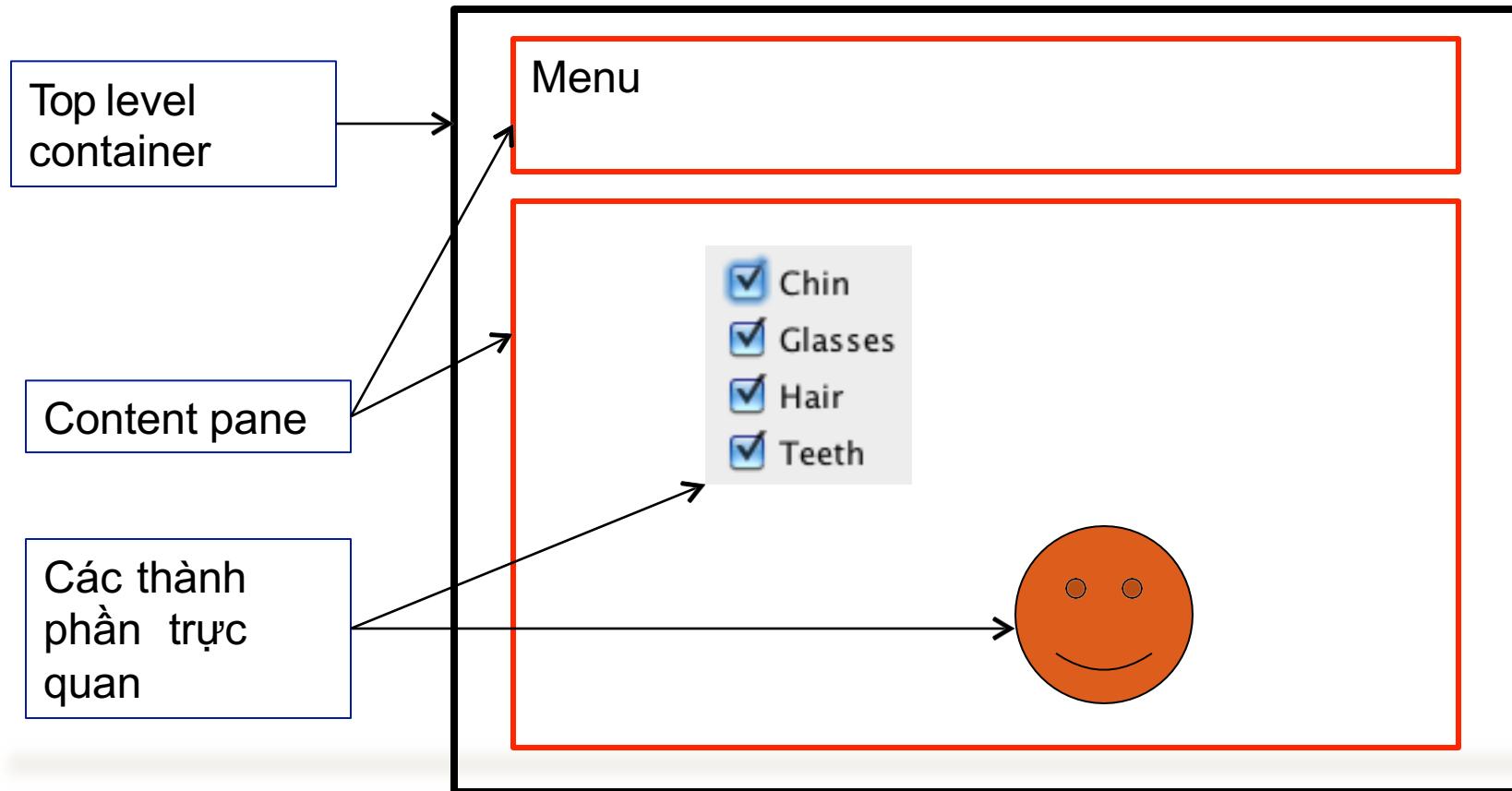
Nội dung

- Giới thiệu giao diện người dùng với Java (Java GUI)
 - Các container: JFrame, JPanel, JDialog
 - Các thành phần: JLabel, JButton, JCheckBox, JRadioButton, JListBox, JComboBox..
 - Layout Manager: FlowLayout, GridLayout, BoderLayout
 - Menu
-

Giao diện người dùng với java

- Dùng thư viện Swing để tạo giao diện cho ứng dụng. Xử lý các tương tác của người dùng đến ứng dụng
- Giao diện người dùng gồm một cửa sổ chính và các control được tạo ra trên cửa sổ này
- Các thành phần tạo ra giao diện nằm trong gói `java.swing`
- Tên của các lớp này bắt đầu bằng ký tự ‘J’

Giao diện người dùng với java



GUI's

- Gói *AWT* được sử dụng trong các phiên bản trước jdk1.2 (java2)
- *Swing* được đưa vào trong phiên bản jdk1.2 và có nhiều chức năng hơn AWT
- *Swing Components* thường có tên bắt đầu với 'J'
 - AWT có lớp *Panel*
 - Swing có lớp tương ứng là *JPanel*

Graphical User Interface (GUI)

AWT (Abstract Windows Toolkits)

```
import java.awt.*;
```

Swing (Java Foundation Classes Package)

```
import javax.swing.*;  
import java.awt.*;  
import java.awt.event.*;
```

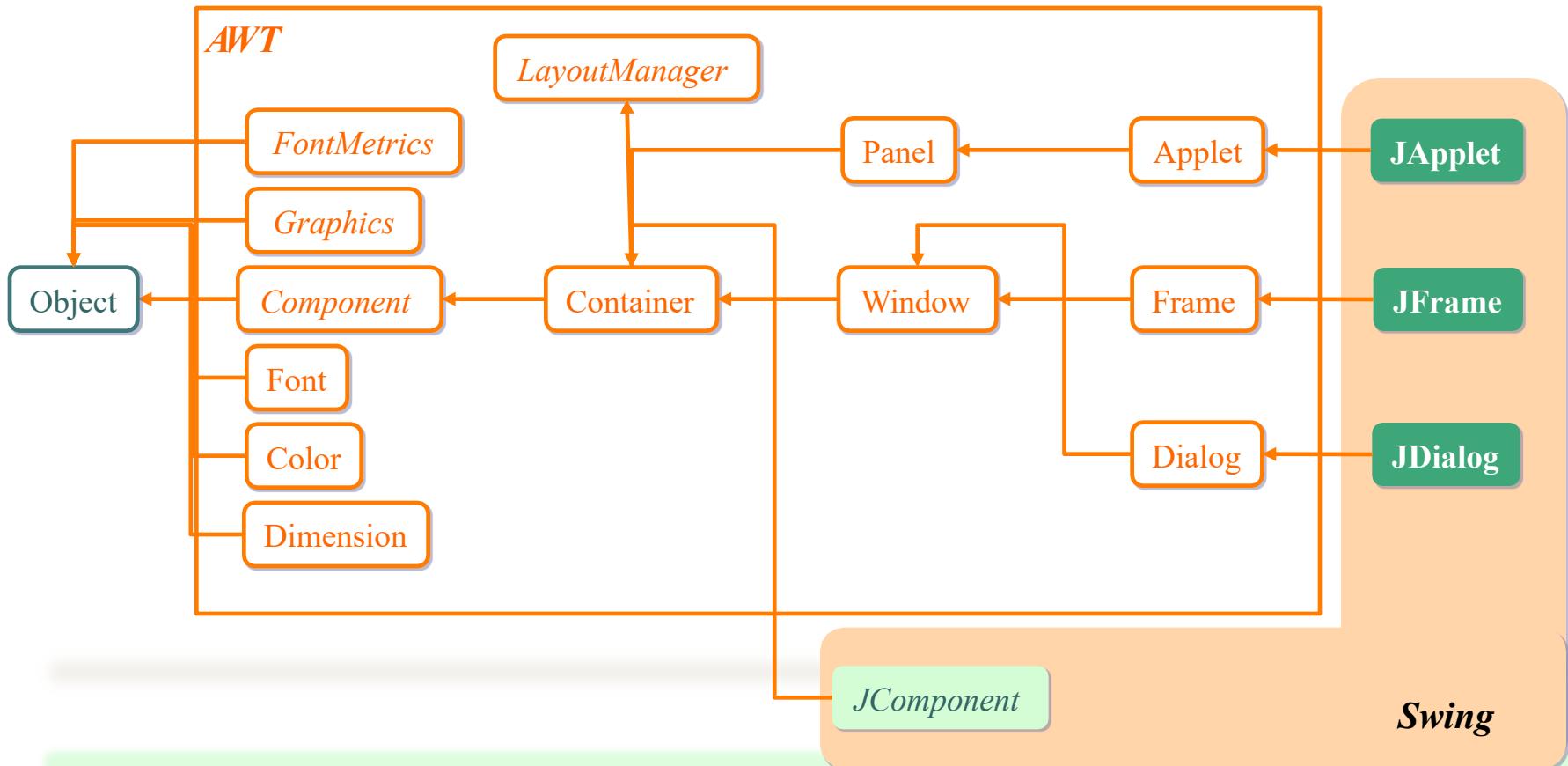
Tất cả Swing component có tên bắt đầu với J...

The Swing API

- Gói Swing là 1 thành phần của *Java Foundation Classes* (JFC)
- JFC cung cấp một tập các chức năng giúp xây dựng các ứng dụng GUI.

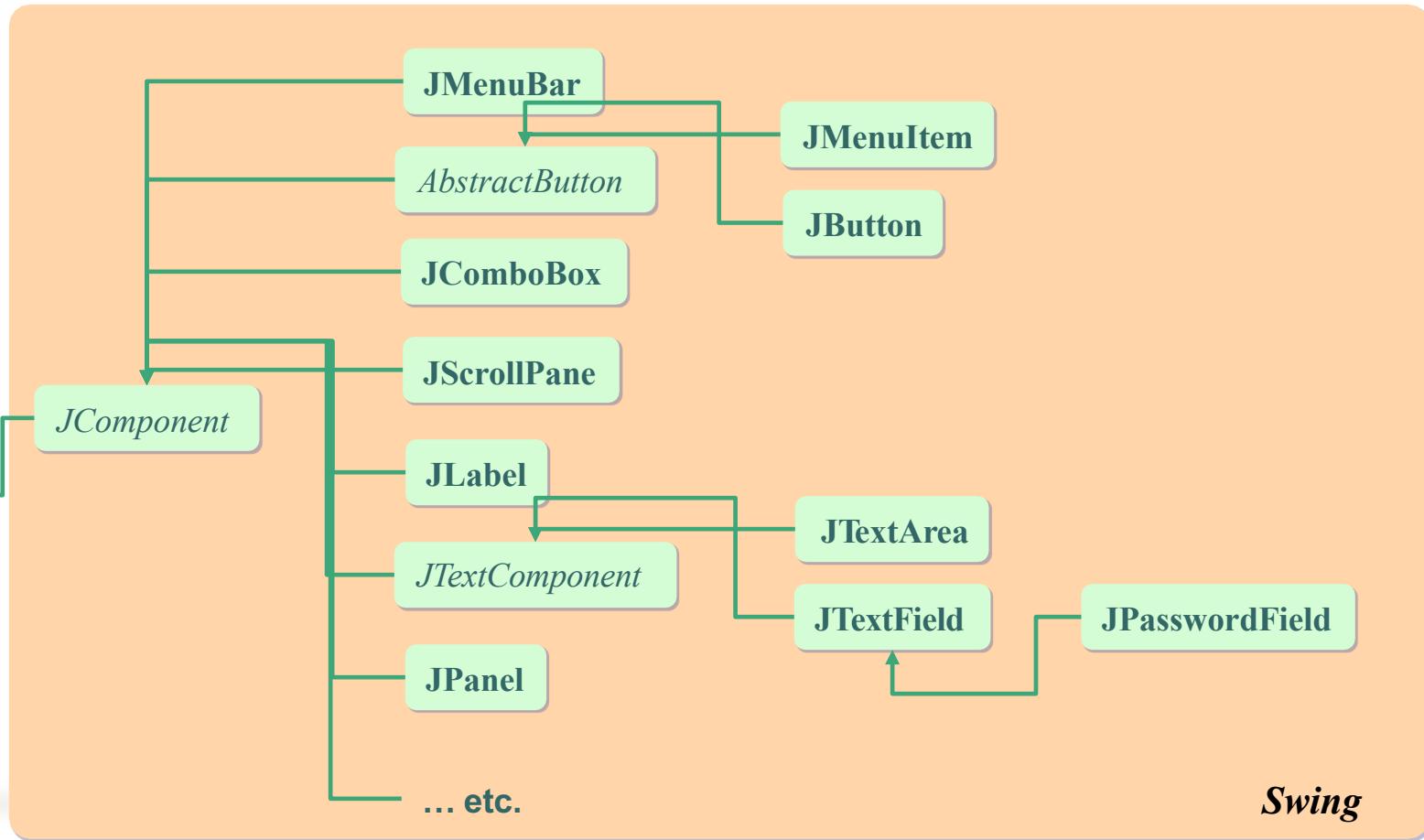
GUI

● Java GUI hierarchy



GUI

- Java GUI hierarchy



Swing Components

- Swing Components có nhiều mức khác nhau
- Các mức này qui định các Component trong 1 ứng dụng GUI kết hợp với nhau.
- Cách kết hợp này gọi là Swing *Containment Hierarchy*

Khái niệm container

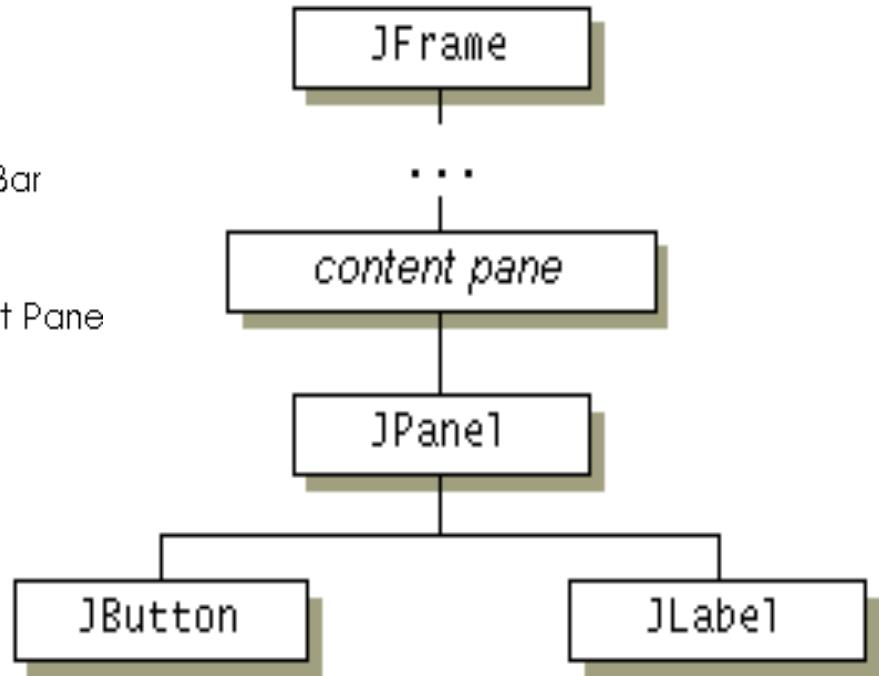
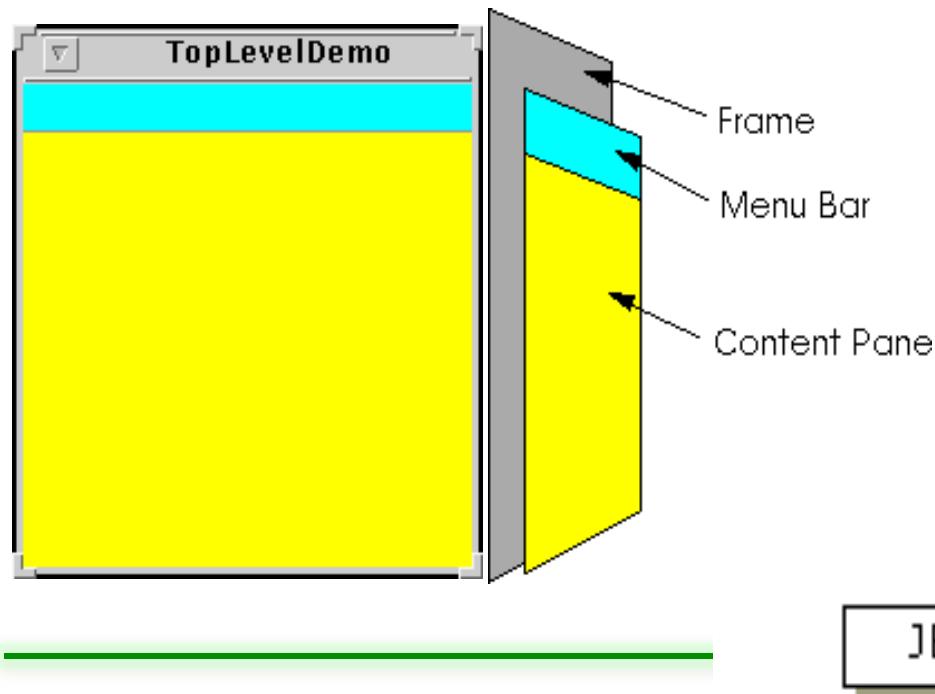
- ❖ Là thành phần mà có thể chứa các thành phần khác, có thể vẽ và tô màu.
 - ✓ Frame/JFrame, Panel/JPanel, Dialog/JDialog, ScrollPane/JScrollPane, ...
- ❖ Gắn component vào khung chứa
 - ✓ `containerObj.add(compObj);`
- ❖ Lấy thông tin của component
 - ✓ `objectName.get...();`
- ❖ Gán thông tin cho component
 - ✓ `objectName.set...();`

Nguyên tắc xây dựng GUI trong java

- ❖ Lựa chọn 1 container: Frame/JFrame, Window/JWindow, Dialog/JDialog, ...
- ❖ Tạo các điều khiển: (buttons, text areas..)
- ❖ Đưa các điều khiển vào vùng chứa
- ❖ Sắp xếp các điều khiển(layout)
- ❖ Thêm các xử lý sự kiện (Listeners)

Containment Hierarchy

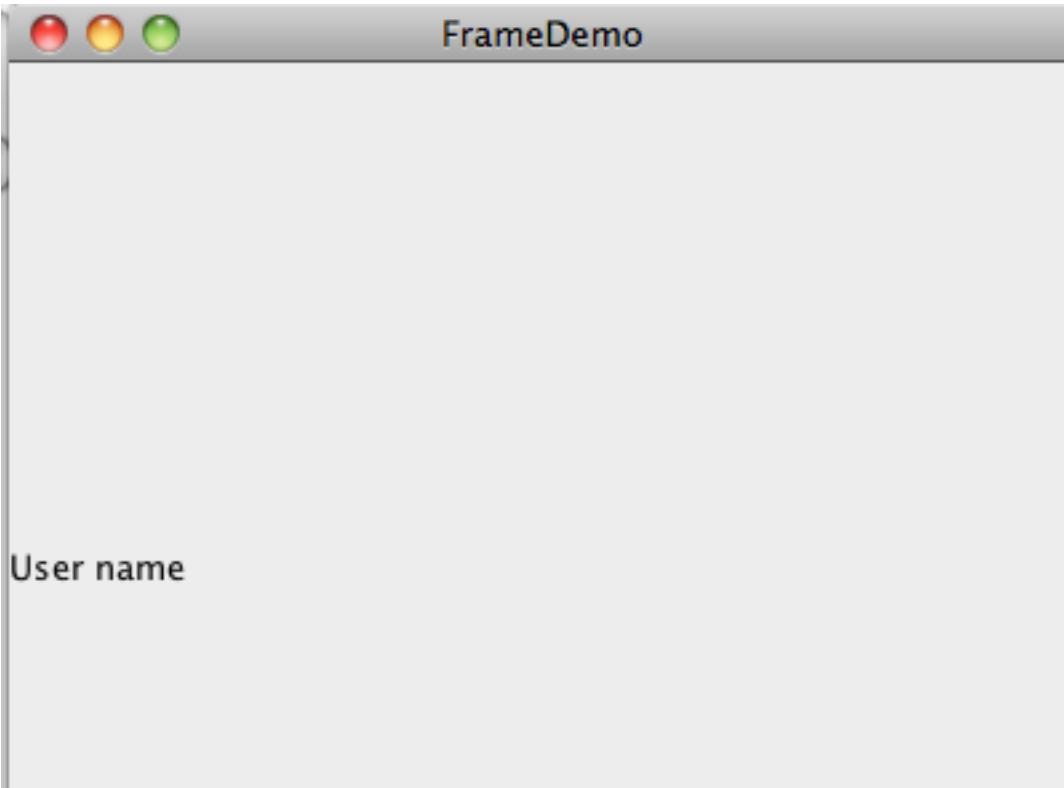
Top-level container: **JFrame**, **JDialog**, **JApplet**



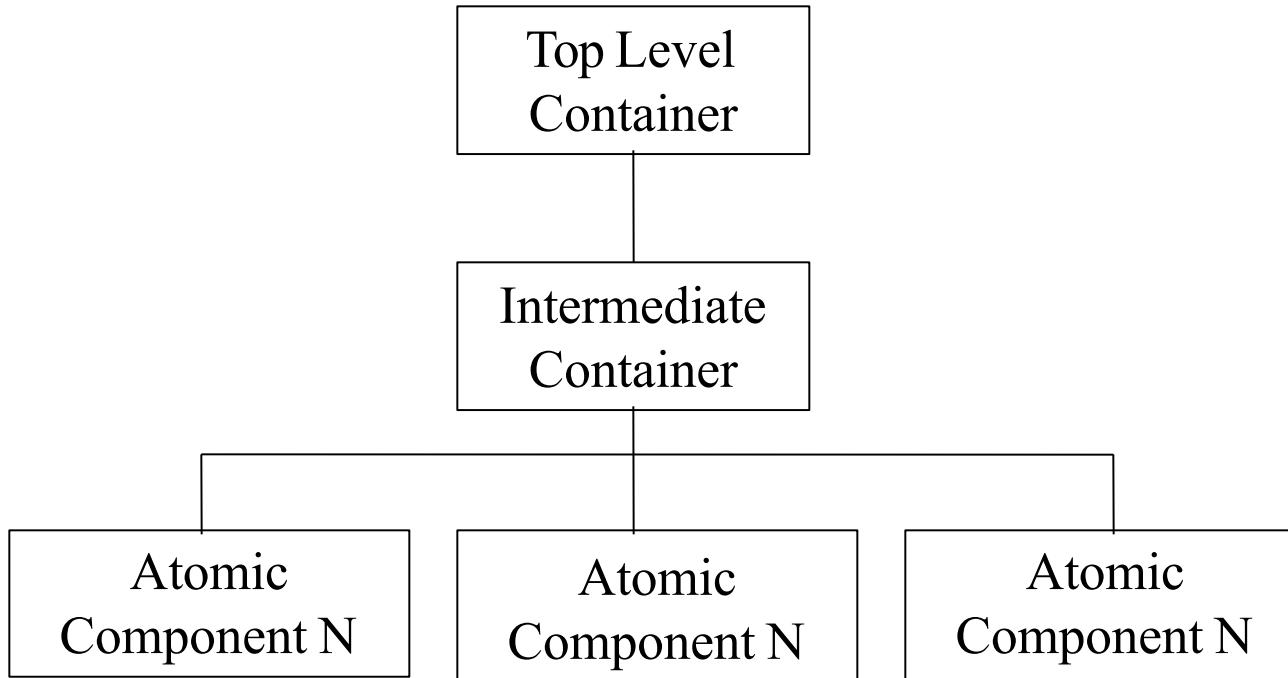
Ví dụ dùng Frame – cách 1

```
private static void createAndShowGUI() {  
    JFrame frame = new JFrame("FrameDemo");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //đóng  
    toàn bộ frame liên quan  
    frame.setSize(400,400); // Kích thước của Frame  
    frame.setLocation(200,200); // vị trí hiển thị  
    // setLocationRelativeTo(null) ; hiển thị giữa màn hình  
    JLabel emptyLabel = new JLabel("User name");  
    frame.add(emptyLabel); //thêm control vào Frame  
    frame.setVisible(true); //hiển thị Frame  
}  
  
public static void main(String[] args) {  
    createAndShowGUI();}
```

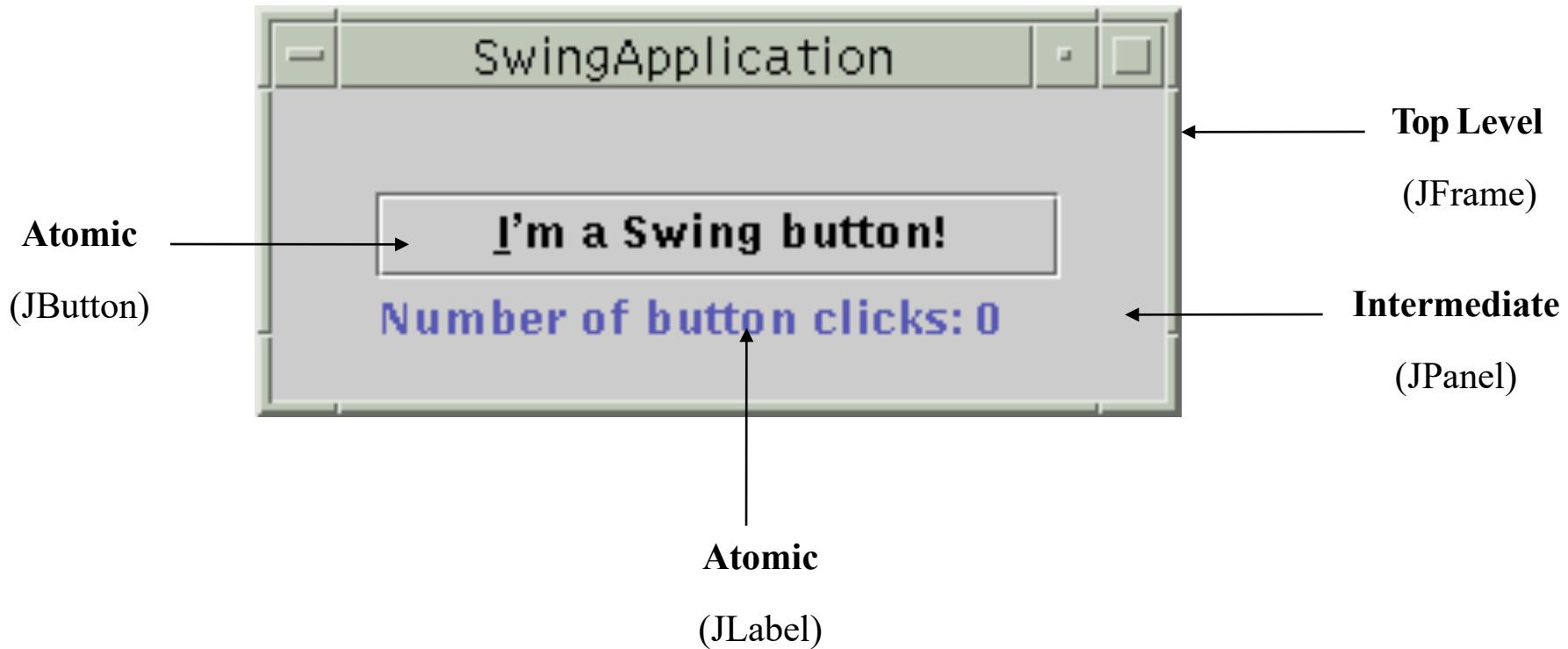
Kết quả



Containment Hierarchy



Containment Hierarchy



Containment Hierarchy

- Top Level Containers
 - Là nơi để hiển thị các **Component** khác
 - *Ví dụ:*
 - JFrame
 - JDialog
 - JApplet

JFrame

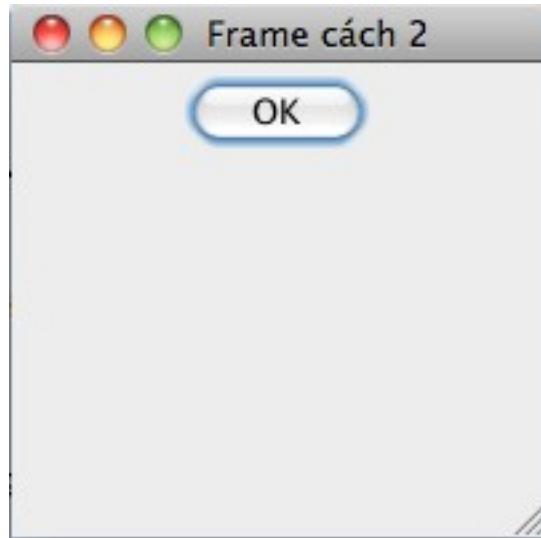
- Đây là cửa sổ chính, dùng để chứa các thành phần giao diện khác. Đóng vai trò là 1 container
- JFrame thường dùng để tạo ra cửa sổ trong chương trình swing
- Hàm dựng
 - JFrame()
 - JFrame(String title)
- Các thành phần đồ họa được đưa vào content pane, không đưa trực tiếp vào đối tượng JFrame
- Ví dụ:
 - frame.getContentPane().add(b);

Ví dụ dùng Frame – cách 2

✧ *Thừa kế từ lớp Frame*

```
public class Cach_2 extends JFrame {  
    public Cach_2(){setLayout(new FlowLayout());  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); setTitle("Frame  
cách 2");  
        setSize(200,200); setLocationRelativeTo(null);  
        JButton button = new JButton("OK");  
        button.setSize(20,30);  
        add(button); setVisible(true);}  
    public static void main(String[] args) { new Cach_2();  
}}
```

Kết quả



Ví dụ dùng JDialog

```
public class Dialog1
{ public Dialog1()
{
    JDialog jd = new JDialog();
    jd.setTitle("Ví dụ Dialog");
    jd.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    jd.setLayout(new FlowLayout());
    JButton ok= new JButton("OK");
    jd.setSize(200,200);
    jd.setLocationRelativeTo(null);
    jd.add(ok);
    jd.setVisible(true);
}
public static void main(String[] args
{
    new Dialog1();
}
}
```

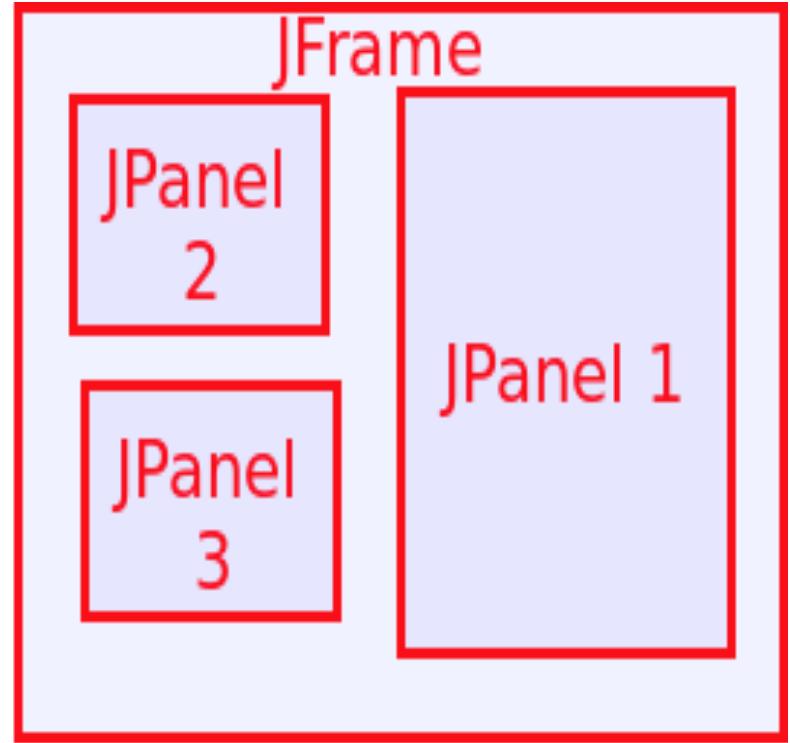
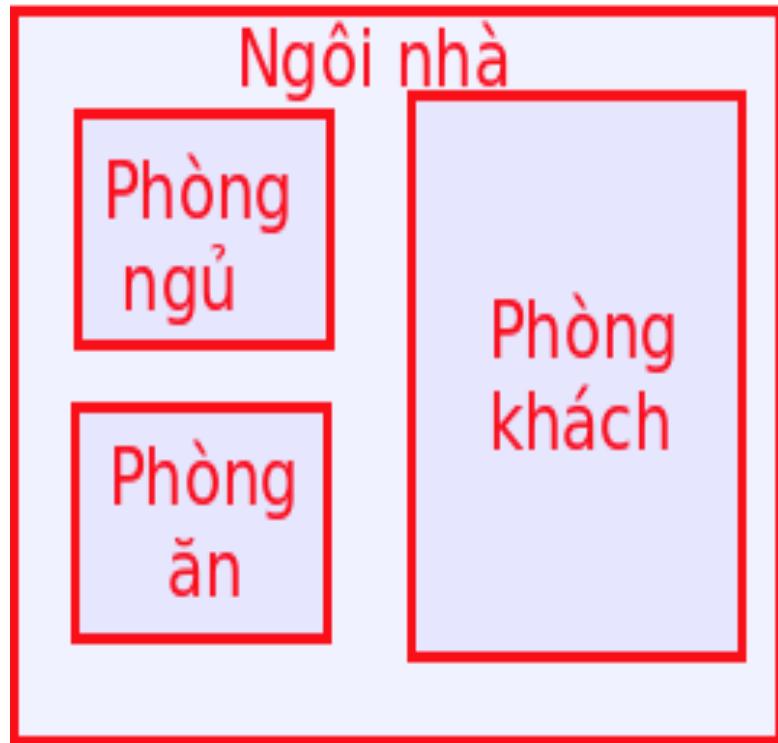
Kết quả JDialog



Containment Hierarchy

- Intermediate Containers
 - Dùng để xác định vị trí của các *Atomic Components*
 - Hoặc cung cấp cơ chế để tương tác với khung chứa nó
 - *Ví dụ:*
 - JPanel
 - JScrollPane
 - JTabbedPane

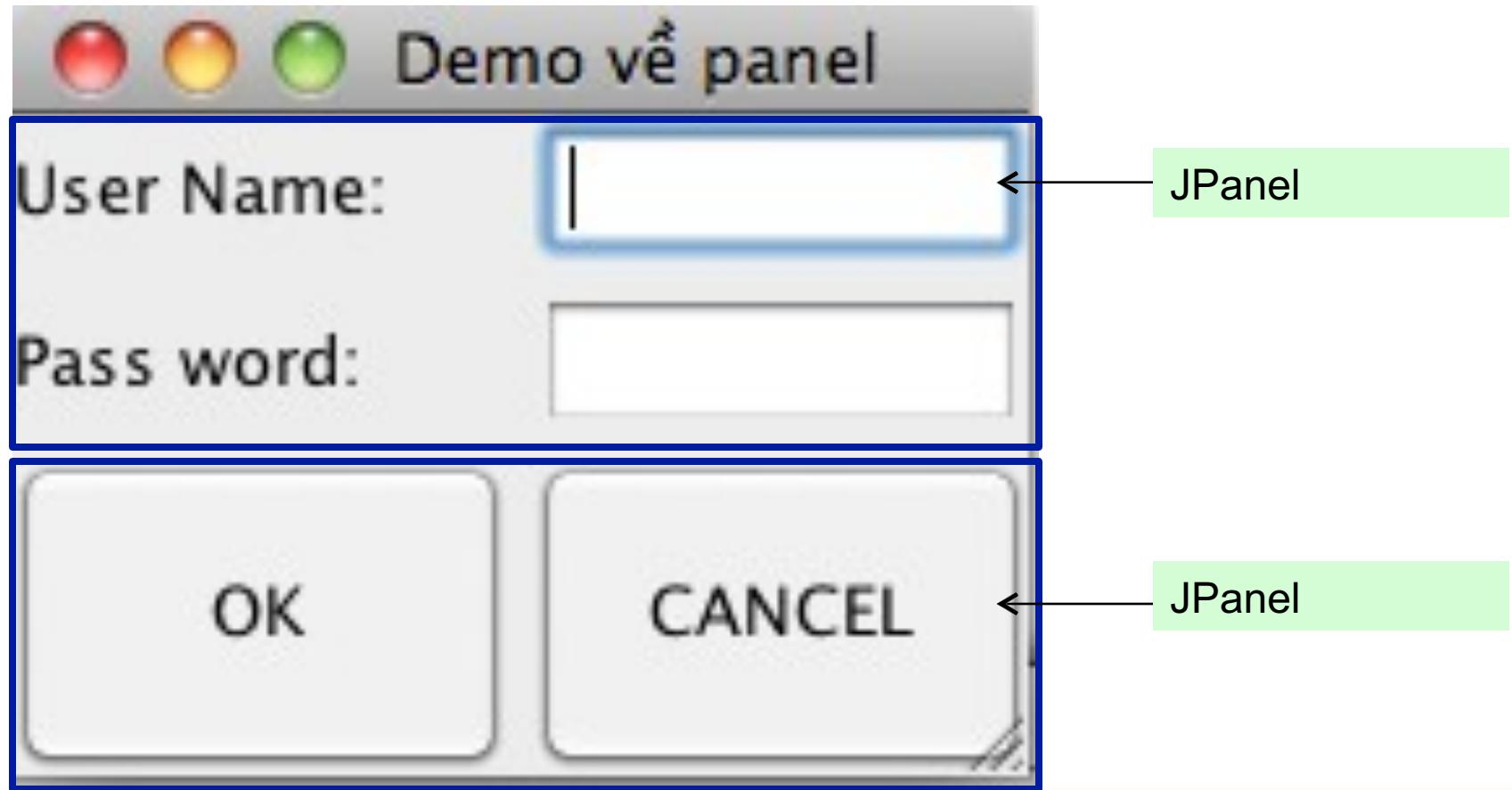
JPanel



JPanel

- JPanel là container trung gian dùng để chứa các component khác
- Thường dùng để phân chia các component trong ứng dụng
- Layout mặc định là FlowLayout
- Các hàm dựng:
 - JPanel()
 - JPanel(LayoutManager lm)

Ví dụ - JPanel



Ví dụ - JPanel

```
public Panel_01()
{
    JFrame fr= new JFrame("Demo về panel");
    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    fr.setLayout(new GridLayout(2,1,5,5));
    //fr.setSize(200,200);
    JPanel pn=new JPanel(new GridLayout(1,2,5,5));

    JButton ok=new JButton("OK");
    JButton cancel=new JButton("CANCEL");

    pn.add(ok);
    pn.add(cancel);
```

Ví dụ - JPanel

```
JPanel pn1=new JPanel(new GridLayout(2,1,5,5));
    JTextField username=new JTextField("");
    JTextField password=new JTextField("");
    JLabel user=new JLabel("User Name:");
    JLabel pass=new JLabel("Pass word:");
    pn1.add(user);
    pn1.add(username);
    pn1.add(pass);
    pn1.add(password);
    fr.getContentPane().add(pn1);
    fr.getContentPane().add(pn);
    fr.setLocationRelativeTo(null);
    fr.pack();
    fr.setVisible(true);
}
```

Project name: **MyPanel**

Containment Hierarchy

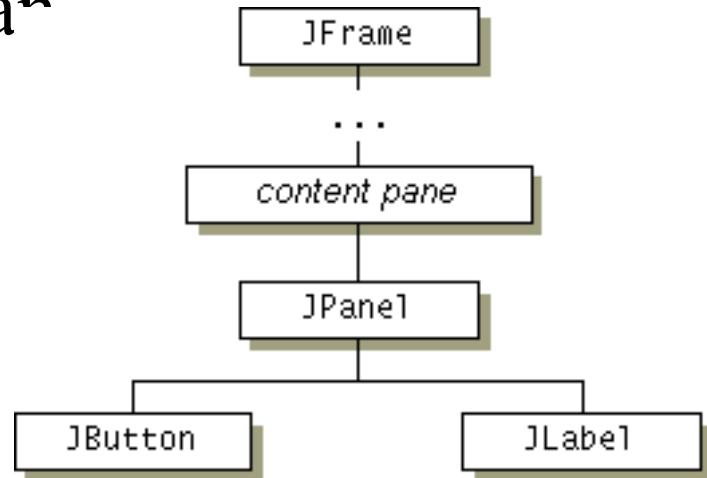
- Atomic Components

- Không chứa các Components khác
- Dùng để cung cấp thông tin đến người sử dụng
- Hoặc lấy thông tin từ người sử dụng
- *Ví dụ:*
 - ❖ JButton
 - ❖ JLabel
 - ❖ JComboBox
 - ❖ JTextField
 - ❖ JTable

Containment Hierarchy

- Mọi Top Level Container có 1 intermediate Container gọi là *Content Pane*
- Các atomic component phải gắn vào content pane
- Ví dụ:

```
JFrame frame = new JFrame(...); JPanel  
pane = new JPanel();  
frame.getContentPane().add(pane);
```



Khái niệm Layout Manager

- ❖ Các loại layout khác nhau:
 - ✓ Flow Layout
 - ✓ Border Layout
 - ✓ Grid Layout
 - ✓ GridBag Layout
 - ✓ Null Layout
 - ✓ ...
- ❖ Trình quản lý layout được thiết lập bằng cách gọi phương thức ‘**setLayout()**’

FlowLayout

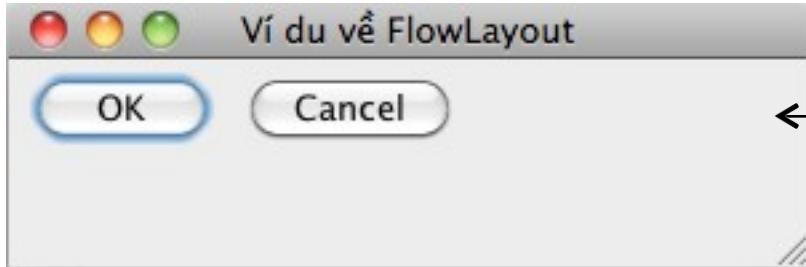
- ❖ Là một lớp dùng để quản lý việc sắp xếp các Component
- ❖ Với FlowLayout các thành phần tự động được sắp xếp từ góc trái trên đến góc phải dưới của màn hình

FlowLayout

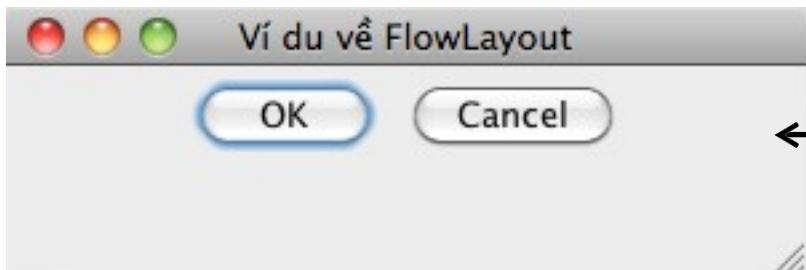
- ❖ Lớp FlowLayout có các constructor sau:

- public FlowLayout()
- public FlowLayout(int alignment)
- public FlowLayout(int alignment, int horizontalGap, int verticalGap)
- alignment có các giá trị:
 - FlowLayout.LEFT
 - FlowLayout.CENTER
 - FlowLayout.RIGHT.
 - Thông số horizontalGap và verticalGap xác định khoảng cách theo chiều ngang và đứng của các Component

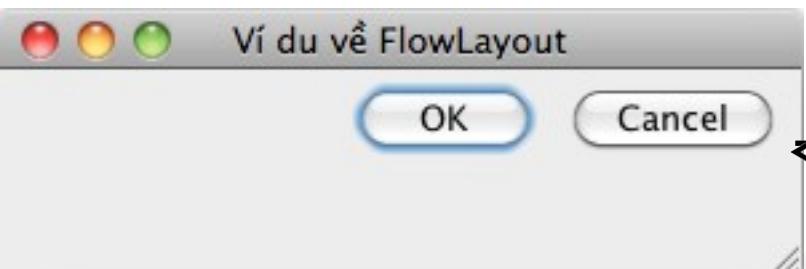
FlowLayout



```
fr.setLayout(new FlowLayout  
 (FlowLayout.LEFT));
```



```
fr.setLayout(new FlowLayout  
 (FlowLayout.CENTER));
```



```
fr.setLayout(new FlowLayout  
 (FlowLayout.RIGHT));
```

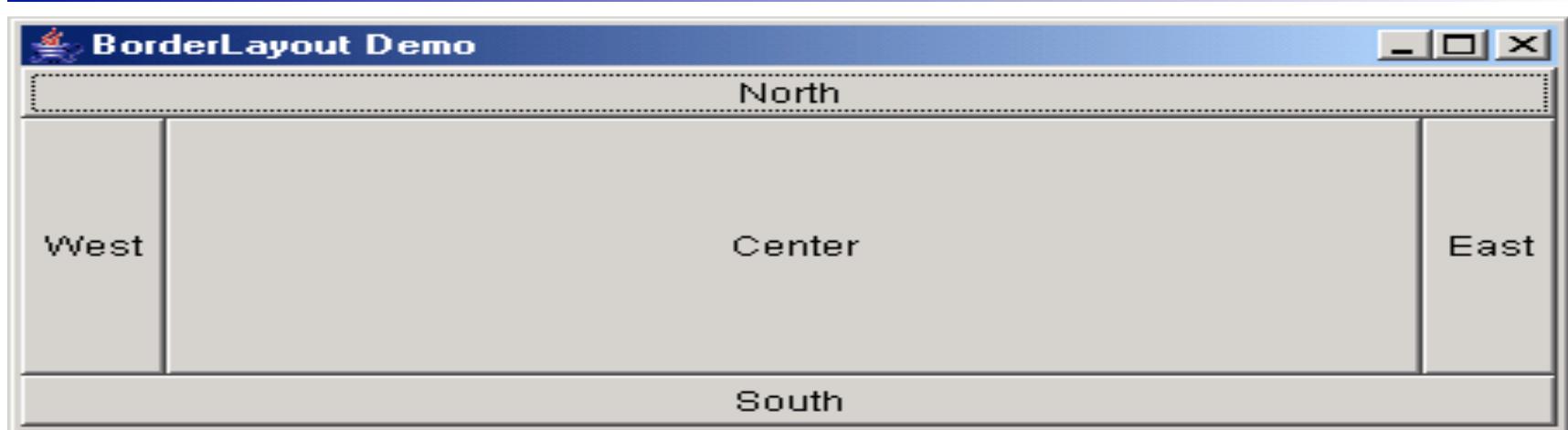
Ví dụ

```
public Flowlayout1()
{
    JFrame fr=new JFrame("Ví du về FlowLayout");
    fr.setSize(300,100);
    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fr.setLayout(new FlowLayout(FlowLayout.RIGHT));
    JButton ok=new JButton("OK");
    JButton cancel=new JButton("Cancel");
    fr.getContentPane().add(ok);
    fr.getContentPane().add(cancel);
    fr.setVisible(true);
}
```

BorderLayout

- ❖ Là trình quản lý layout mặc định cho Window, Frame và Dialog
- ❖ Trình quản lý này có thể xếp xắp đến 5 thành phần trong container
- ❖ Các thành phần có thể được đặt vào 5 hướng NORTH, EAST, SOUTH, WEST và CENTER của container
- ❖ **Ví dụ:** Để thêm một thành phần vào vùng North của container
 - ✓ `Button b1= new Button(“North Button”);`
 - ✓ `setLayout(new BorderLayout());`
 - ❖ Thêm một thành phần vào container
 - ✓ `add(b1, BorderLayout.NORTH);`

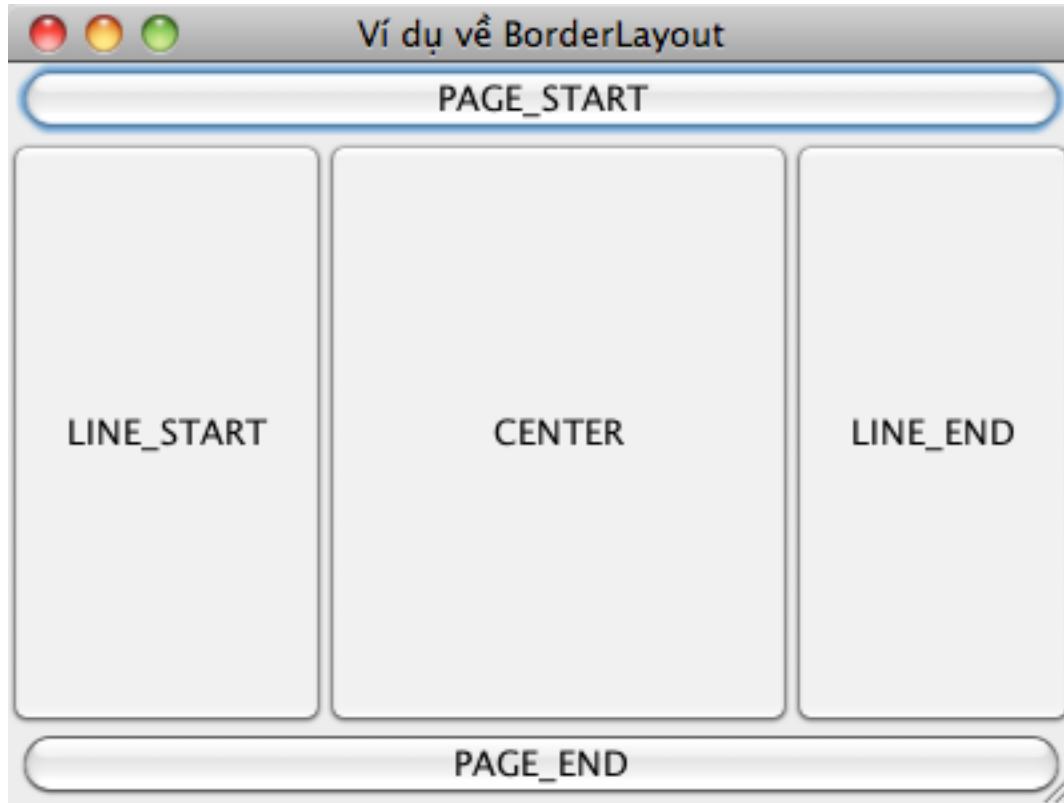
BorderLayout



Code chương trình

```
public BorderLayout()
{
    JFrame fr=new JFrame("Ví dụ về BorderLayout");
    fr.setSize(400,300);
    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fr.setLayout(new BorderLayout());
    JButton buttonNorth= new JButton("Bắc");
    JButton buttonSouth= new JButton("Nam");
    JButton buttonWest= new JButton("Đông");
    JButton buttonEast= new JButton("Tây");
    JButton buttonCenter= new JButton("Giữa");
    fr.add(buttonWest,BorderLayout.WEST);
    fr.add(buttonEast, BorderLayout.EAST);
    fr.add(buttonSouth, BorderLayout.SOUTH);
    fr.add(buttonNorth, BorderLayout.NORTH);
    fr.add(buttonCenter,BorderLayout.CENTER);
}
```

Định dạng LINE, PAGE, CENTER



Code chương trình

```
public BorderLayout(){  
    JFrame fr=new JFrame("Ví dụ về BorderLayout");  
    fr.setSize(400,300);  
    fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    fr.setLayout(new BorderLayout());  
    JButton buttonNorth= new JButton("PAGE_START");  
    JButton buttonSouth= new JButton("PAGE_END");  
    JButton buttonWest= new JButton("LINE_START");  
    JButton buttonEast= new JButton("LINE_END");  
    JButton buttonCenter= new JButton("CENTER");  
    fr.add(buttonWest,BorderLayout.LINE_START);  
    fr.add(buttonEast, BorderLayout.LINE_END);  
    fr.add(buttonSouth, BorderLayout.PAGE_END);  
    fr.add(buttonNorth, BorderLayout.PAGE_START);  
    fr.add(buttonCenter,BorderLayout.CENTER);  
    fr.setLocationRelativeTo(null);  
    fr.setVisible(true);}
```

GridLayout



GridLayout

- ❖ 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
- ❖ Tạo và thêm GridLayout
 - **GridLayout()**: Tạo GridLayout với 1 hàng, 1 cột
 - **GridLayout(int rows, int cols)**: Tạo GridLayout với số hàng và số cột được chỉ định
 - **GridLayout(int rows, int cols, int hgap, int vgap)**: Tạo GridLayout với số hàng, số cột, khoảng cách giữa các hàng và các cột.
- ❖ *Thêm GridLayout: setLayout(layout);*

Code chương trình

```
public GridLayout()
{
    JFrame jf=new JFrame("Demo gridLayout");
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jf.setSize(300,200);
    jf.setLayout(new GridLayout(3,3,5,5));
    JButton jb7= new JButton("7");
    jf.add(jb7);

    JButton jb8= new JButton("8");
    jf.add(jb8);
    JButton jb9= new JButton("9");
    jf.add(jb9);

    JButton jb4= new JButton("4");
    jf.add(jb4);
```

Code chương trình

```
JButton jb5= new JButton("5");
jf.add(jb5);
```

```
JButton jb6= new JButton("6");
jf.add(jb6);
```

```
JButton jb1= new JButton("1");
jf.add(jb1);
```

```
JButton jb2= new JButton("2 ");
jf.add(jb2);
```

```
JButton jb3= new JButton("3");
jf.add(jb3);
```

```
jf.setLocationRelativeTo(null);
jf.setVisible(true);
```

```
}
```

Projectname: GridLayout

GridBagLayout



GridLayout

- ❖ Bố trí các thành phần một cách chính xác
- ❖ Các thành phần không cần có cùng kích thước
- ❖ Các thành phần được sắp xếp trong một lưới chứa các dòng và các cột
- ❖ Thứ tự đặt các thành phần không tuân theo hướng từ trái-sang-phải và trên-xuống-dưới
- ❖ Hàm constructor
 - ✓ `GridLayout gb = new GridLayout();`
- ❖ Lớp ‘**GridLayoutConstraints**’ lưu trữ tất cả các thông tin mà lớp GridLayout yêu cầu: Vị trí và kích thước mỗi thành phần

GridBagLayout

- ❖ ***gridx*** và ***gridy***: Vị trí ô trên khung lưới của đối tượng mà ta đưa vào: **ví dụ**: $\text{gridx}=1$, $\text{gridy}=2$ tức là đặt đối tượng tại ô cột 1, hàng 2
- ❖ ***gridwidth*, *gridheight***: Số ô chiếm của đối tượng theo chiều ngang (width), và theo chiều dọc (height)

GridLayout

- ❖ *fill*: giả sử một đối tượng đưa vào không khít hết các ô nó chiếm thì có thể dùng các thông số sau:
 - Giá trị ***GridBagConstraints.NONE*** tức là giữ nguyên không đổi kích thước đối tượng.
 - Giá trị ***GridBagConstraints.VERTICAL*** giãn chiều cao của đối tượng cho khít
 - Giá trị ***GridBagConstraints.HORIZONTAL*** giãn chiều ngang của đối tượng cho khít
 - Giá trị ***GridBagConstraints.BOTH*** giãn cả 2 chiều của đối tượng cho khít

GridBagLayout

- ❖ *ipadx, ipady*: tăng kích thước hai bên trái phải (hoặc trên dưới) đối tượng thêm ipadx (ipady) pixel khi kích thước khung chứa thay đổi
- ❖ *insets*: đối tượng thuộc lớp Insets chỉ ra khoảng cách từ phía so với đối tượng khác. *Ví dụ*: constraint.insets = new Insets (5,5,5,5) thì các đối tượng sẽ cách nhau 5 pixel các chiều
- ❖ *anchor*: Xác định vị trí hiển thị so với ô đặt nó. Các giá trị: NORTH, NORTHWEST, NORTHEAST, WEST, EAST, SOUTH, SOUTHWEST, SOUTHEAST.
- ❖ *weightx, weighty*: Sự phân phối của khoảng trống trong GridBagLayout. Giá trị mặc định cho các biến này là 0.

Ví dụ

- Viết chương trình thể hiện giao diện sau:

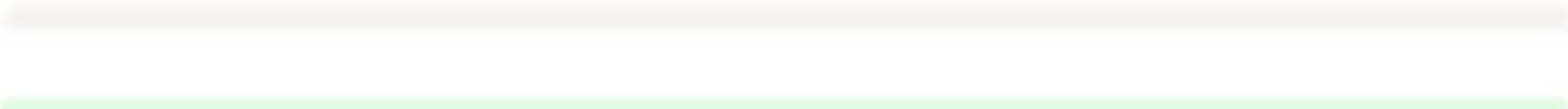


Code chương trình

```
public class MyGridBagLayout1 extends JFrame {  
    Button B2 = new Button("NEXT");  
    Button B1 = new Button("East");  
    GridBagLayout gridbag;  
    public MyGridBagLayout1()  
    {  
        super("Ví dụ GridBagLayout_ ví dụ 2");  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
        setSize(300,200);  
        setBackground(Color.yellow);  
        gridbag = new GridBagLayout();  
        GridBagConstraints c = new GridBagConstraints();  
        setLayout(gridbag);
```

Code chương trình

```
c.weightx = 0;  
c.gridx = 1;  
c.gridy = 0;  
c.fill = GridBagConstraints.HORIZONTAL;  
c.anchor = GridBagConstraints.NORTHEAST;  
gridbag.setConstraints(B1,c);  
add(B1);  
c.weightx = 1;  
c.weighty = 1;
```



Code chương trình

```
c.gridx = 0;  
c.anchor = GridBagConstraints.WEST;  
c.fill = GridBagConstraints.BOTH;  
gridbag.setConstraints(B2,c);  
//B2.setBackground(Color.pink);  
add(B2);  
setLocationRelativeTo(null);  
setVisible(true);  
  
}  
public static void main(String[] args) {  
    new MyGridBagLayout1();  
}
```

Ví dụ 2

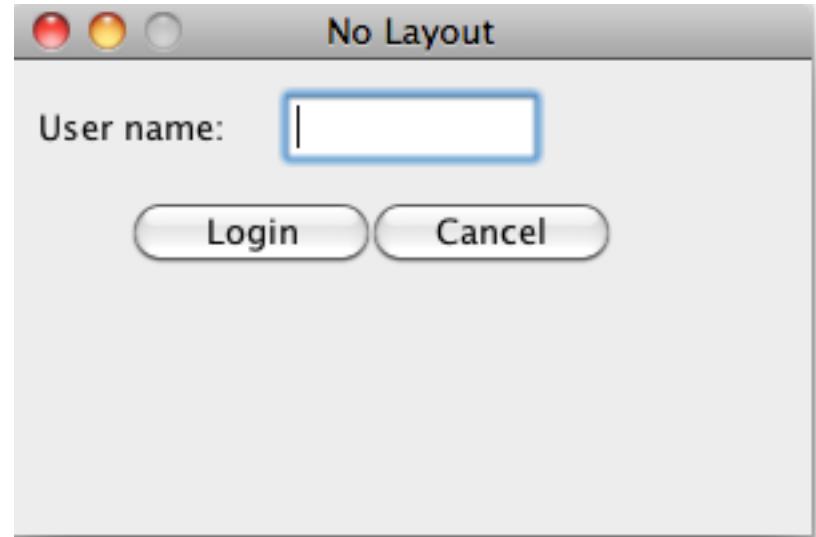


Name project: MyGridLayout

NullLayout

- Tự do trong việc định vị trí và kích thước của các components

```
JFrame fr = new JFrame("No Layout");  
fr.setLayout(null);
```



Các hàm liên quan

- ❖ Định kích thước cho đối tượng
 - Object.setSize(int x, int y);
- ❖ Xác định vị trí hiển thị đối tượng
 - Object.setLocation(int x, int y);

Code chương trình (tt)

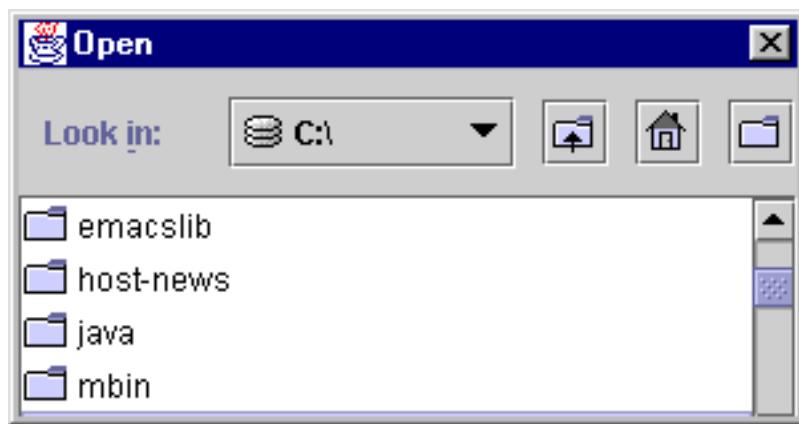
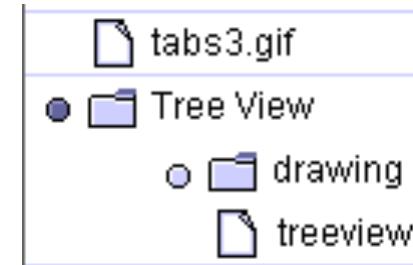
```
public NoLayout()
{
    JFrame jf=new JFrame("No Layout");
    jf.setSize(300,200);
    jf.setResizable(false);
    jf.setLayout(null);
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JButton Ok= new JButton("Login");
    JButton Cancel= new JButton("Cancel");
    JLabel name=new JLabel("User name:");
    JTextField NameT = new JTextField();
    NameT.setSize(100,30);
    name.setSize(100, 30);
    Ok.setSize(100,30);
    Cancel.setSize(100,30);
```

Code chương trình

```
name.setLocation(10,10);
NameT.setLocation(100,10);
Ok.setLocation(40,50);
Cancel.setLocation(130,50);
jf.add(name);
jf.add(NameT);

jf.add(Ok);
jf.add(Cancel);
jf.setLocationRelativeTo(null);
jf.setVisible(true);
}
```

Swing Components



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

- Monday
- Monday
- Tuesday
- Wednesday
- Thursday
- Friday

Sử Dụng Swing Component

`javax.swing`

Class JComponent

`java.lang.Object`

└ `java.awt.Component`

 └ `java.awt.Container`

 └ `javax.swing.JComponent`

All Implemented Interfaces:

[ImageObserver](#), [MenuContainer](#), [Serializable](#)

Direct Known Subclasses:

[AbstractButton](#), [BasicInternalFrameTitlePane](#), [Box](#), [Box.Filler](#), [JColorChooser](#), [JComboBox](#), [JFileChooser](#),
[JInternalFrame](#), [JInternalFrame.JDesktopIcon](#), [JLabel](#), [JLayeredPane](#), [JList](#), [JMenuBar](#), [JOptionPane](#), [JPanel](#),
[JPopupMenu](#), [JProgressBar](#), [JRootPane](#), [JScrollBar](#), [JScrollPane](#), [JSeparator](#), [JSlider](#), [JSpinner](#), [JSplitPane](#),
[JTabbedPane](#), [JTable](#), [JTableHeader](#), [JTextComponent](#), [JToolBar](#), [JToolTip](#), [JTree](#), [JViewport](#)

Using Swing Components

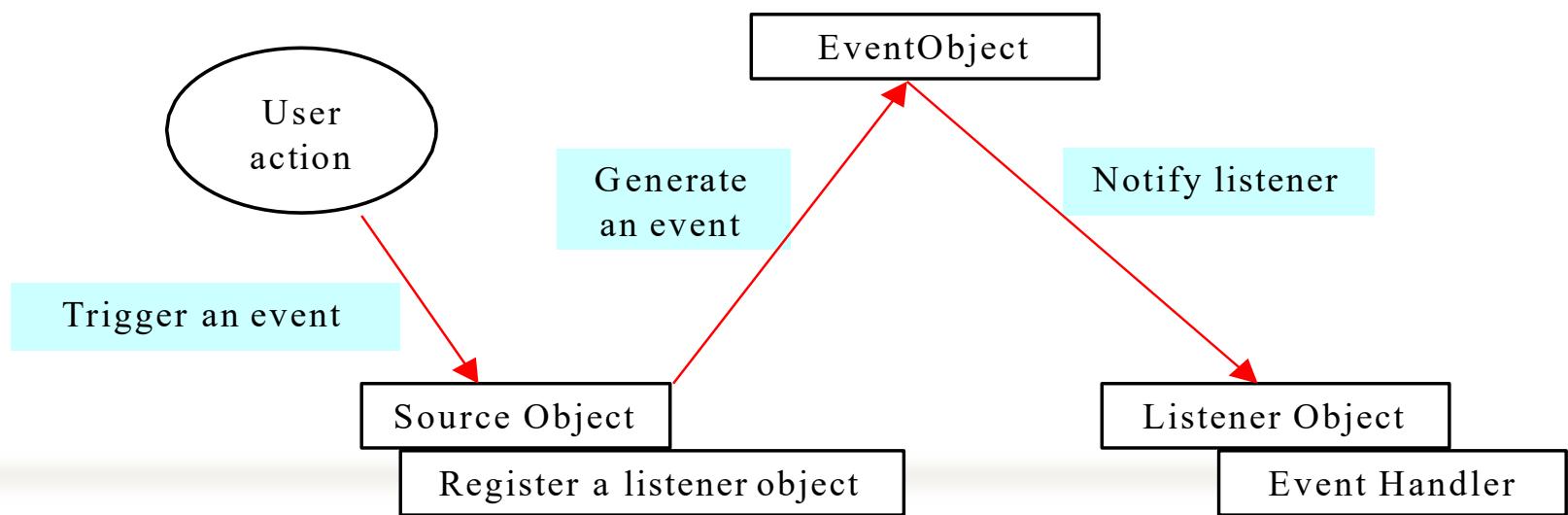
add, set, get method

- Phương thức dùng để gắn components:
`objectName.add(...);`
- Phương thức dùng để lấy thuộc tính:
`objectName.getxxx();`
- Phương thức dùng để gán thuộc tính:
`objectName.setxxx();`

Mô hình xử lý sự kiện

Có 3 yếu tố quan trọng trong mô hình xử lý sự kiện:

- ✓ Nguồn phát sinh sự kiện (event source)
- ✓ Sự kiện (event object)
- ✓ Bộ lắng nghe sự kiện (event listener)



Xử Lý Sự Kiện

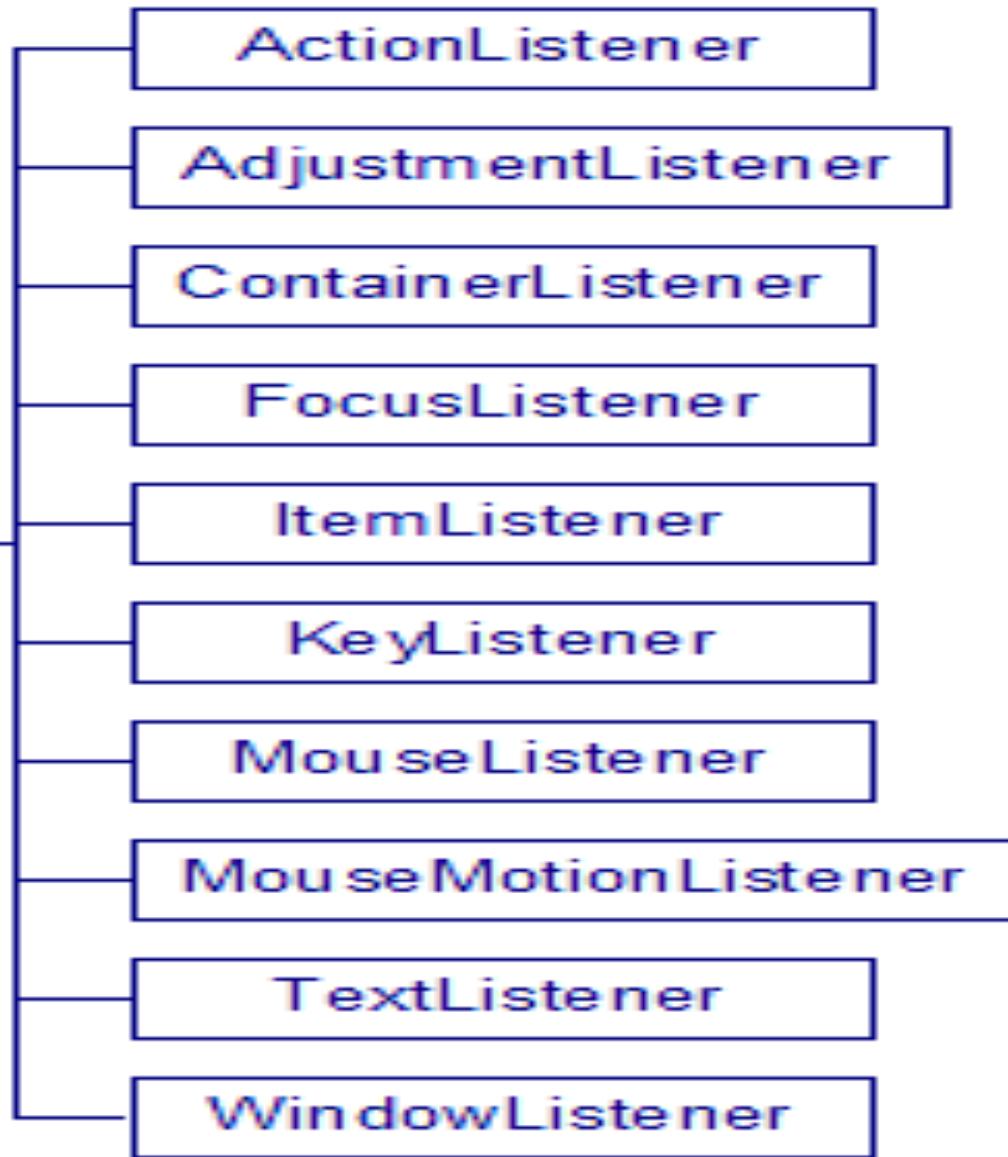
- Khai báo lớp xử lý sự kiện
`public class MyClass implements <Event>Listener`
- Cài đặt các phương thức trong listener interface.
Ví dụ: `ActionListener`
- `public void actionPerformed(ActionEvent e)`
 {
 Xử lý sự kiện;
 }
- Gắn bộ xử lý vào component
`someComponent.add<Event>Listener(instanceOfMyClass);`

Hành động, sự kiện, lắng nghe

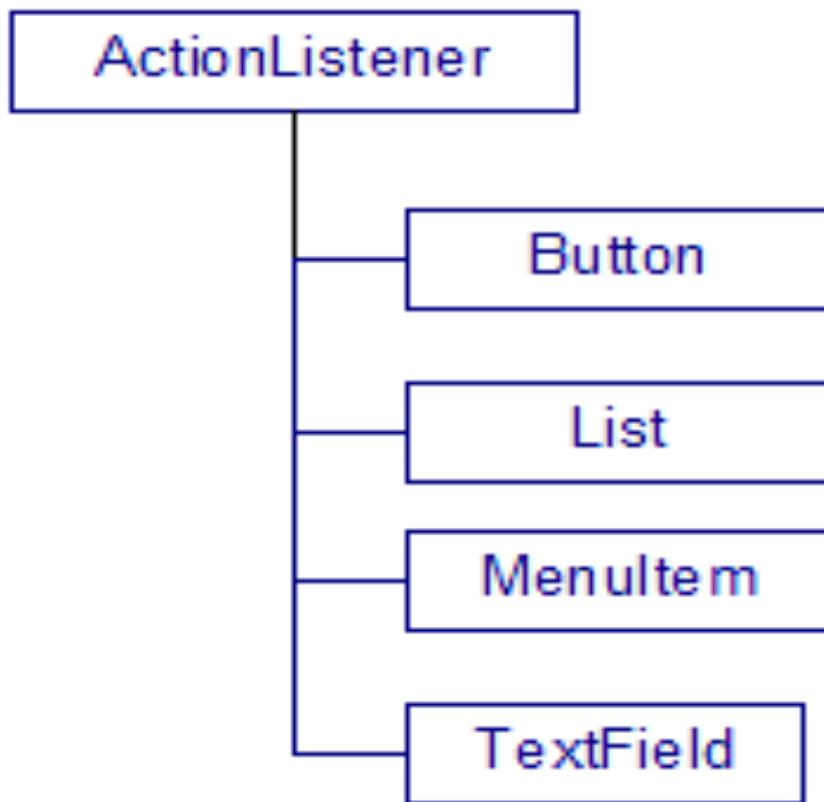
Đối tượng	Sự kiện	Bộ lắng nghe
Window, Frame, ...	WindowEvent	WindowListener
Button, MenuItem, ...	ActionEvent	ActionListener
TextComponent, ...	TextEvent	TextListener
List, ...	ActionEvent	ActionListener
...	ItemEvent	ItemListener
	ComponentEvent	ComponentListener
	MouseEvent	MouseListener
	MouseEvent	MouseMotionListener
	KeyEvent	KeyListener

Bộ lắng nghe sự kiện

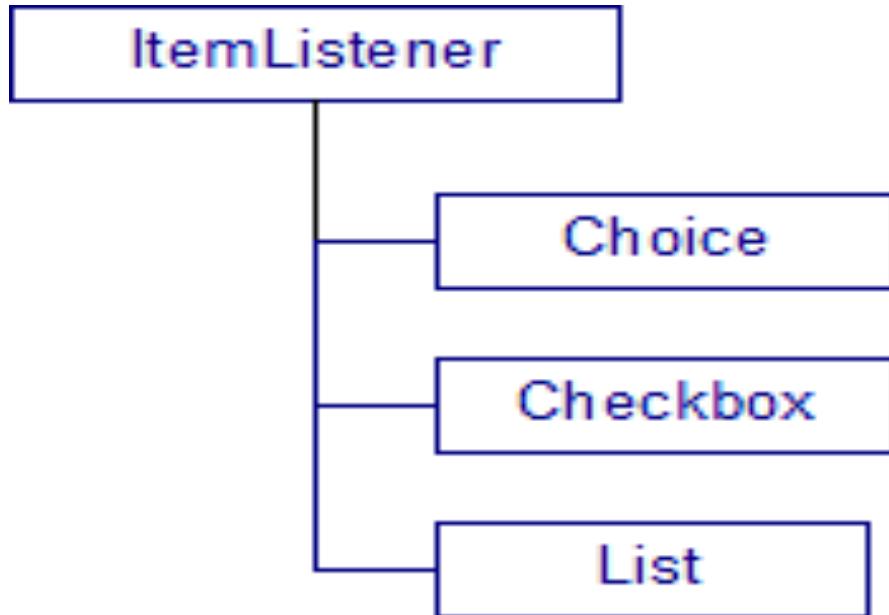
Event Listener



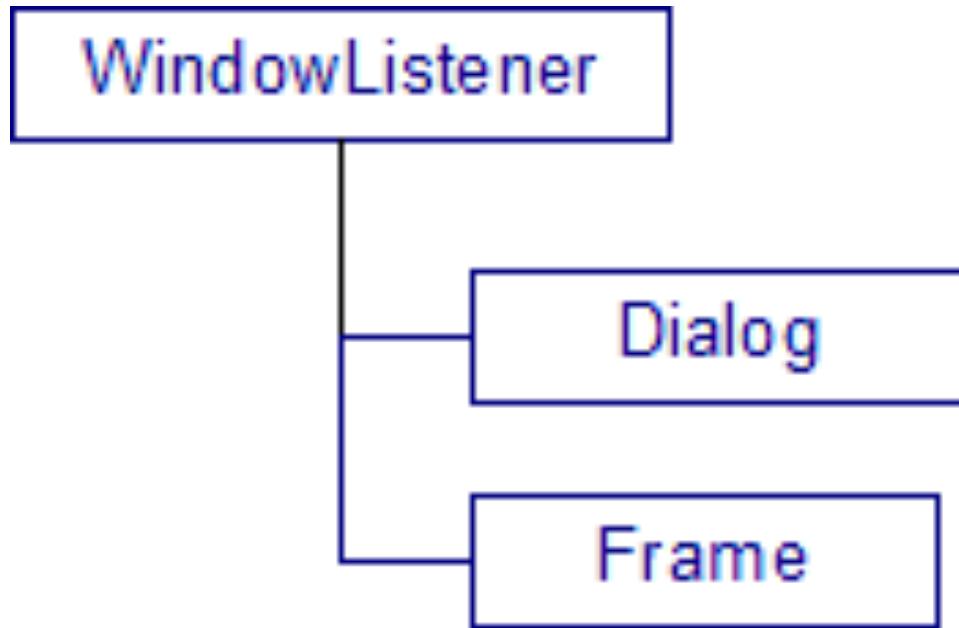
Các đối tượng sử dụng ActionListener



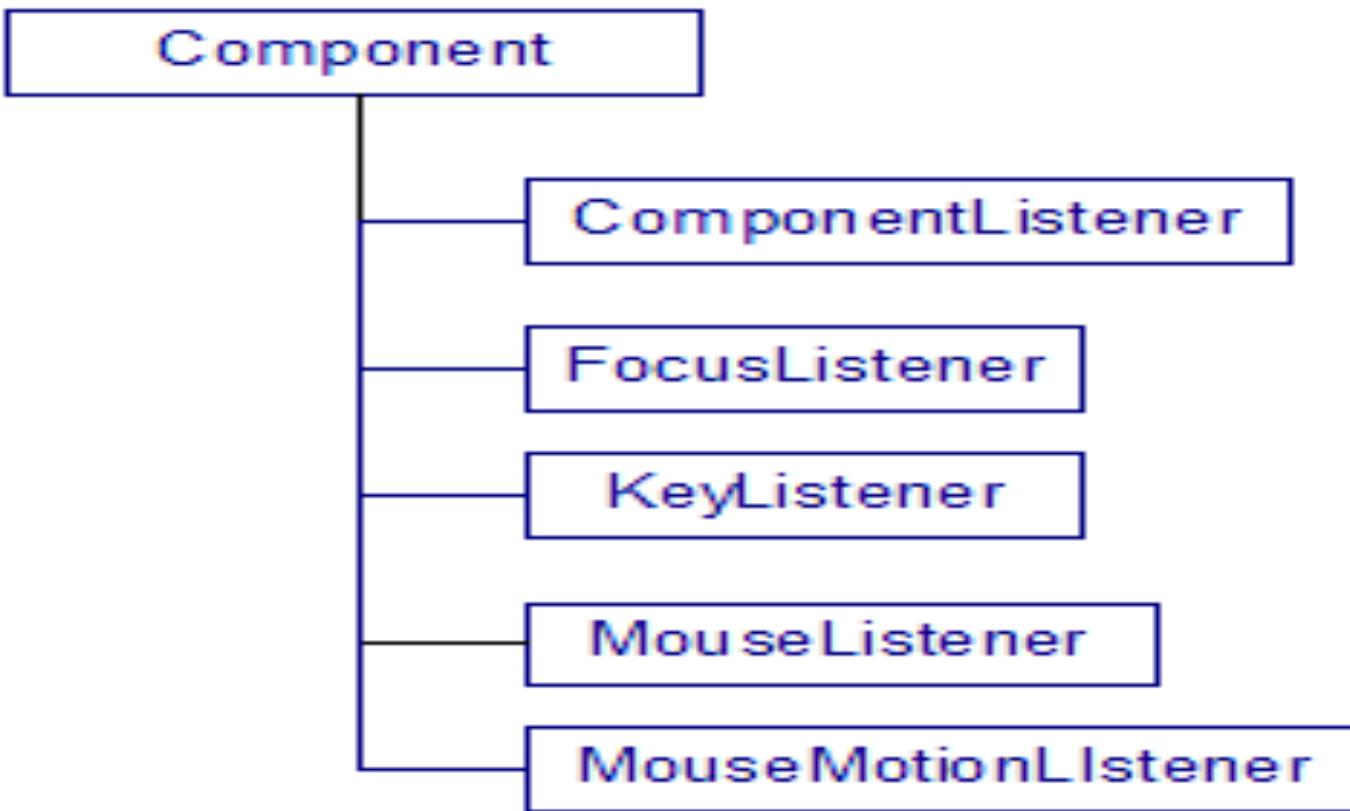
ItemListener



Window Listener



Component



Hàm liên quan đến sự kiện

- getSource(): Trả về nguồn sự kiện
- toString(): Trả về chuỗi tương đương với sự kiện

Hàm đăng ký bộ lắng nghe sự kiện

- **Button**
 - ✧ public **void** addActionListener(ActionListener a)
- **MenuItem**
 - ✧ public **void** addActionListener(ActionListener a)
- **TextField**
 - ✧ public **void** addActionListener(ActionListener a)
 - ✧ public **void** addTextListener(TextListener a)
- **TextArea**
 - ✧ public **void** addTextListener(TextListener a)

Hàm đăng ký bộ lắng nghe sự kiện

- **Checkbox**
 - ✧ public void addItemListener(ItemListener a)
- **Choice**
 - ✧ public void addItemListener(ItemListener a)
- **List**
 - ✧ public void addActionListener(ActionListener a)
 - ✧ public void addItemListener(ItemListener a)

JButton

- ❖ *Button* là một thành phần gây ra một sự kiện hành động khi được kích chuột.
- ❖ Các constructor của JButton:

JButton()

JButton(String text)

JButton(String text, Icon icon)

JButton(Icon icon)

Các thuộc tính JButton

- ❖ text
- ❖ icon
- ❖ mnemonic
- ❖ Horizontal Alignment
- ❖ Vertical Alignment
- ❖ Horizontal TextPosition
- ❖ Vertical TextPosition

Ví dụ

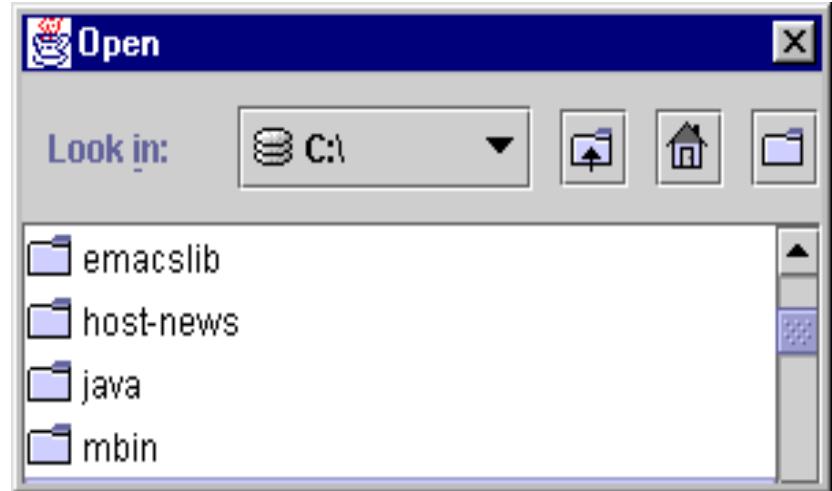
- Viết chương trình có giao diện sau:



- Khi nhấn chọn nút “Red” thì màu nền đổi màu đỏ, khi click chọn nút “Blue” chương trình đổi màu nền thành màu xanh

SWING Dialog Boxes

- Nhận thông tin từ người sử dụng
 - số liệu,...
 - danh mục tập tin,...
- Hiển thị kết quả
 - hiển thị thông tin cảnh báo
 - in kết quả lên màn hình,...



JOptionPane

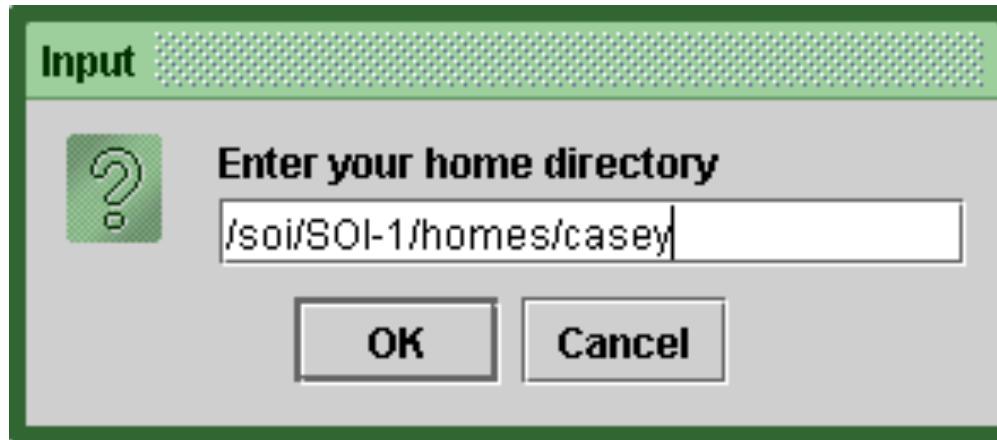
```
JOptionPane.showInputDialog("Enter your home directory");
```



JOptionPane

Static method call

```
JOptionPane.showInputDialog("Enter your home directory");
```



JOptionPane

- ❖ `showConfirmDialog` Asks a confirming question, like yes/no/cancel.
 - ❖ `showInputDialog`
 - ❖ `showMessageDialog`
 - ❖ `showOptionDialog`
-
- 

Sử Dụng JOptionPane

```
import javax.swing.*; // import JAVA SWING graphical interface libraries

// This class demonstrates use of JOptionPane r
public class SimpleDialog
{
    public String inputDialog(String s)
    {
        return JOptionPane.showInputDialog(s); // static method call
    }
}
```

<code>static int</code>	<code>showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType)</code> Brings up a dialog where the number of choices is determined by the optionType parameter, where the messageType parameter determines the icon to display.
<code>static int</code>	<code>showConfirmDialog(Component parentComponent, Object message, String title, int optionType, int messageType, Icon icon)</code> Brings up a dialog with a specified icon, where the number of choices is determined by the optionType parameter.
<code>static String</code>	<code>showInputDialog(Component parentComponent, Object message)</code> Shows a question-message dialog requesting input from the user parented to parentComponent.
<code>static String</code>	<code>showInputDialog(Component parentComponent, Object message, Object initialValue)</code> Shows a question-message dialog requesting input from the user and parented to parentComponent.
<code>static String</code>	<code>showInputDialog(Component parentComponent, Object message, String title, int messageType)</code> Shows a dialog requesting input from the user parented to parentComponent with the dialog having the title title and message type messageType.
<code>static Object</code>	<code>showInputDialog(Component parentComponent, Object message, String title, int messageType, Icon icon, Object[] selectionValues, Object initialValue)</code> Prompts the user for input in a blocking dialog where the initial selection, possible selections, and all other options can be specified.
<code>static String</code>	<code>showInputDialog(Object message)</code> Shows a question-message dialog requesting input from the user.
<code>static String</code>	<code>showInputDialog(Object message, Object initialValue)</code> Shows a question-message dialog requesting input from the user, with the input value initialized to initialValue.

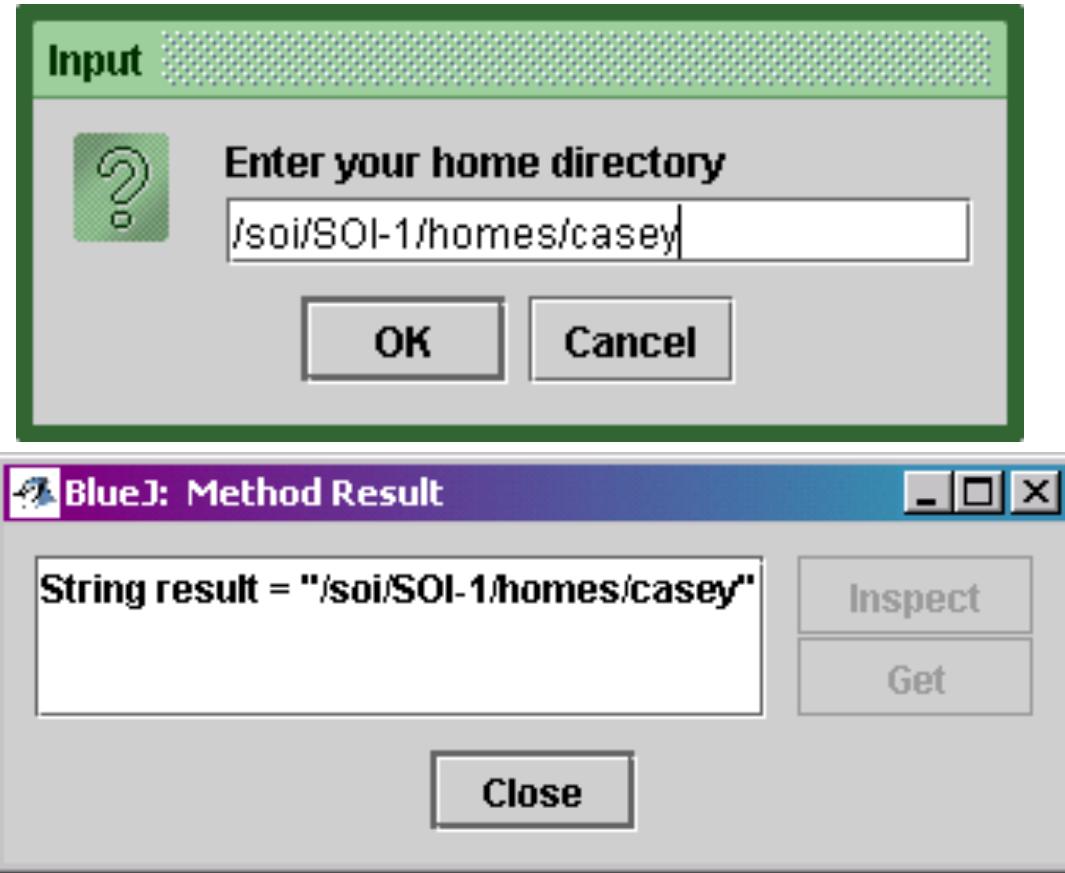
Using JOptionPane

```
import javax.swing.*; // import JAVA SWING graphical interface libraries

// This class demonstrates use of JOptionPane
public class SimpleDialog
{
    public String inputDialog(String s)
    {
        return JOptionPane.showInputDialog(s); // static method call
    }
}
```

JOptionPane

```
JOptionPane.showInputDialog("Enter your home directory");
```



Ví dụ

Ví dụ 1: code\GUI\MessageInJava

Ví dụ 2: code\GUI\MessageBox

JLabel

- ❖ *Label* dùng để hiển thị một chuỗi văn bản thông thường nhằm mô tả thêm thông tin cho các đối tượng khác.
- ❖ Các constructor của JLabel:

JLabel ()

JLabel (String text)

JLabel (String text, int hAlignment)

JLabel (Icon icon)

JLabel (Icon icon, int hAlignment)

JLabel (String text, Icon icon, int hAlignment)

Các thuộc tính JLabel

- text
- icon
- horizontalAlignment
- verticalAlignment

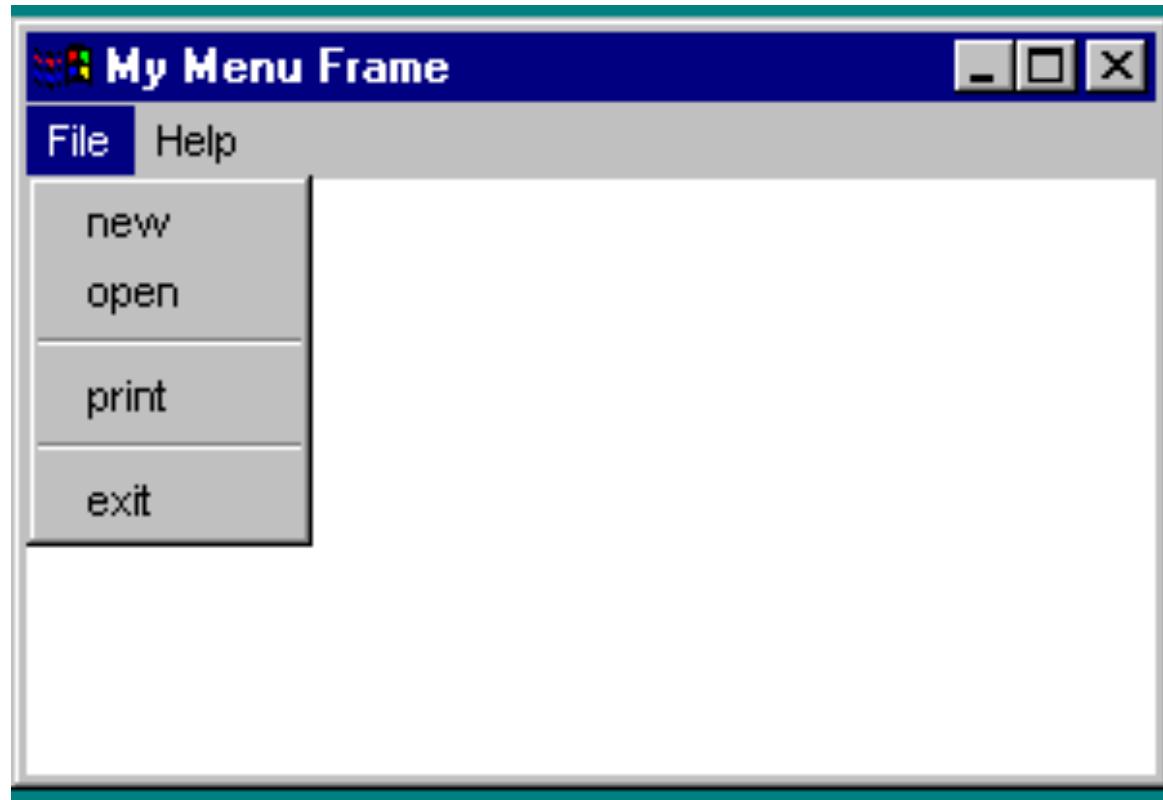
Ví Dụ



Menus

- Java cung cấp một số lớp - JMenuBar, JMenu, JMenuItem, JCheckBoxMenuItem, và JRadioButtonMenuItem – để thực thi menu trong một frame.
- Một JFrame hoặc JApplet có thể chứa một *menu bar* trên đó có gắn các *pull-down menu*. Các menu chứa các *menu item* để người dùng lựa chọn (hoặc bật/tắt). Menu bar có thể được xem như một cấu trúc để hỗ trợ các menu.

Menu Demo



Lớp JMenuBar

Menu bar chứa các menu; menu bar chỉ có thể được thêm vào 1 frame. Đoạn code sau tạo và thêm một JMenuBar vào 1 frame:

```
JFrame f = new JFrame();  
f.setSize(300, 200);  
f.setVisible(true);  
JMenuBar mb = new JMenuBar();  
f.setJMenuBar(mb);
```

Lớp Menu

Bạn gắn các menu vào một JMenuBar. Đoạn code sau tạo 2 menu File và Help, và thêm chúng vào JMenuBar mb:

```
JMenu fileMenu = new JMenu("File", false);  
JMenu helpMenu = new JMenu("Help", true);  
mb.add(fileMenu);  
mb.add(helpMenu);
```

Lớp JMenuItem

Đoạn code sau thêm các mục chọn (menu item) và các separator trong menu fileMenu:

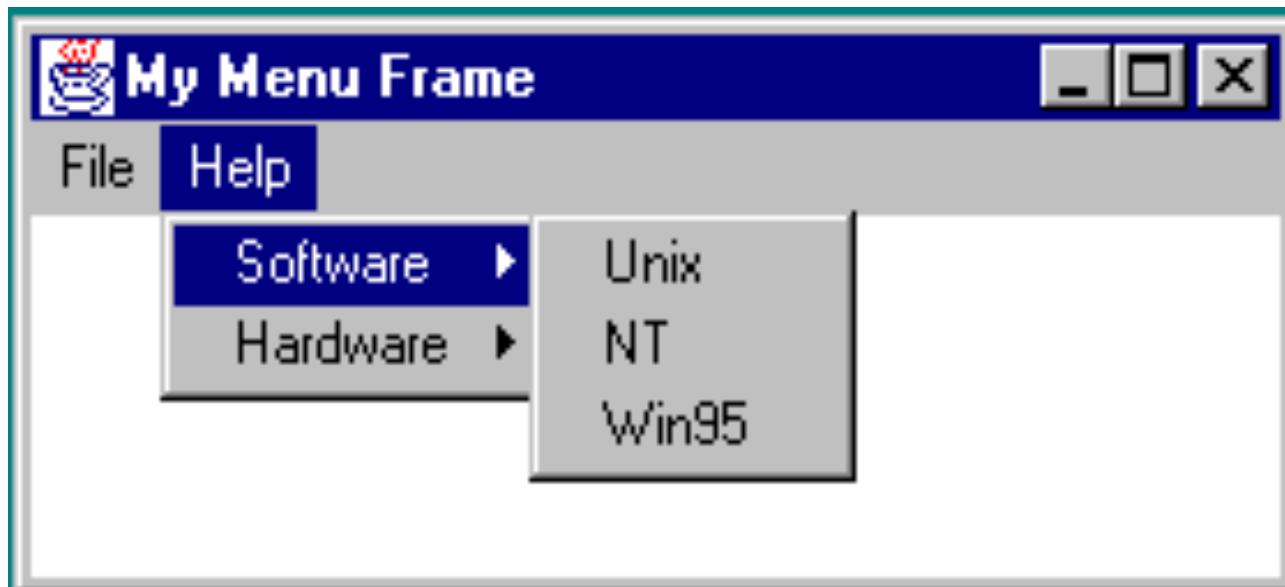
```
fileMenu.add(new JMenuItem("New")) ;  
fileMenu.add(new JMenuItem("Open")) ;  
fileMenu.addSeparator() ;  
fileMenu.add(new JMenuItem("Print")) ;  
fileMenu.addSeparator() ;  
fileMenu.add(new JMenuItem("Exit")) ;
```

Submenus

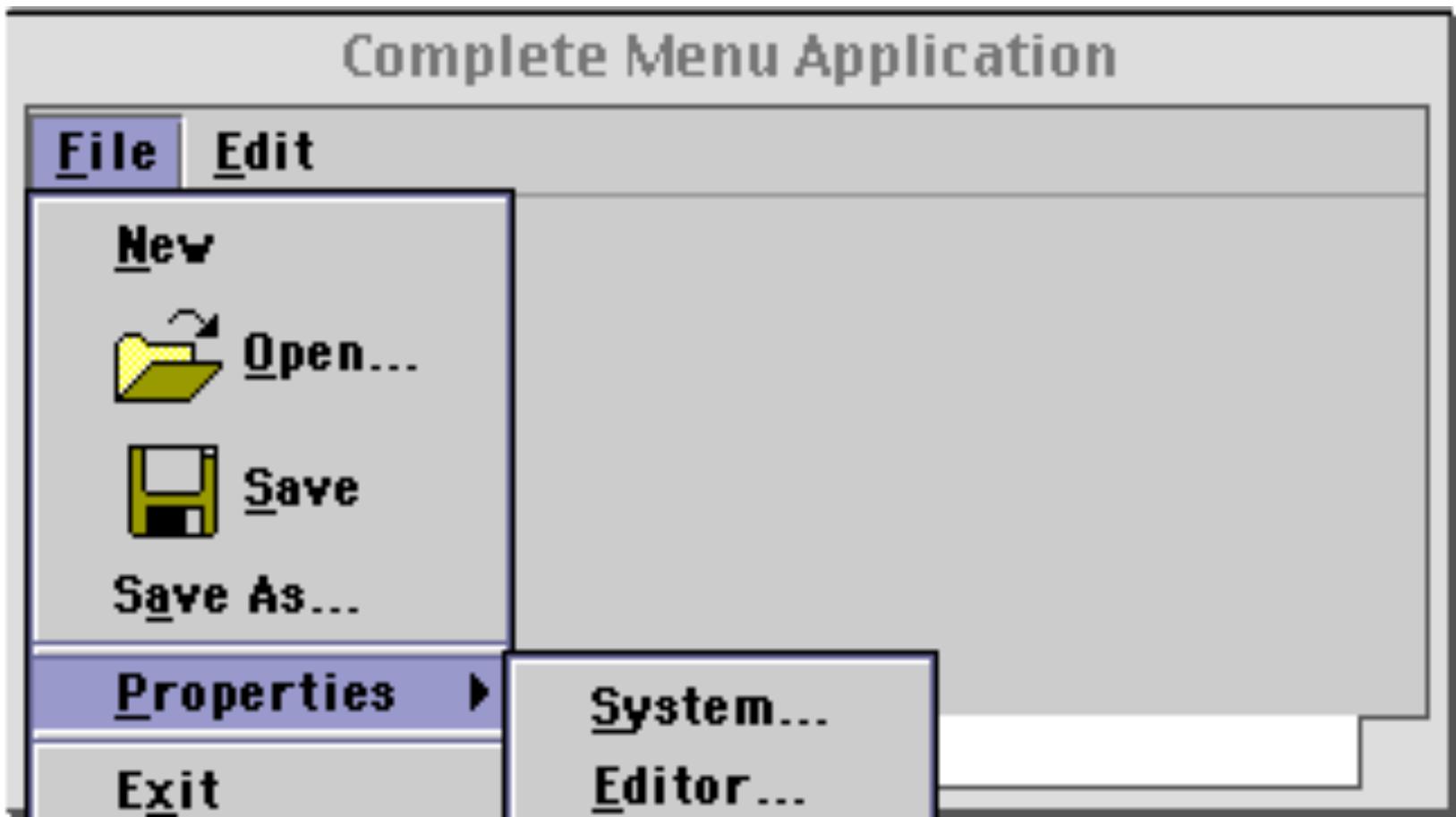
Bạn có thể thêm các submenus vào các menu item. Đoạn code sau thêm các submenu “Unix”, “NT”, và “Win95” vào trong mục chọn “Software”.

```
JMenu softwareHelpSubMenu = new JMenu("Software");  
JMenu hardwareHelpSubMenu = new JMenu("Hardware");  
helpMenu.add(softwareHelpSubMenu);  
helpMenu.add(hardwareHelpSubMenu);  
softwareHelpSubMenu.add(new JMenuItem("Unix"));  
softwareHelpSubMenu.add(new JMenuItem("NT"));  
softwareHelpSubMenu.add(new JMenuItem("Win95"));
```

Submenu Demo



Kết quả



Ví dụ

Ví dụ 1: Src: code/GUI/MyMenu

Ví dụ 2: Src: code/GUI/MyMenuAndJPanel

JTextField

- ❖ ***Text field*** là ô nhập dữ liệu dạng văn bản trên 1 dòng.
- ❖ Các constructor của JTextField:

JTextField()

JTextField(int columns)

Tạo một text field trống có số cột xác định.

JTextField(String text)

Tạo một text field với văn bản có sẵn.

JTextField(String text, int columns)

Tạo một text field với văn bản có sẵn và số cột xác định.

Các thuộc tính JTextField

- ❖ text
 - ❖ horizontalAlignment
 - ❖ editable
 - ❖ columns
-
- 

Các phương thức JTextField

- `getText ()`

Trả về chuỗi ký tự trong text field.

- `setText (String text)`

Đặt chuỗi ký tự trong text field.

- `setEditable (boolean editable)`

Cho phép hoặc vô hiệu hóa soạn thảo trong text field. Mặc định, `editable` là `true`.

- `setColumns (int)`

Thiết lập số cột trong text field. Chiều dài của text field có thể thay đổi.

Ví dụ

- Ví dụ 1: project name: GUI\MyTextField



JTextArea

- ❖ **TextArea** là khung cho phép người sử dụng nhập vào nhiều dòng văn bản.
- ❖ Các constructor của JTextArea:

JTextArea ()

JTextArea (String s)

JTextArea (int rows, int columns)

JTextArea (String s, int rows, int columns)

Các phương thức tính JTextArea

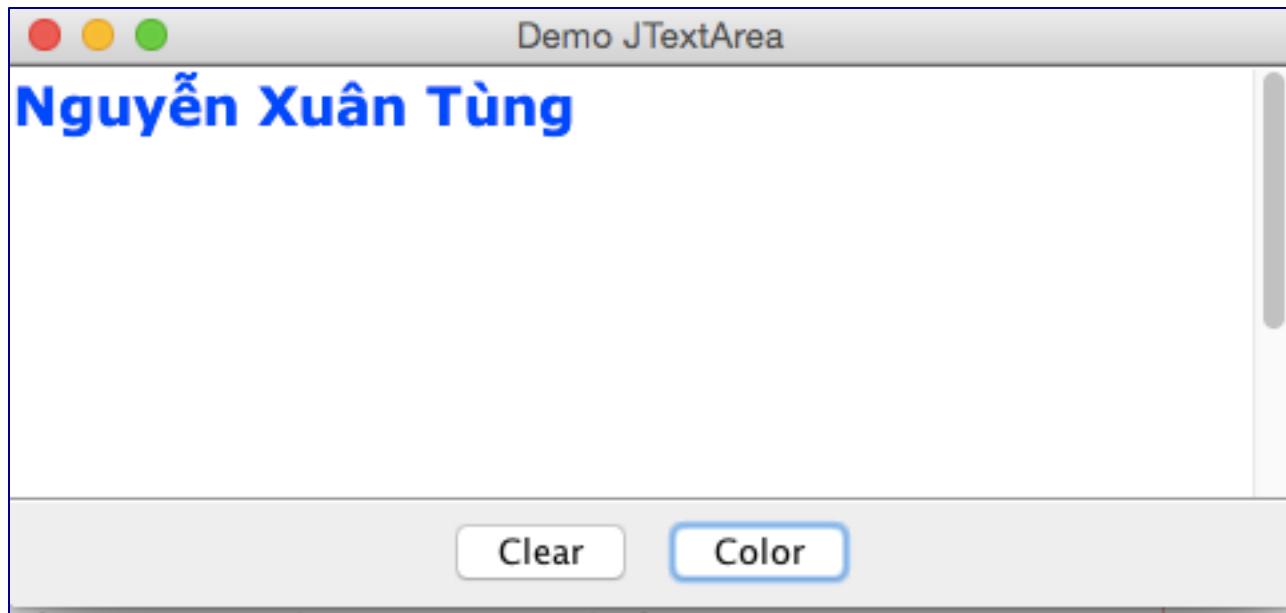
- `getText()` : Lấy nội dung của JTextArea
- `setText(String text)` : Đặt nội dung cho JTextArea
- `append(String str)` : Thêm chuỗi vào cuối nội dung của JTextArea
- `setColumns(int columns)` : Đặt số cột cho JTextArea
- `setRows(int rows)` : Đặt số dòng cho JTextArea
- `setLineWrap(boolean wrap)` : Đặt/Hủy line wrap

Các phương thức textArea

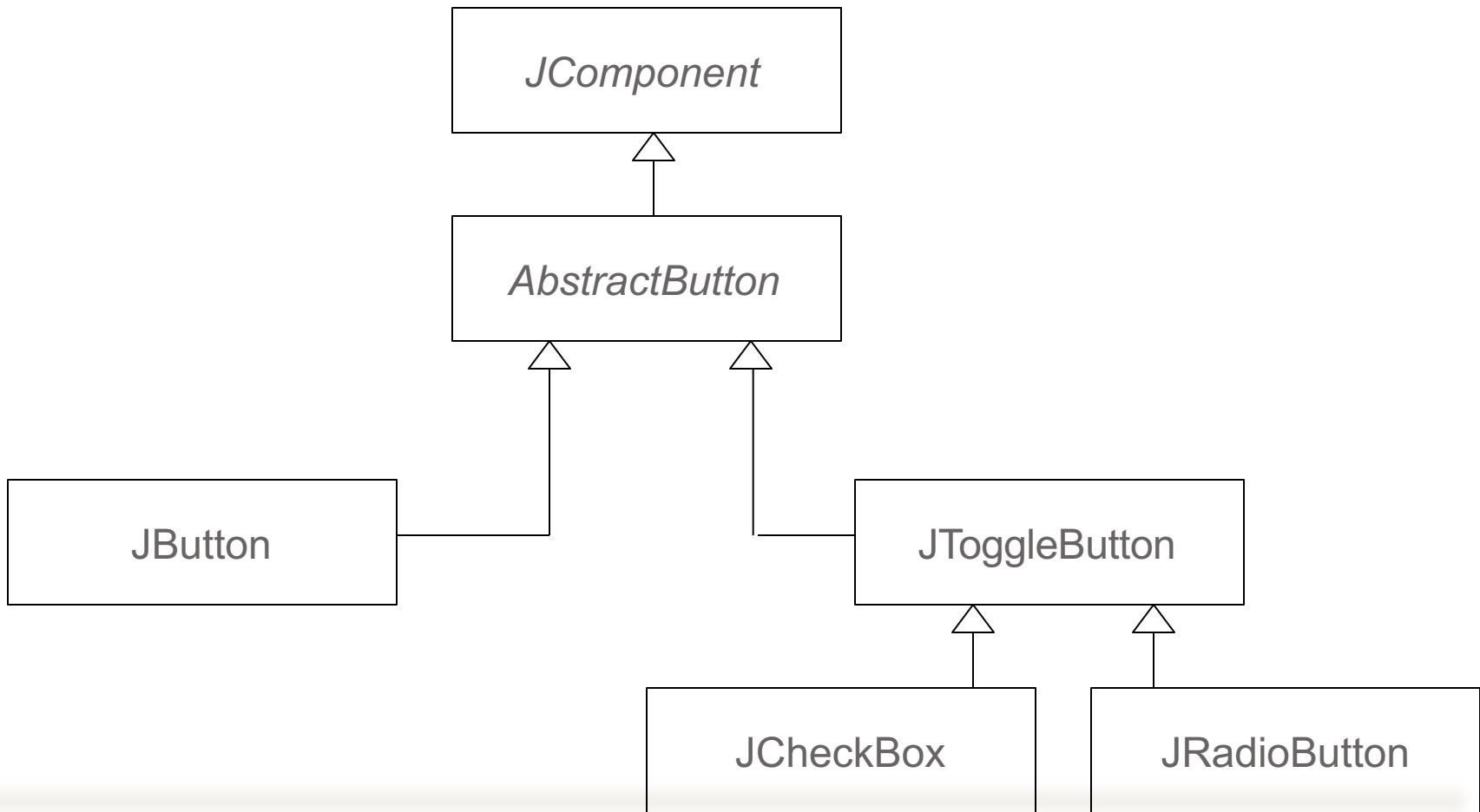
- `setWrapStyleWord(boolean word)`: Đặt/ Hủy word wrap
- `setDragEnabled(boolean b)` : Enable / Disable thao tác kéo nội dung
- `setCaretPosition(int position)` : Đặt vị trí con trỏ nháy
- `setCaretColor(Color c)` : Đặt màu cho con trỏ nháy
- **`public void insert(String s, int position)`**: Được sử dụng để chèn text vào vị trí đã cho.

Ví dụ

- Project name: GUIDemoJTextArea



Swing Button Classes



JCheckBox và JRadioButton

- Các nút lệnh thay đổi trạng thái
 - Nhận các giá trị on/off hoặc true/false
 - Swing hỗ trợ các kiểu:
 - **JCheckBox**
 - **JRadioButton**

Item Event

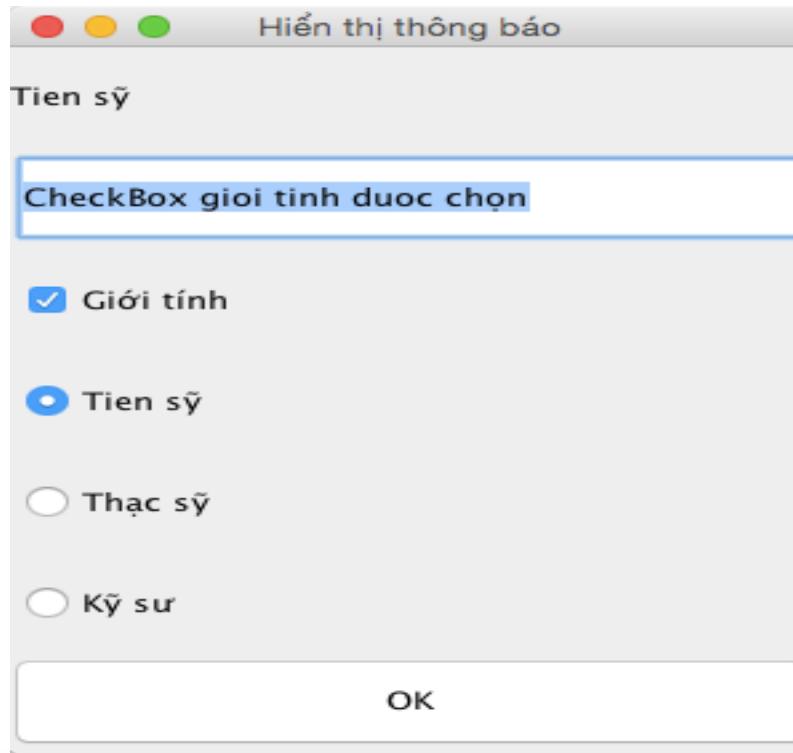
- Được tạo ra khi người dùng chọn các mục khác nhau trên JCheckBox, JRadioButton,..
- Các phương thức
 - Object `getItem()`: trả về mục được chọn
 - int `getStateChange()`: trả về trạng thái trạng thái của mục chọn (`DESELECTED/SELECTED`)

Item Listener

- `void itemStateChanged(ItemEvent e)`: được gọi thi hành khi người dùng chọn hoặc bỏ chọn 1 mục.

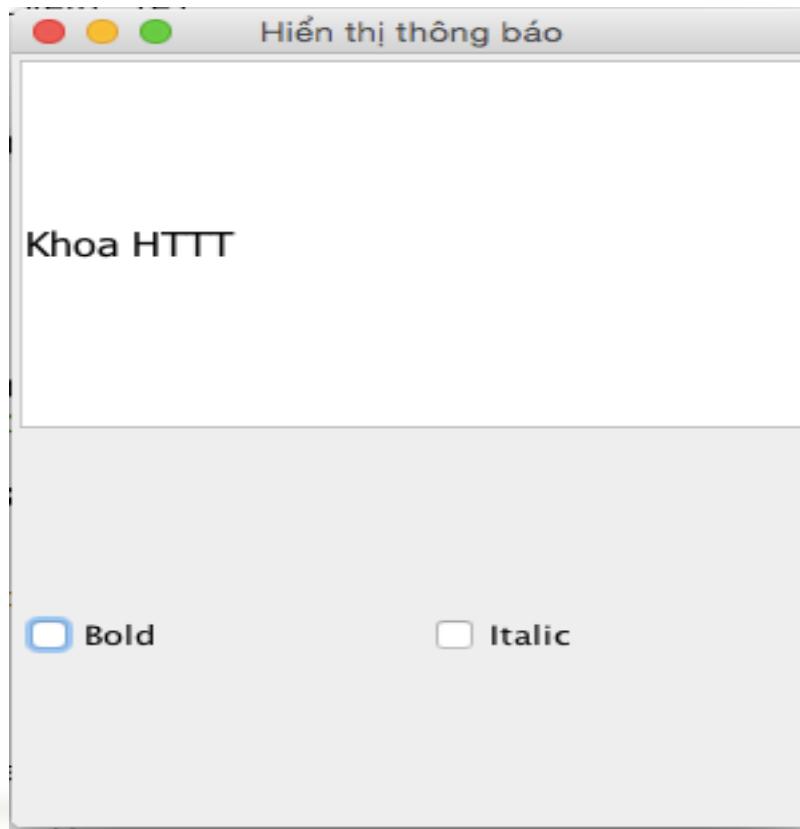
Ví dụ

- Ví dụ 1: project name: GUI\mycheckbox



Ví dụ

- Ví dụ 2: GUI\MyCheckBoxBoldAndItalicFont



JComboBox

- **JComboBox**

- Dùng để liệt kê danh sách các mục mà người dùng có thể chọn
- Còn được gọi là *drop-down list*
- *Phát sinh sự kiện ItemEvent khi người sử dụng chọn 1 mục trong danh sách*
- *JComboBox(Object[] items)*

- **Các phương thức**

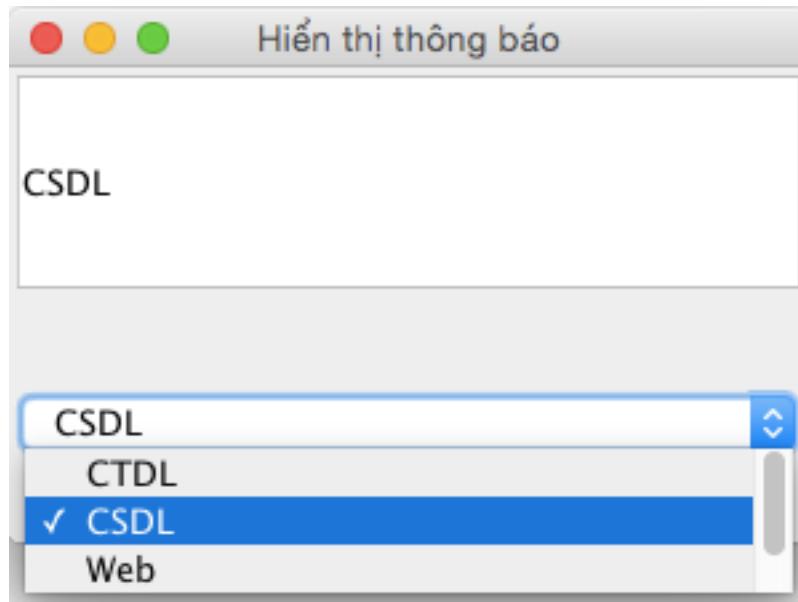
String **getSelectedItem()**: Lấy nội dung mục chọn

Int **getSelectedIndex()**: Lấy vị trí mục chọn

String **getItemAt(index)**: Lấy nội dung mục chọn tại vị trí index

Ví dụ

- Ví dụ 1: GUI\MyComboBox



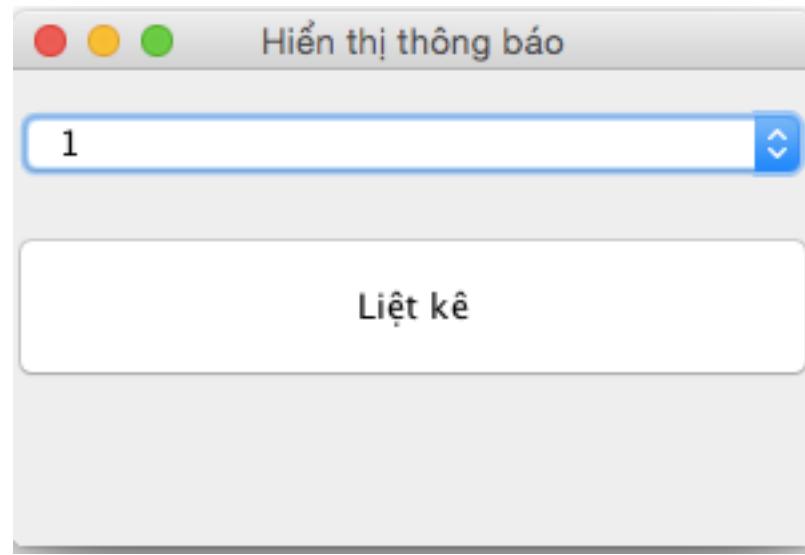
Ví dụ

- **Ví dụ 2:** Dữ liệu được thêm thông qua đối tượng DefaultComboBoxModel
- **Projectname:** GUI\myshowcombobox



Ví dụ

- **Ví dụ 3: GUI\myComboBox_CSDL 2**

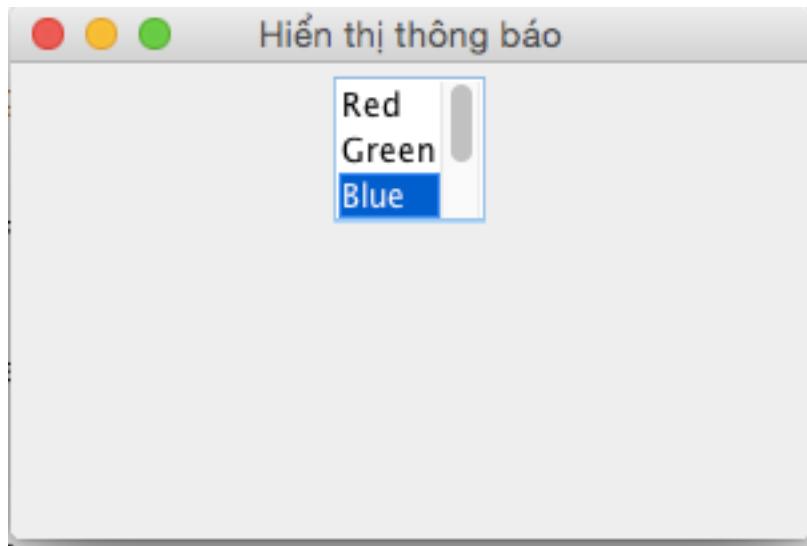


JList

- Jlist
 - Danh sách các mục chọn
 - Có thể chọn 1 hoặc nhiều mục
 - Phát sinh **ListSelectionEvent** khi người dùng chọn
- Các phương thức
 - **JList(Object[] listData)**
 - int **getSelectedIndex()**
 - Object[] **getSelectedValues()**
 - void **setListData(Object[] listData)**
 - void **setSelectedIndex(int idx)**
- **ListSelectionListener**
 - void **valueChanged(ListSelectionEvent e)**

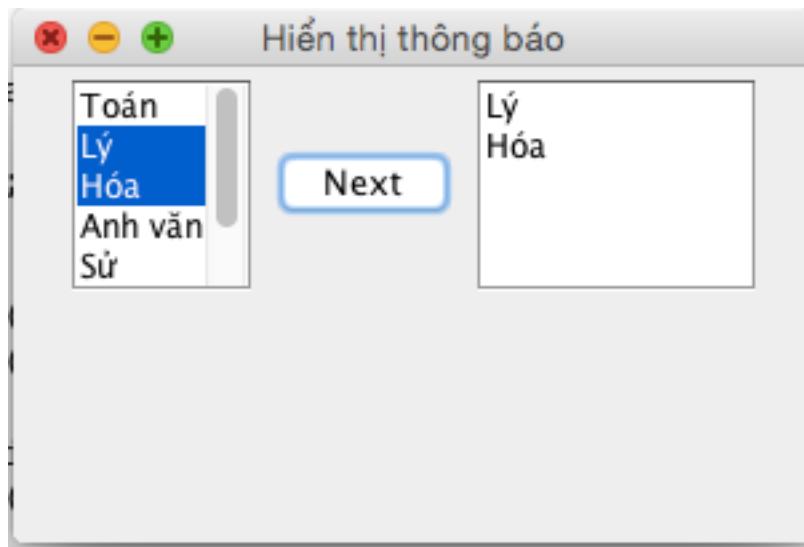
Ví dụ - Single List

- Ví dụ 1: GUI\MySingleListBox



Ví dụ - Multiple-Selection Lists

- Ví dụ 2: GUI\MyMultiList



JTabbedPane

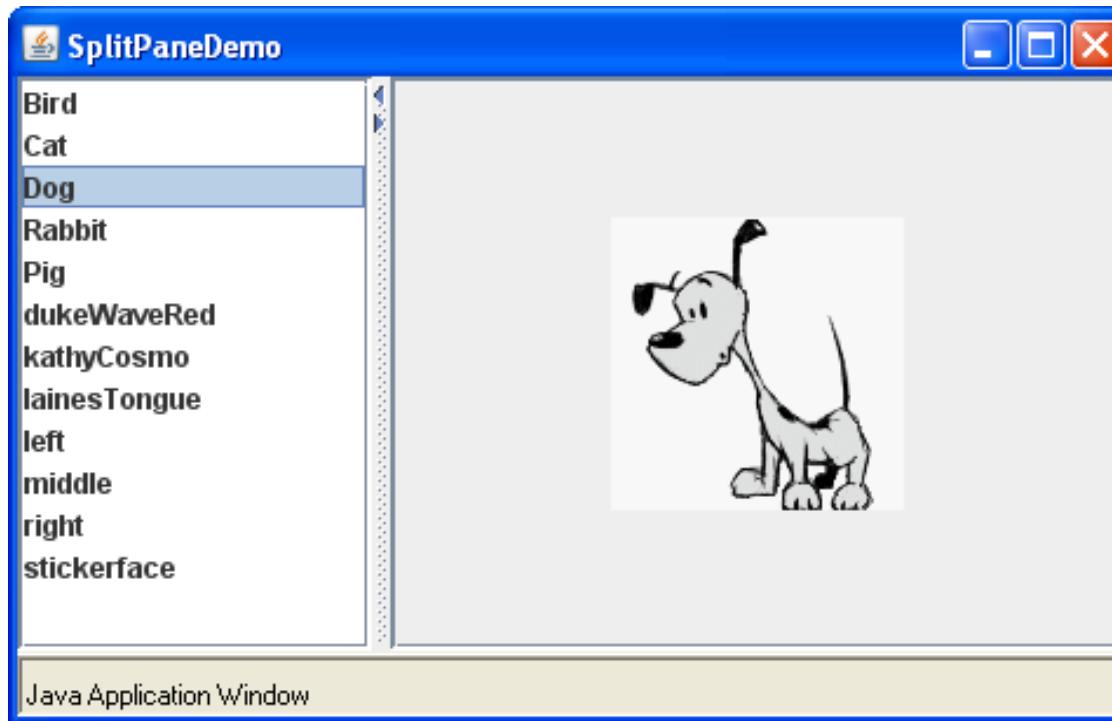
- JTabbedPane()
- addTab(..): Thêm 1 tab

Ví dụ - Tab

- Ví dụ: GUI\MyTabbedPane



JSplitPane



<http://java.sun.com/docs/books/tutorial/uiswing/examples/components/>

JSplitPane

```
JFrame fr = new JFrame("JSplitPane Demo");
fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JSplitPane splitPane;
JLabel picture = new JLabel();
String[] imageNames = { "Bird", "Cat", "Dog", "Rabbit", "Pig", "dukeWaveRed",
                      "kathyCosmo", "lainesTongue", "left", "middle", "right", "stickerface" };

//Create the list of images and put it in a scroll pane.
JList list = new JList(imageNames);
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
list.setSelectedIndex(0);
MyListSelectionListener listener = new MyListSelectionListener(picture, imageNames);
list.addListSelectionListener(listener);
JScrollPane listScrollPane = new JScrollPane(list);
```

JSplitPane, JScrollPane, JList

```
// Create a split pane with the two scroll panes in it.  
splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, listScrollPane, pictureScrollPane);  
splitPane.setOneTouchExpandable(true);  
splitPane.setDividerLocation(150);  
  
// Provide minimum sizes for the two components in the split pane.  
Dimension minimumSize = new Dimension(100, 50);  
listScrollPane.setMinimumSize(minimumSize);  
pictureScrollPane.setMinimumSize(minimumSize);  
  
//Provide a preferred size for the split pane.  
splitPane.setPreferredSize(new Dimension(400, 200));  
  
fr.getContentPane().add(splitPane);  
fr.pack();  
fr.setVisible(true);
```

JSplitPane, JScrollPane, JList

```
public class MyListSelectionListener implements ListSelectionListener {  
    String[] imageNames;  
    JLabel picture;  
  
    MyListSelectionListener(JLabel pic, String[] images) {  
        picture = pic;  
        imageNames = images;  
    }  
  
    public void valueChanged(ListSelectionEvent e) {  
        JList list = (JList)e.getSource();  
        ImageIcon icon = new ImageIcon(imageNames[list.getSelectedIndex()] + ".gif");  
        picture.setIcon(icon);  
    }  
}
```

Mouse Event Handling

- Event-listener của mouse events
 - **MouseListener**
 - **MouseMotionListener**

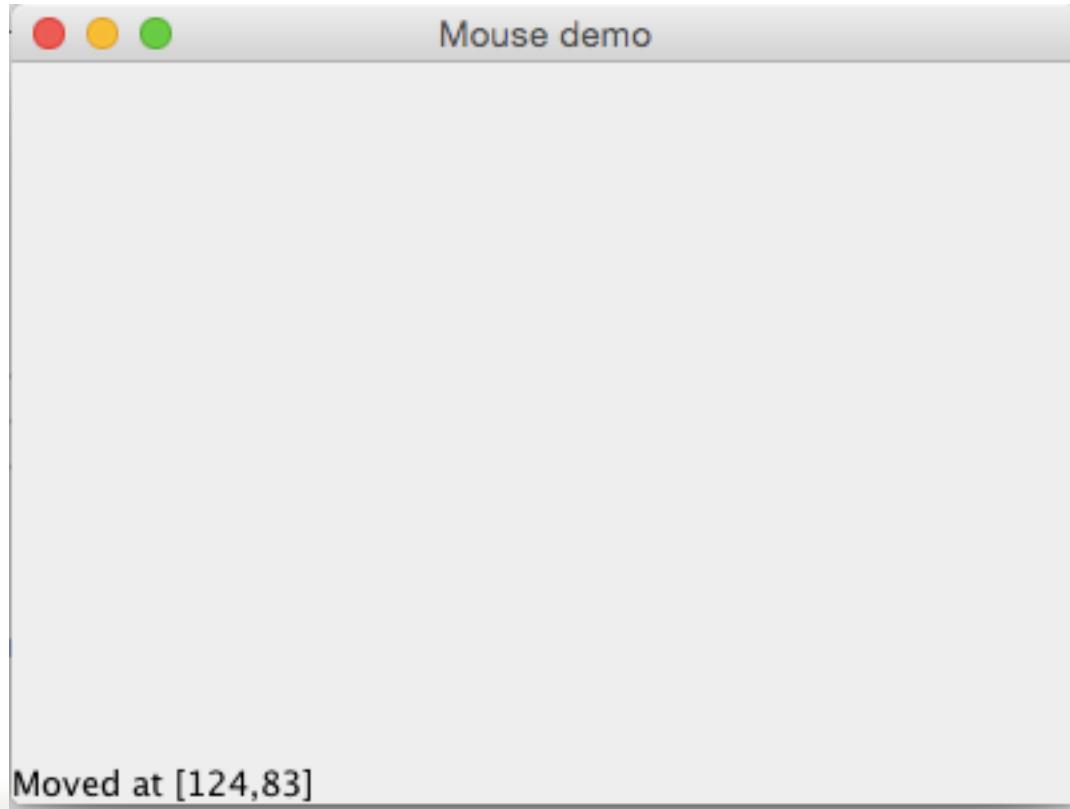
MouseListener and MouseMotionListener interface methods

MouseListener and MouseMotionListener interface methods	
<i>Methods of interface MouseListener</i>	
<code>public void mousePressed(MouseEvent event)</code>	Được gọi khi chuột nhấn trên một thành phần
<code>public void mouseClicked(MouseEvent event)</code>	Được gọi khi nhất, thả chuột
<code>public void mouseReleased(MouseEvent event)</code>	Được gọi khi chuột đã được nhấn trên 1 thành phần
<code>public void mouseEntered(MouseEvent event)</code>	Được gọi khi chuột nhập vào một thành phần (nhận focus nhập)
<code>public void mouseExited(MouseEvent event)</code>	Được gọi khi chuột thoát ra khỏi 1 thành phần
<i>Methods of interface MouseMotionListener</i>	
<code>public void mouseDragged(MouseEvent event)</code>	Khi nhấn và rê chuột
<code>public void mouseMoved(MouseEvent event)</code>	Khi di chuyển chuột

Fig. 12.16 `MouseListener` and `MouseMotionListener` interface methods.

Ví dụ

- Project Name: GUI\myMouse



Keyboard Event Handling

- Interface **KeyListener**

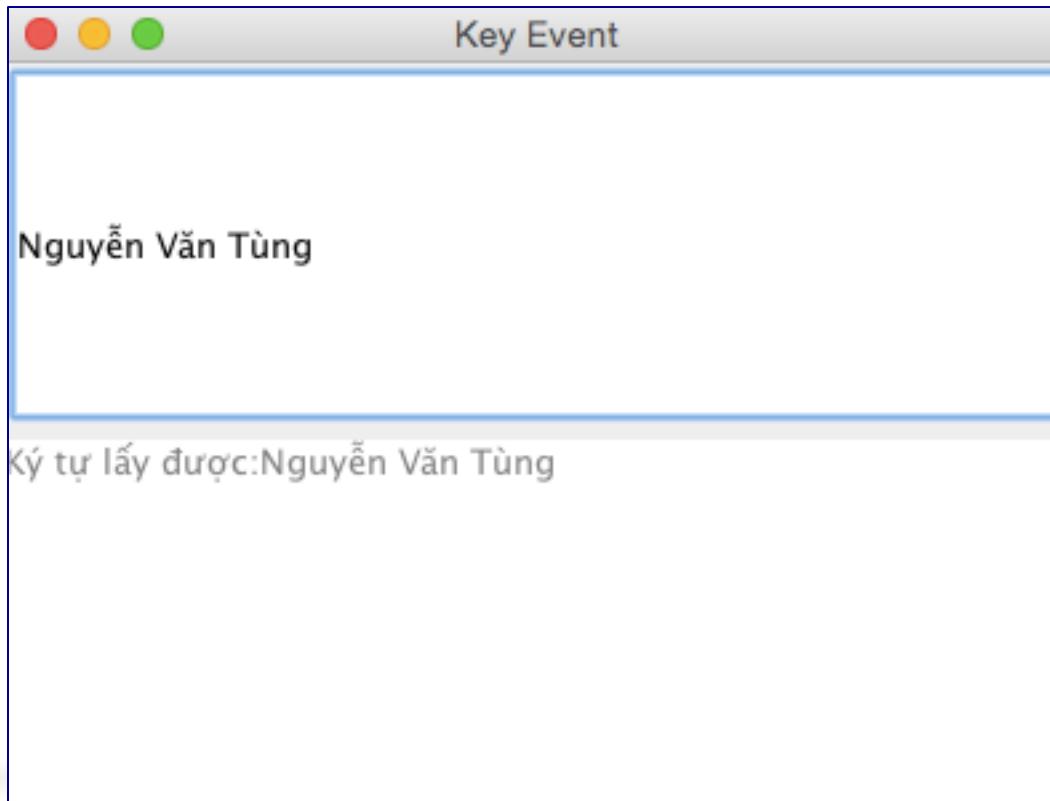
- Dùng để xử lý *key events*
 - Phát sinh khi 1 phím được nhấn và thả ra.
 - **KeyEvent**
 - Chứa *virtual key code* đại diện cho các phím

- Các phương thức

- public void keyPressed(KeyEvent e)
 - public void keyTyped(KeyEvent e)
 - public void keyReleased(KeyEvent e)
-

Ví dụ

- Project Name: GUI\myKeyEvent



Pluggable Look And Feel

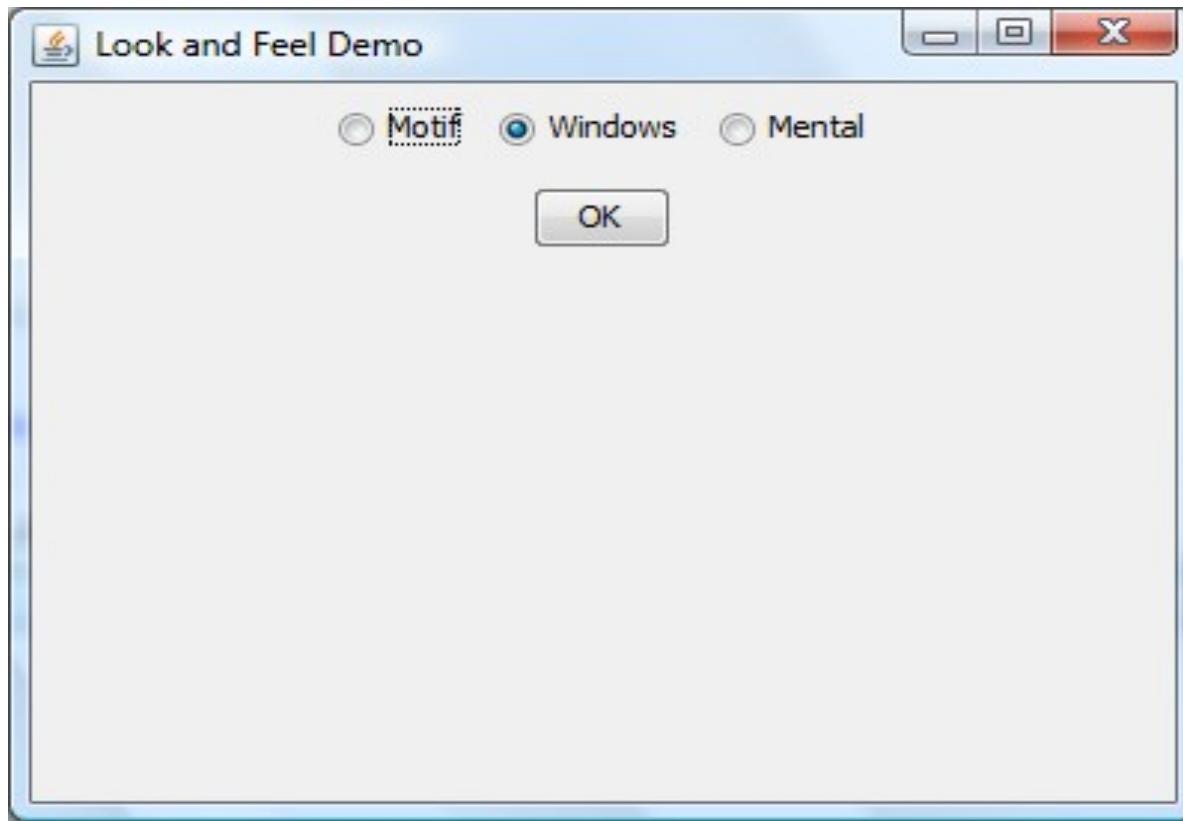
- Swing hỗ trợ pluggable look-and-feel.
 - Swing hỗ trợ 3 loại:
 - Motif -
"com.sun.java.swing.plaf.motif.MotifLookAndFeel"
 - Windows -
"com.sun.java.swing.plaf.windows.WindowsLookAndFeel"
 - Metal (Java platform) -
"javax.swing.plaf.metal.MetalLookAndFeel"
-

Thay Đổi Look and Feel

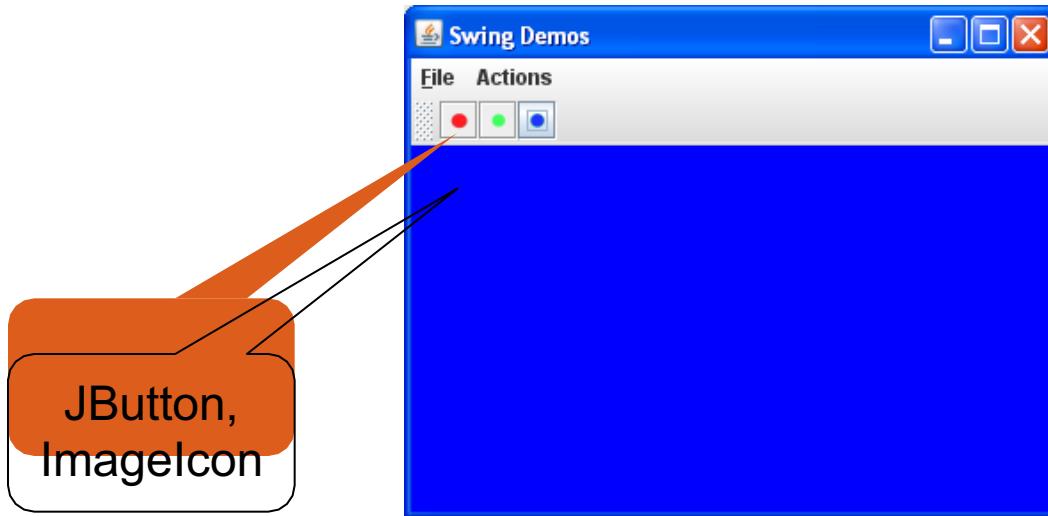
- UIManager.setLookAndFeel(String className) throws UnsupportedLookAndFeelException
- SwingUtilities.updateComponentTreeUI(Component c)

```
try {  
    UIManager.setLookAndFeel(  
        "com.sun.java.swing.plaf.Motif.MotifLookAndFeel" );  
    SwingUtilities.updateComponentTreeUI( myFrame );  
}  
catch( UnsupportedLookAndFeel e ) {  
}
```

Look And Feel Demo



JToolBar, Icon, ImageIcon



```
// Create toolbar
JToolBar toolbar = new JToolBar();
MainToolBarListener actionListener = new MainToolBarListener(this);
Icon red = new ImageIcon("images/red.png");
redIcon = new JButton(red);
redIcon.addActionListener(actionListener);
toolbar.add(redIcon);
```

JToolBar, Icon, ImageIcon

- Đặt tooltip cho Icon trên thanh toolbar

```
// Create toolbar
JToolBar toolbar = new JToolBar();
MainToolBarListener actionListener = new MainToolBarListener(this);
Icon red = new ImageIcon("images/red.png");
redIcon = new JButton(red);
redIcon.setToolTipText("Set red color for the background");
redIcon.addActionListener(actionListener);
toolbar.add(redIcon);
```

Graphics Context và Object

- Graphics context
 - Hỗ trợ thao tác vẽ trên màn hình
 - **Đối tượng Graphics** quản lý graphics context
 - Điều khiển cách vẽ
 - Cung cấp các phương thức để vẽ, chọn font, màu....
 - **Graphics** là 1 lớp trừu tượng!
- Class Component
 - Là lớp cơ sở của các thành phần trong **java.awt** và **javax.swing**
 - Phương thức **paint(Graphics g)**

Lớp Color

- Hỗ trợ các thao tác trên màu sắc.
- Color(int red, int green, int blue)
- Lớp Graphics:
 - void setColor(Color c): chọn màu dùng để vẽ
 - Color getColor(): lấy về màu đang chọn

Lớp Font

- `Font(String name, int style, int size)`
 - Name: tên font có trong hệ thống
 - Style: FONT.PLAIN, FONT.ITALIC, FONT.BOLD
 - Size: kích thước đơn vị point (1/72 inch)
- Lớp Graphics
 - `Font getFont()`
 - `void setFont(Font f)`

Lớp Graphics

- **drawString(s, x, y)**
 - Draw string at x, y
- **drawLine(x1, y1, x2, y2)**
 - Draw line from x1, y1 to x2, y2
- **drawRect(x1, y1, width, height)**
 - Draws rectangle with upper left corner x1, y1
- **fillRect(x1, y1, width, height)**
 - As above, except fills rectangle with current color
- **clearRect(x1, y1, width, height)**
 - As above, except fills rectangle with background color

Lớp Graphics

- **draw3DRect(x1, y1, width, height, isRaised)**
 - Draws 3D rectangle, raised if **isRaised is true**, else lowered.
- **fill3DRect**
 - As previous, but fills rectangle with current color
- **drawRoundRect(x, y, width, height, arcWidth, arcHeight)**
 - Draws rectangle with rounded corners. See diagram next slide.
- **fillRoundRect(x, y, width, height, arcWidth, arcHeight)**
- **drawOval(x, y, width, height)**
 - Draws oval in bounding rectangle (see diagram)
 - Touches rectangle at midpoint of each side
- **fillOval(x, y, width, height)**



Ví Dụ

```
public class LinesRectsOvals extends JFrame {  
    private String s = "Using drawString!";  
    public LinesRectsOvals() {  
        super( "Drawing lines, rectangles and ovals" );  
        setSize( 400, 165 );  
        setVisible(true); }  
    public void paint( Graphics g ) {  
        g.setColor( Color.red );  
        g.drawLine( 5, 30, 350, 30 );  
        g.setColor( Color.blue );  
        g.drawRect( 5, 40, 90, 55 );  
        g.fillRect( 100, 40, 90, 55 );  
        g.setColor( Color.cyan );  
        g.fillRoundRect( 195, 40, 90, 55, 50, 50 );  
        g.drawRoundRect( 290, 40, 90, 55, 20, 20 );
```

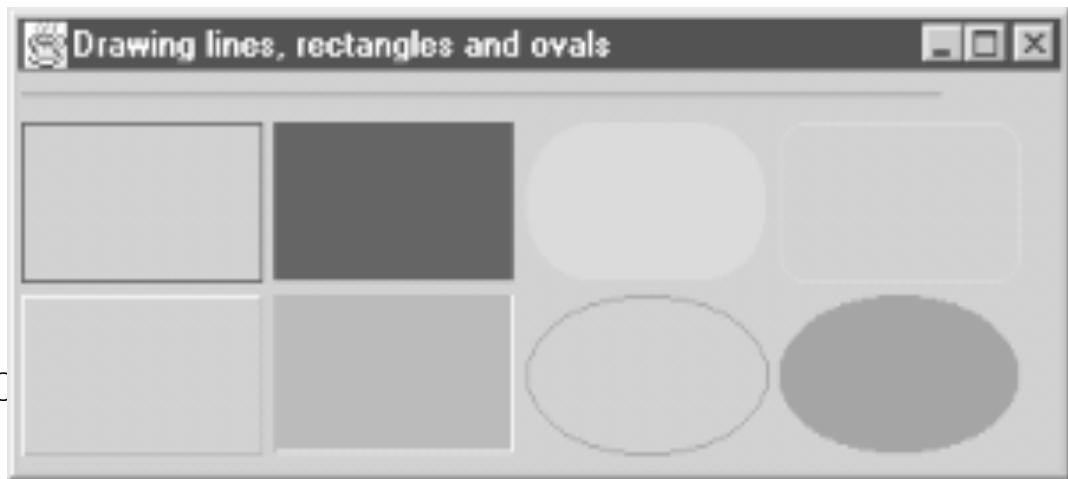


```
g.setColor( Color.yellow );
g.draw3DRect( 5, 100, 90, 55, true );
g.fill3DRect( 100, 100, 90, 55, false );

g.setColor( Color.magenta );
g.drawOval( 195, 100, 90, 55 );
g.fillOval( 290, 100, 90, 55 );

}

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

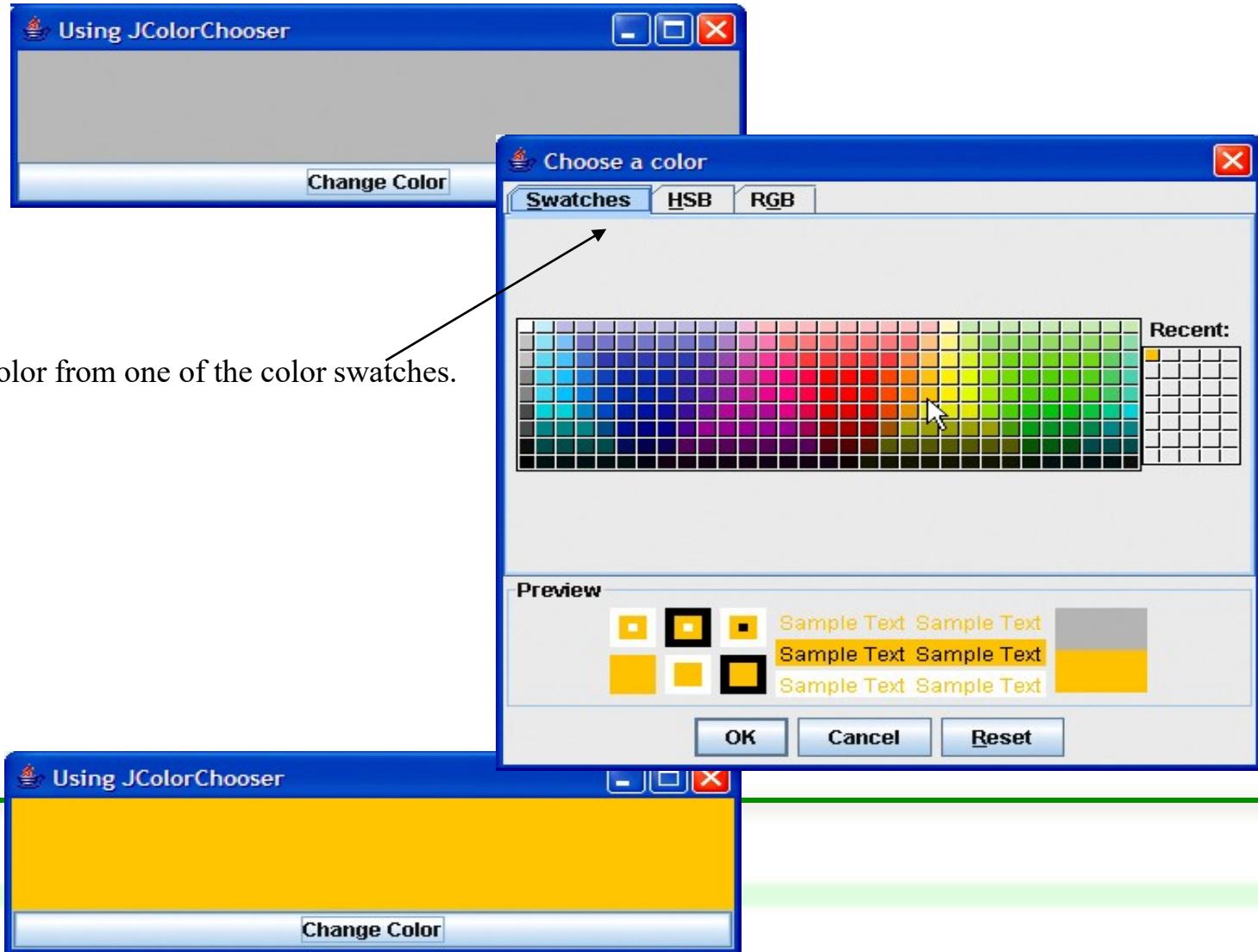


JColorChooser

- static Color showDialog (Component parent, String title, Color initColor)
 - Mở hộp thoại chọn màu
 - Trả về đối tượng màu đã chọn

JColorChooser Demo

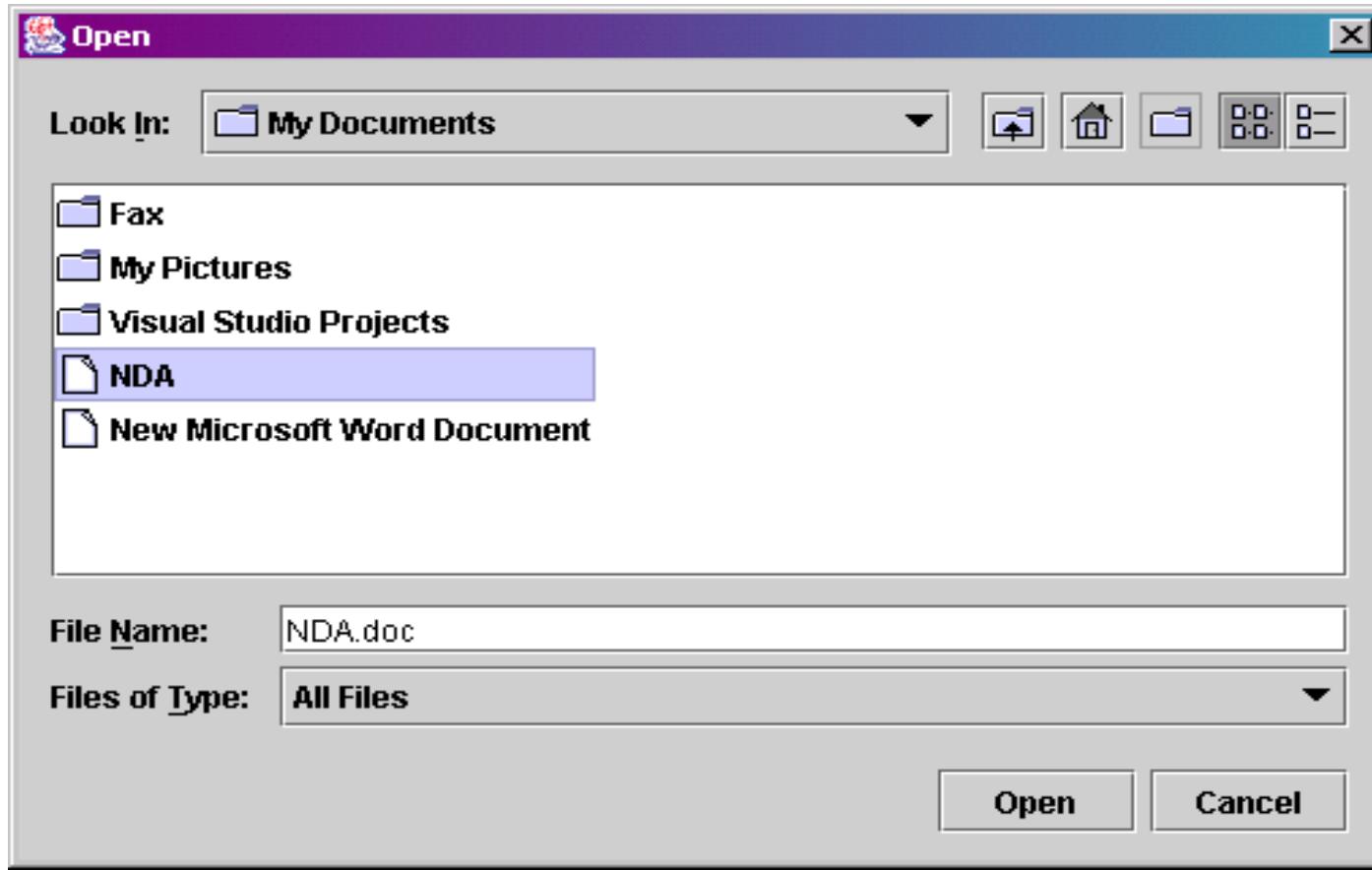
Select a color from one of the color swatches.



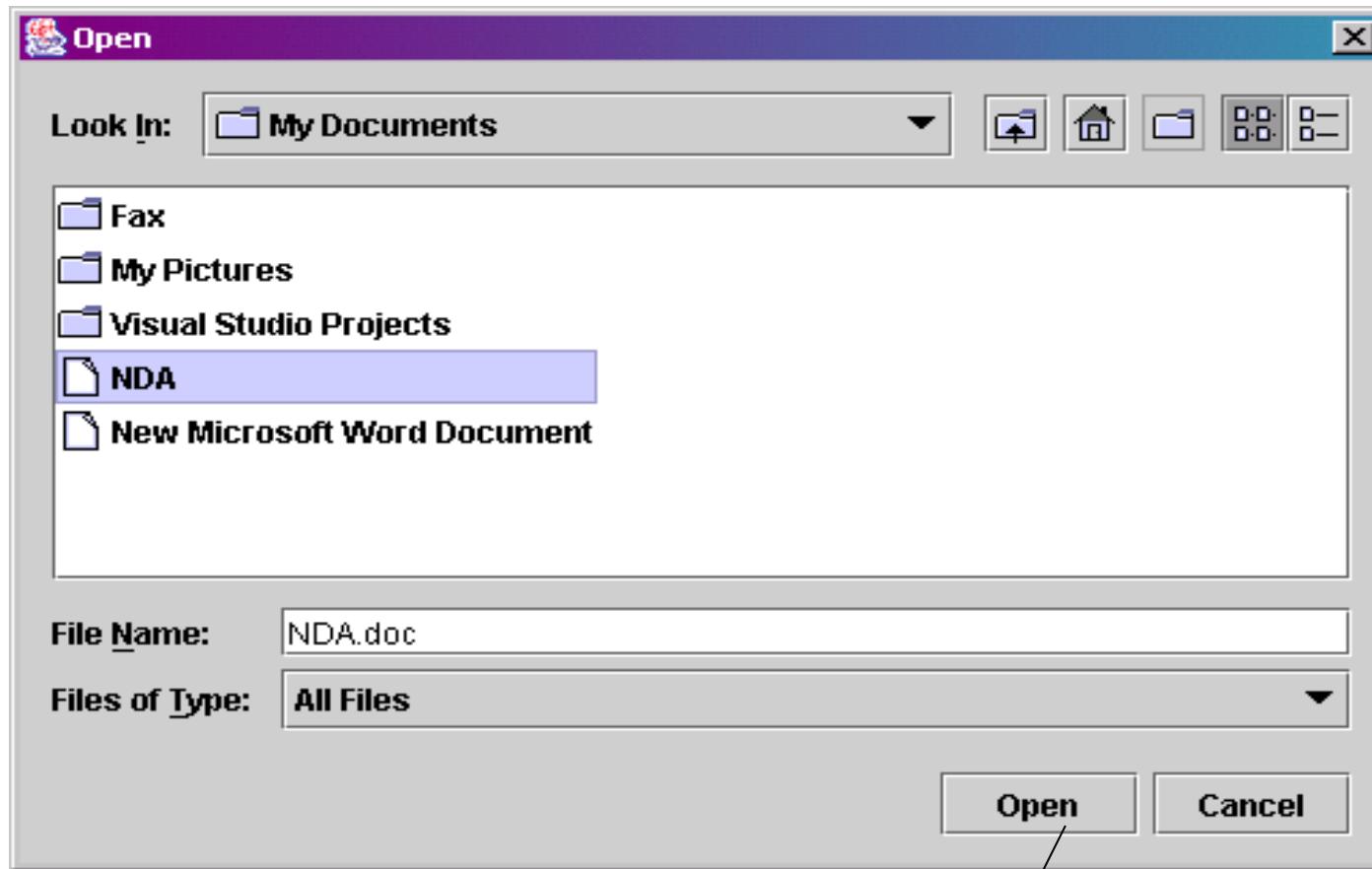
JFileChooser

- int showOpenDialog(Component parent): mở hộp thoại đọc tập tin
- int showSaveDialog(Component parent): mở hộp thoại lưu tập tin
- JFileChooser(): tạo đối tượng dùng để mở hộp thoại ghi/đọc tập tin
- File getSelectedFile(): lấy thông tin về tập tin/thư mục được chọn
 - String getPath()
 - String getName()

JFileChooser

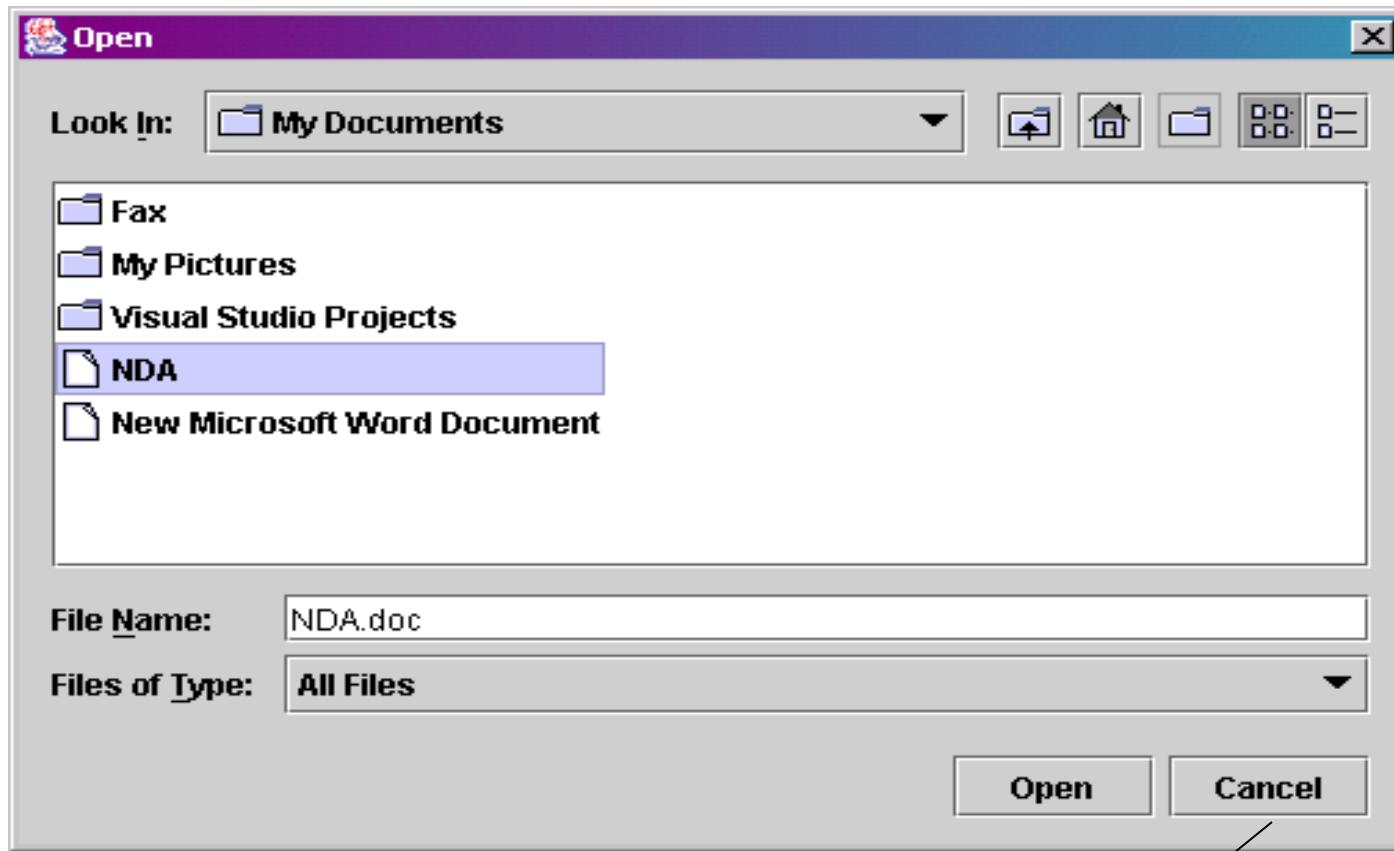


JFileChooser



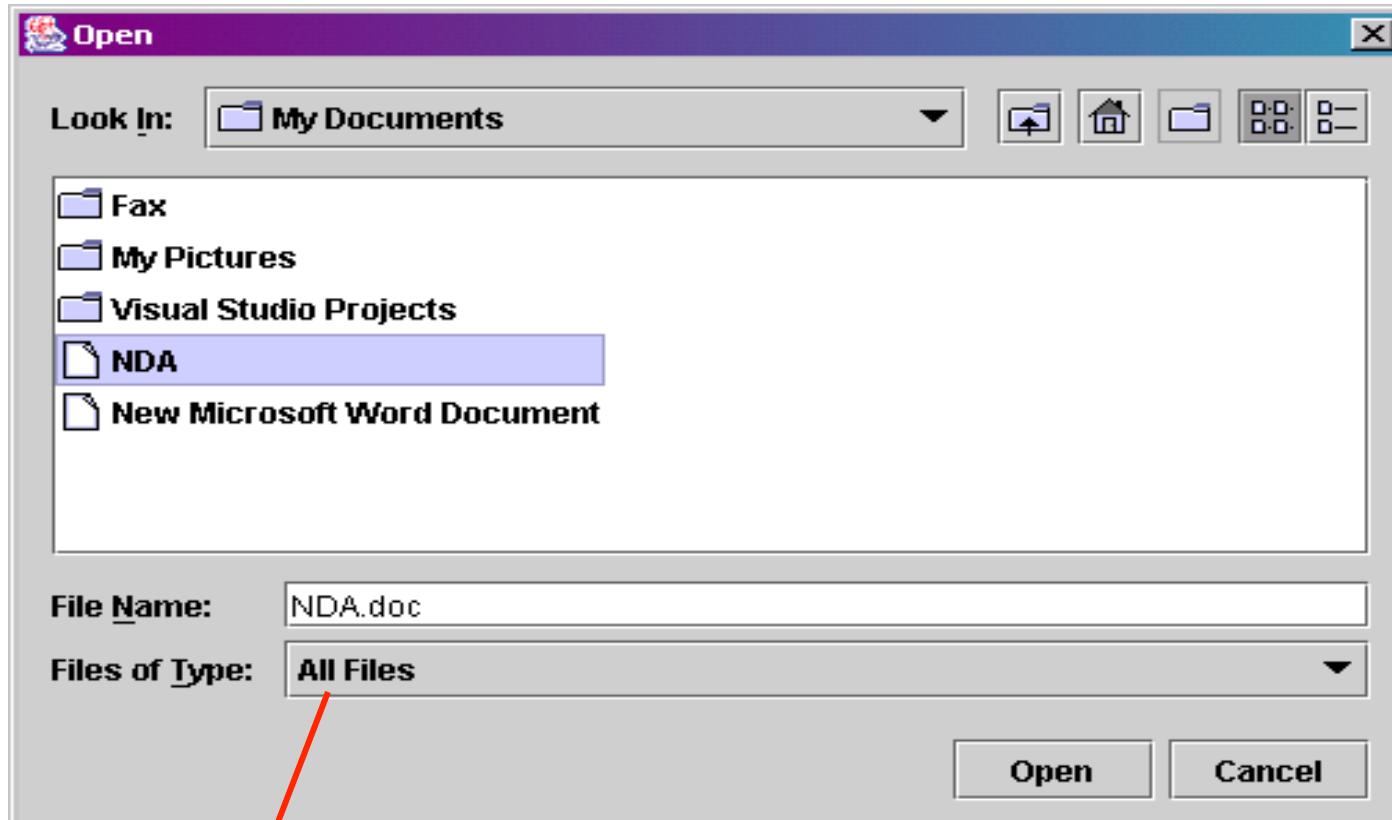
JFileChooser.APPROVE_OPTION

JFileChooser



JFileChooser.CANCEL_OPTION

JFileChooser



```
if (returnValue == JFileChooser.APPROVE_OPTION)
    filename = fc.getName();
```

showOpenDialog

```
public int showOpenDialog(Component parent)
    throws HeadlessException
```

Pops up an "Open File" file chooser dialog. Note that the text that appears in the approve button is determined by the L&F.

Parameters:

parent - the parent component of the dialog, can be null; see `showDialog` for details

Returns:

the return state of the file chooser on popdown:

- JFileChooser.CANCEL_OPTION
- JFileChooser.APPROVE_OPTION
- JFileChooser.ERROR_OPTION if an error occurs or the dialog is dismissed

Throws:

[HeadlessException](#) - if `GraphicsEnvironment.isHeadless()` returns true.

See Also:

[GraphicsEnvironment.isHeadless\(\)](#), [showDialog\(Component, String\)](#)

showSaveDialog

```
public int showSaveDialog(Component parent)
    throws HeadlessException
```

Pops up a "Save File" file chooser dialog. Note that the text that appears in the approve button is determined by the L&F.

showOpenDialog

```
public int showOpenDialog(Component parent)
    throws HeadlessException
```

Pops up an "Open File" file chooser dialog. Note that the text that appears in the approve button is determined by the L&F.

Parameters:

parent - the parent component of the dialog, can be null; see `showDialog` for details

Returns:

the return state of the file chooser on popdown:

- JFileChooser.CANCEL_OPTION
- JFileChooser.APPROVE_OPTION
- JFileChooser.ERROR_OPTION if an error occurs or the dialog is dismissed

Throws:

[HeadlessException](#) - if `GraphicsEnvironment.isHeadless()` returns true.

See Also:

[GraphicsEnvironment.isHeadless\(\)](#), [showDialog\(Component, String\)](#)

showSaveDialog

```
public int showSaveDialog(Component parent)
    throws HeadlessException
```

Pops up a "Save File" file chooser dialog. Note that the text that appears in the approve button is determined by the L&F.

<code>String</code>	<code>getName(File f)</code> Returns the filename.
<code>File</code>	<code>getSelectedFile()</code> Returns the selected file.
<code>File[]</code>	<code>getSelectedFiles()</code> Returns a list of selected files if the file chooser is set to allow multiple selection.
<code>String</code>	<code>getTypeDescription(File f)</code> Returns the file type.
<code>FileChooserUI</code>	<code>getUI()</code> Gets the UI object which implements the L&F for this component.
<code>String</code>	<code>getUIClassID()</code> Returns a string that specifies the name of the L&F class that renders this component.
<code>boolean</code>	<code>isAcceptAllFileFilterUsed()</code> Returns whether the AcceptAll FileFilter is used.
<code>boolean</code>	<code>isDirectorySelectionEnabled()</code> Convenience call that determines if directories are selectable based on the current file selection mode.
<code>boolean</code>	<code>isFileHidingEnabled()</code> Returns true if hidden files are not shown in the file chooser; otherwise, returns false.
<code>boolean</code>	<code>isFileSelectionEnabled()</code> Convenience call that determines if files are selectable based on the current file selection mode.
<code>boolean</code>	<code>isMultiSelectionEnabled()</code> Returns true if multiple files can be selected.

Using JFileChooser

```
import java.io.*;          // import JAVA file and stream handling classes
import javax.swing.*;      // import JAVA SWING graphical interface libraries

// This class demonstrates use of JFileChooser
public class SimpleDialog
{
    String fileName;
    File theFile;
    public int fileChooserDialog()
    {
        // Make an instance, call the showOpenDialog method and return result
        JFileChooser fc = new JFileChooser();
        int retval = fc.showOpenDialog(null);
        if(retval==JFileChooser.APPROVE_OPTION){
            theFile=fc.getSelectedFile();
            fileName=fc.getName(theFile);
        }
        return retval;
    }
}
```

Using JFileChooser

```
import java.io.*;          // import JAVA file and stream handling classes
import javax.swing.*;      // import JAVA SWING graphical interface libraries

// This class demonstrates use of JFileChooser
public class SimpleDialog
{
    String fileName;
    File theFile;
    public int fileChooserDialog()
    {
        // Make an instance, call the showOpenDialog method and return result
        JFileChooser fc = new JFileChooser();
        int retval = fc.showOpenDialog(null);
        if(retval==JFileChooser.APPROVE_OPTION){
            theFile=fc.getSelectedFile();
            fileName=fc.getName(theFile);
        }
        return retval;
    }
}
```

Using JFileChooser

```
import java.io.*;          // import JAVA file and stream handling classes
import javax.swing.*;      // import JAVA SWING graphical interface libraries

// This class demonstrates use of JFileChooser
public class SimpleDialog
{
    String fileName;
    File theFile;
    public int fileChooserDialog()
    {
        // Make an instance, call the showOpenDialog method and return result
        JFileChooser fc = new JFileChooser();
        int retval = fc.showOpenDialog(null);
        if(retval==JFileChooser.APPROVE_OPTION){
            theFile=fc.getSelectedFile();
            fileName=fc.getName(theFile);
        }
        return retval;
    }
}
```

a File object
a String

Lọc Tập Tin Hiển Thị

- `FileNameExtensionFilter(String dispC, String filter)`
 - `FileNameExtensionFilter filter = new FileNameExtensionFilter("JPG & GIF Images", "jpg", "gif");`
- `JFileChooser :`
 - `setFileFilter(FileNameExtensionFilter filter)`



Mã sản phẩm	<input type="text"/>		
Tên sản phẩm	<input type="text"/>		
Giá	<input type="text"/>		
STT	Mã Số	Tên sp	giá
1	101	iphoe5	670000
2	102	iphone5s	780000
3	103	iphone6s	700000

Duyệt Bảng

Thêm

Delete