

LẬP TRÌNH JAVA



Nội dung chương 5

5.1 Giới thiệu chung về AWT và Swing

5.2 Cấu trúc chung của các giao diện GUI

5.3 Giao diện với các đối tượng cơ bản

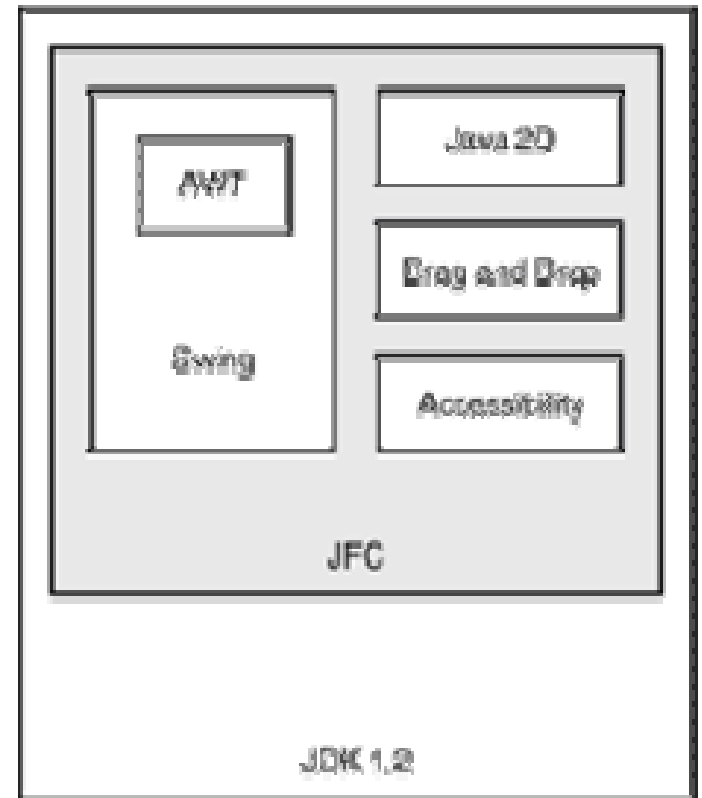
5.4 Giao diện với các đối tượng nâng cao

5.5 Các kỹ thuật tạo Tables



5.1 Giới thiệu chung về AWT và Swing

- JFC- Java Foundation Class
- JFC là thư viện các lớp được tạo ra nhằm đơn giản hoá quá trình thiết kế và giảm thời gian lập trình.
- JFC mở rộng AWT bằng cách thêm vào các lớp và các GUI component.





GIỚI THIỆU VỀ THIẾT KẾ GUI

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



AWT

- AWT: viết tắt của **Abstract Windows Toolkit**.
- AWT được thiết kế phục vụ cho việc tạo ra giao diện người dùng và xử lý đồ họa.
- Là thư viện các lớp hỗ trợ mọi thứ cần thiết cho developer tạo giao diện đồ họa cho các ứng dụng java.
 - `import java.awt.*;`
 - `import java.awt.event.*;`

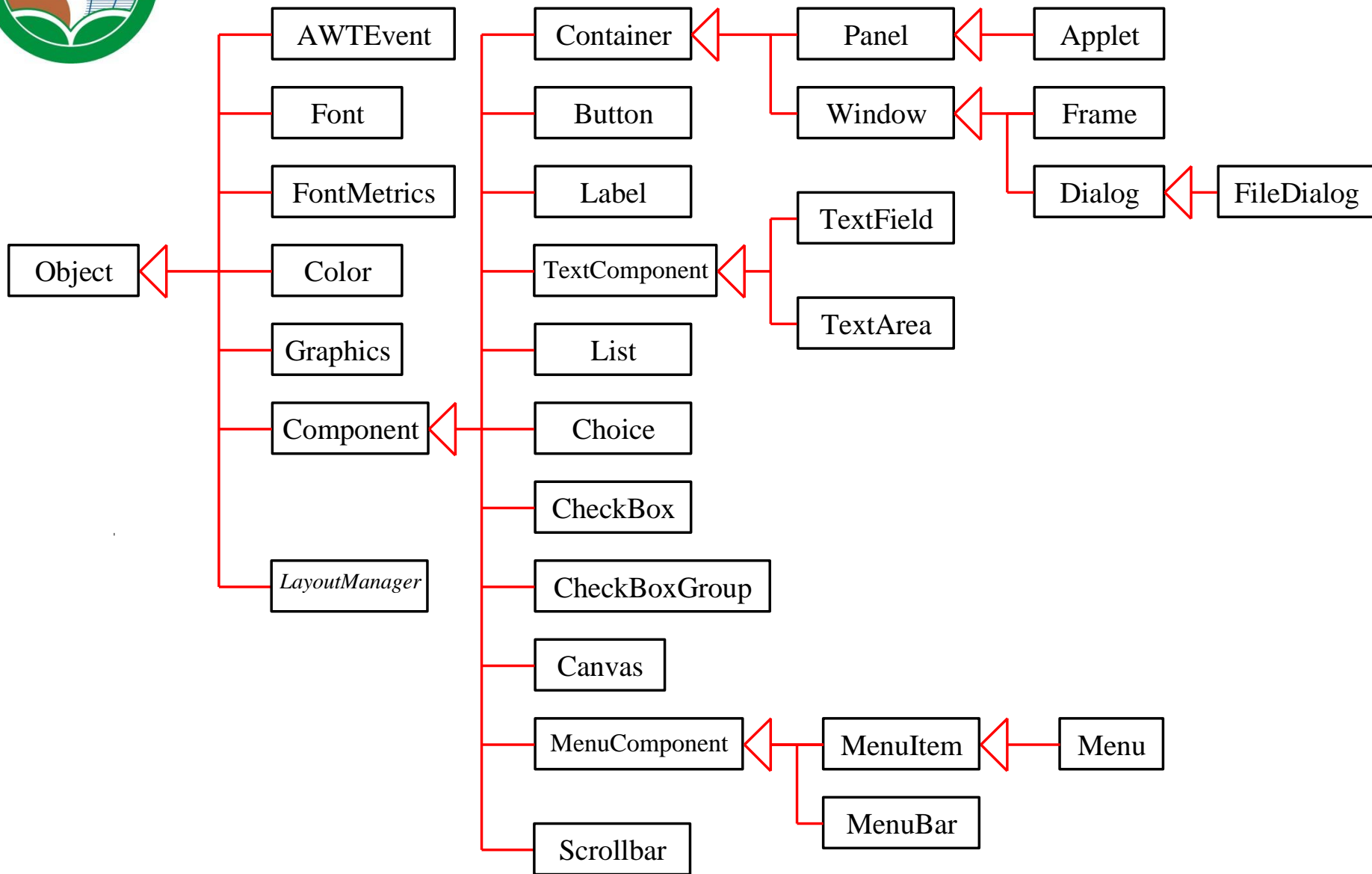


AWT

- Hầu như tất cả các component trong awt đều kế thừa từ lớp `java.awt.Component`.
- Một số component trong awt:
 - Label
 - Button
 - CheckBox
 - Container
 - List
 - Scrollbar
 - ...



GIỚI THIỆU AWT





Swing

- Swing là một tập hợp các thành phần giao diện bao gồm các thành phần đơn giản nhất như các label, button tới các thành phần phức tạp như table, tree.
- Hầu hết các thành phần giao diện trong swing đều kế thừa từ JComponent.
- Bản thân JComponent lại kế thừa từ java.awt.Container.



Swing

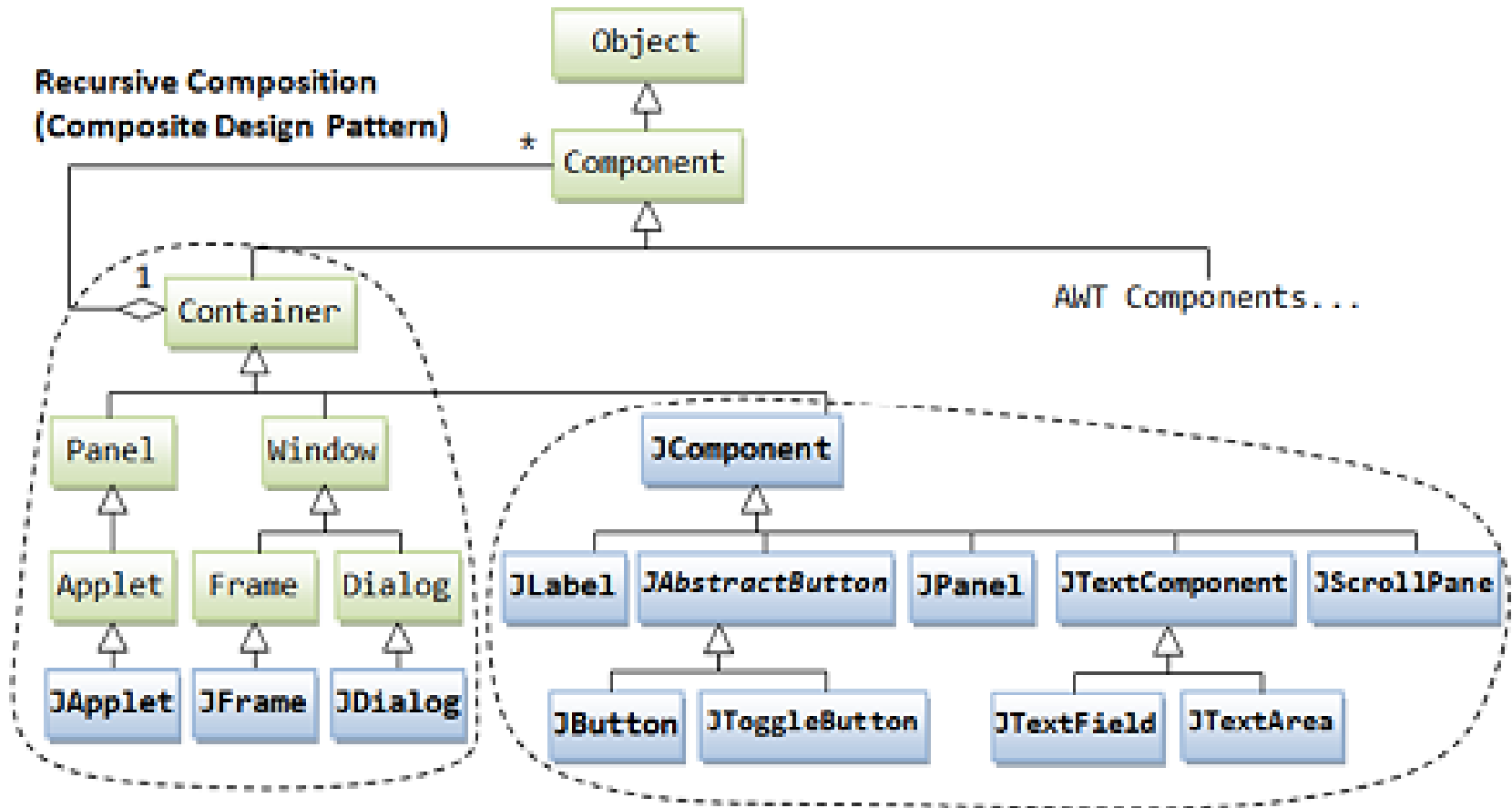
- Đa số các component có trong awt đều có trong swing, và tên của các component tương ứng sẽ có chữ J đứng trước.
- Một số component của swing:
 - JComponent
 - JLabel
 - JButton
 - JList
 - JMenuBar
 - JOptionPane
 - JPanel
 - JFrame
 - JApplet
 - ...



Swing

- Swing cung cấp 18 gói có thể sử dụng xây dựng giao diện đồ hoạ.
- Các thành phần tạo giao diện nằm trong gói `javax.swing`
- Tên của các lớp này bắt đầu bằng chữ J
- Ưu điểm:
 - Cung cấp thêm các đối tượng mới để xây dựng giao diện đồ hoạ look and feel.
 - Hỗ trợ các thao tác sử dụng bàn phím thay chuột
 - Sử dụng tài nguyên hiệu quả hơn.

Các phần tử của Swing





Các thư viện của Swing

- **javax.swing** : các component thông thường của swing.
- **javax.swing.border** : chứa các lớp và interface hỗ trợ làm việc với các kiểu border.
- **javax.swing.event** : chứa các lớp và interface hỗ trợ làm việc với sự kiện và listener.
- **javax.swing.filechooser** : chứa các lớp và interface hỗ trợ làm việc với JFileChooser (lựa chọn file).



Các thư viện của Swing

- **javax.swing.plaf** : chứa các look & feel API dùng để định nghĩa custom look & feel.
- **javax.swing.plaf.basic** : chứa các lớp và interface hỗ trợ làm việc với các look & feel cơ bản.
- **javax.swing.plaf.metal** : chứa các lớp và interface hỗ trợ làm việc với metal look & feel. Đây là look & feel default của java.
- **javax.swing.plaf.multi** : chứa các lớp và interface hỗ trợ làm việc với multiplexing look & feel.



Các thư viện của Swing

- **javax.swing.colorchooser** : chứa các lớp và interface hỗ trợ làm việc với hộp thoại lựa chọn màu sắc.
- **javax.swing.table** : chứa các lớp và interface hỗ trợ làm việc với giao diện dạng bảng
- **javax.swing.text** : chứa các lớp và interface hỗ trợ việc thao tác với dữ liệu text.
- **javax.swing.text.html** : chứa các lớp và interface hỗ trợ cho việc tạo và xem tài liệu html.



Các thư viện của Swing

- **javax.swing.text.html.parser** : chứa các lớp và interface hỗ trợ cho việc phân tích tài liệu html
- **javax.swing.text.rtf** : chứa các lớp và interface hỗ trợ cho việc thao tác với rich text format.
- **javax.swing.tree** : chứa các lớp và interface hỗ trợ cho việc tạo giao diện dưới dạng cây và quản lý dữ liệu phân cấp.
- **javax.swing.undo** : chứa các lớp hỗ trợ cho việc thực hiện các chức năng undo, redo.



Nội dung chương 5

5.1 Giới thiệu chung về AWT và Swing

5.2 Cấu trúc chung của các giao diện GUI

5.3 Giao diện với các đối tượng cơ bản

5.4 Giao diện với các đối tượng nâng cao

5.5 Các kỹ thuật tạo Tables



Cấu trúc giao diện GUI

- Giao diện GUI: tổ chức các đối tượng GUI trên 1 vật chứa (container) và quản lý bởi trình quản lý bố trí (layout manager).



NGUYÊN TẮC XÂY DỰNG GUI

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



Swing Container

- Top-lever container:
 - JFrame: sử dụng làm cửa sổ chính của chương trình
 - JApplet: sử dụng trên trình duyệt
 - JDialog: cửa sổ thông báo
- Secondary container: JPanel
- Thêm các đối tượng vào cửa sổ JFrame:
 - Không thể thêm trực tiếp mà phải tương tác qua ContentPane của JFrame
 - Đối tượng JFrame cung cấp 2 phương thức:
 - getContentPane(): trả lại 1 đối tượng ContentPane thuộc lớp Container
 - setContentPane(JPanel): thiết lập nội dung cho ContentPane



Thiết lập thuộc tính hiển thị Component

// javax.swing.Jcomponent

- setBackground(Color bgcolor) //màu nền
- setForeground(Color fgcolor) // màu chữ
- setFont(Font font) // font chữ
- Thiết lập viền
 - setBorder(Border border)
 - setPreferredSize(Dimension dim)
 - setMaximumSize(Dimension dim)
 - setMinimumSize(Dimension dim)
- setToolTipText(String s)// Thiết lập chỉ dẫn



Swing

- `setText(String s)` // thiết lập tên hiển thị trên nhãn
- `setIcon(Icon img)` //Thiết lập biểu tượng
- `SwingConstants.RIGHT` hoặc `LEFT` // canh lề ngang
- `setHorizontalAlignment(int alignment)`
- `SwingConstants.TOP` hoặc `BOTTOM` // canh lề dọc
- `setVerticalAlignment(int alignment)`
- `setHorizontalTextPosition(int textPosition)`// căn lề cho phần tên
- `setVerticalTextPosition(int textPosition)`



JFrame

- JFrame là container chính trong ứng dụng swing dùng để chứa các thành phần giao diện khác.
- JFrame thường dùng để tạo ra cửa sổ trong chương trình swing.
- Hàm khởi tạo:
 - JFrame()
 - JFrame(String title)
- Các thành phần đồ họa được đưa vào content pane, không đưa trực tiếp vào đối tượng JFrame.
 - VD: `frame.getContentPane().add(b);`



JFrame

- Khi đưa các component vào JFrame, ta đưa vào ContentPane của JRootPane này. ContentPane là một JPanel.

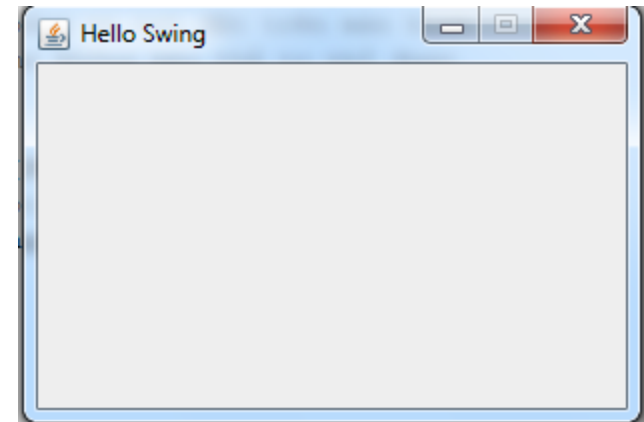
[getContentPane\(\)](#)

[setContentPane\(\)](#)



Ví dụ JFrame

```
6 package helloswing;
7 import javax.swing.JFrame;
8 public class HelloSwing extends JFrame {
9
10     public HelloSwing() {
11         super("Hello Swing");
12         setSize(300,200); // Chiều dài, rộng
13         setVisible(true); // Hiển thị
14         setLocation(500,300); // Khoảng vị trí đặt trên màn hình
15         setResizable(false); // Frame không kéo thả to nhỏ được
16     }
17
18     public static void main(String[] args) {
19         // TODO code application logic here
20         HelloSwing hl=new HelloSwing();
21     }
22 }
23
```





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 khởi tạo:
 - JPanel()
 - JPanel(LayoutManager lm)



Các thành phần giao tiếp người dùng cơ bản

- Form được dùng để thu thập thông tin từ phía người dùng.
- Trong quá trình giao tiếp, component được dùng để cho phép nhập dữ liệu là `textfield` và `textArea`.
- Để khởi tạo các phần tử, cần thực hiện các bước sau:
 - Tạo phần tử
 - Thiết lập thuộc tính (size, font, color...)
 - Xác định vị trí cho nó
 - Đưa nó vào giao diện



Các loại component

The diagram illustrates a web form with the following components and labels:

- Label**: Points to the text "Name :".
- Text field**: Points to the input box for the name.
- Favorite sports**: Points to the section containing three checkboxes: Cricket, Badminton, and Golf.
- Checkbox**: Points to one of the checkboxes in the "Favorite sports" section.
- Gender**: Points to the section containing two radio buttons: Male and Female.
- Radio button**: Points to one of the radio buttons in the "Gender" section.
- Comments**: Points to the text area for comments.
- Text Area**: Points to the input box for comments.
- Submit**: Points to the "Submit" button.
- Reset**: Points to the "Reset" button.
- Button**: Points to the "Submit" and "Reset" buttons.



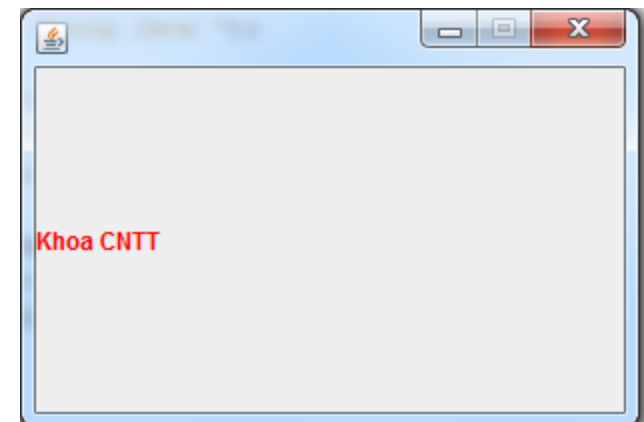
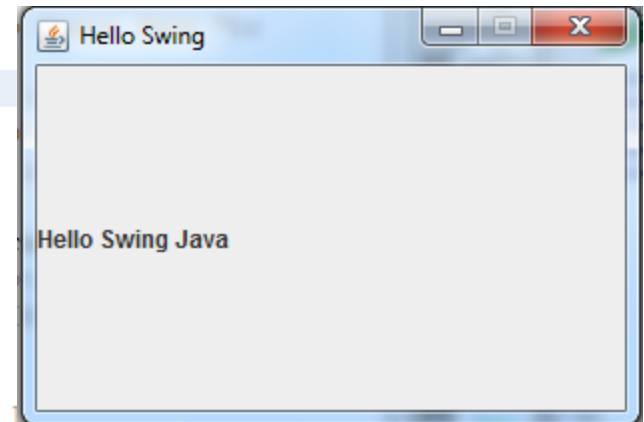
JLabel

- Được dùng để hiển thị văn bản (text) và hình ảnh (image).
- Tạo hiệu ứng trực quan cho màn hình giao diện
- Hàm khởi tạo của JLabel:
 - JLabel (Icon img)
 - JLabel (String st)
 - JLabel (String st, Icon img, int align)



Ví dụ

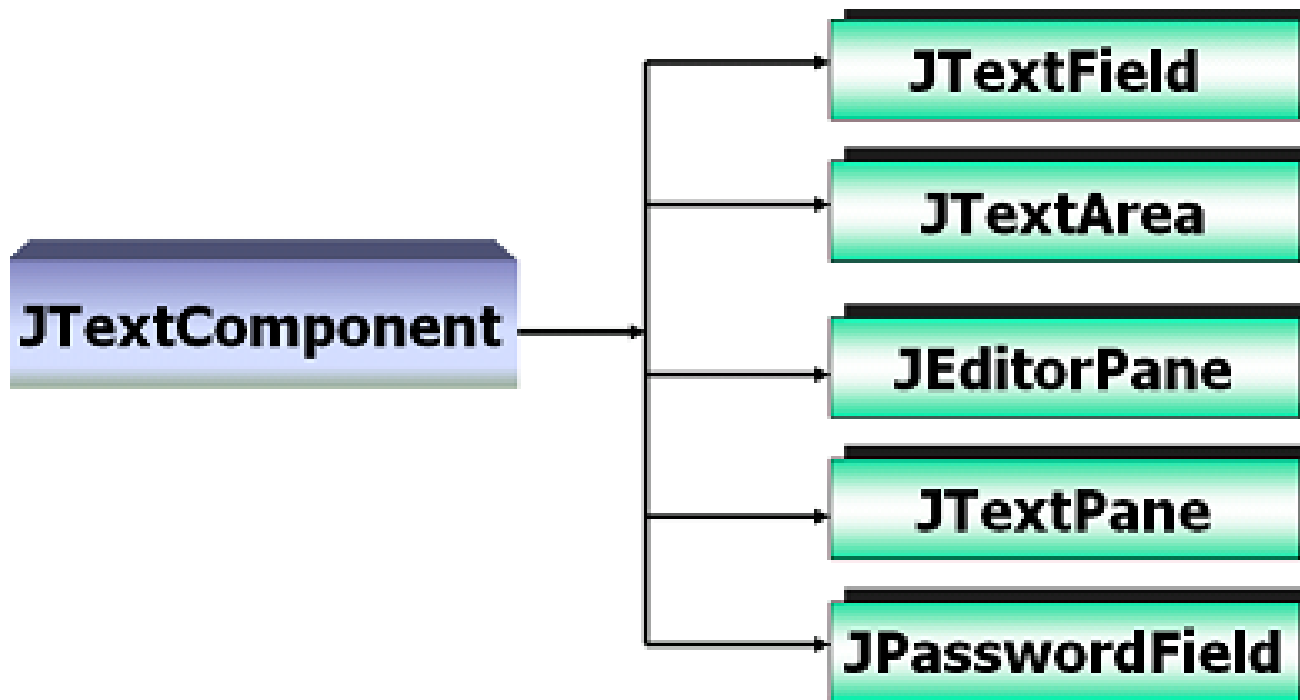
```
6 package helloswing;
7 import java.awt.Color;
8 import javax.swing.JFrame;
9 import javax.swing.JLabel;
10 public class LabelDemo extends JFrame {
11     public LabelDemo() {
12         setSize(300,200); // Chiều dài, rộng
13         setLocation(500,300); // Khoảng vị trí đặt trên màn hình
14         setResizable(false); // Frame không kéo thả to nhỏ được
15         JLabel lb=new JLabel("Hello Swing Java ");
16         add(lb);
17         lb.setText("Khoa CNTT"); // thay đổi chuỗi String trước
18         lb.setToolTipText("Trường ĐH TNMTHN"); // Nổi chữ khi trỏ đến Khoa CNTT
19         lb.setForeground(Color.red); // Màu sắc cho chữ
20     }
21     public static void main(String[] args) {
22         // TODO code application logic here
23         LabelDemo lb=new LabelDemo();
24         lb.setVisible(true);
25     }
26 }
27
```





JtextComponent

- Đây là lớp cha của tất cả các lớp hiển thị văn bản trong Swing





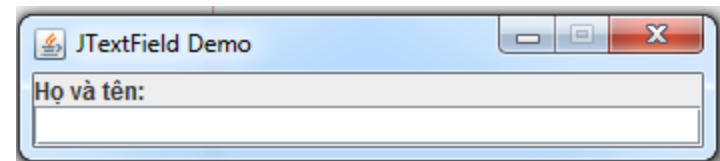
JTextField

- JTextField cho phép soạn thảo chỉ 1 dòng văn bản
- Các hàm khởi tạo của JTextField:
 - JTextField()
 - JTextField(int columns)
 - JTextField(String text)
 - JTextField(String text, int column)
- Chuỗi hiển thị được xác định trong trường text, và số cột hiển thị được xác định trong trường column.



Ví dụ

```
package helloswing;  
import javax.swing.JFrame;  
import javax.swing.JLabel;  
import javax.swing.JTextField;  
public class JTextFielddemo extends JFrame {  
    public JTextFielddemo() {  
        super("JTextField Demo");  
        setSize(300,200); // Chiều dài, rộng  
        setLocation(500,300); // Khoảng vị trí đặt trên màn hình  
        setResizable(false); // Frame không kéo thả to nhỏ được  
        JLabel lb=new JLabel("Họ và tên: ");  
        add(lb);  
        JTextField jthoten=new JTextField(30);  
        add(jthoten,"South",1);  
        pack(); // setsize vừa đủ  
    }  
    public static void main(String[] args) {  
        JTextFielddemo jtf=new JTextFielddemo();  
        jtf.setVisible(true);  
    }  
}
```





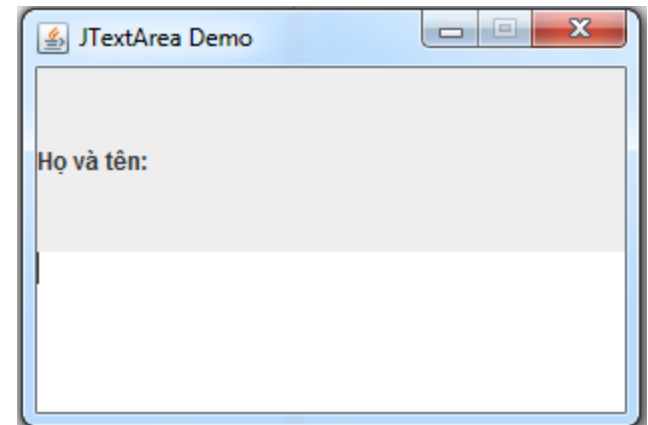
JTextArea

- Đây là component cho phép nhập vào nhiều dòng văn bản.
- Có hỗ trợ các thanh cuộn
- Các hàm khởi tạo:
 - JTextArea()
 - JTextArea(int rows, int cols)
 - JTextArea(String text)
 - JTextArea(String text, int rows, int cols)



Ví dụ JTextArea

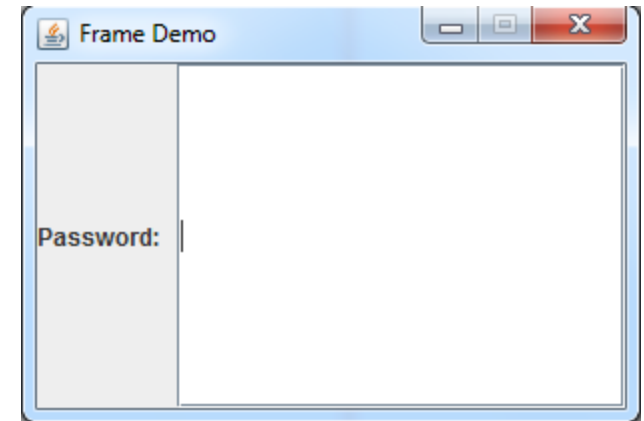
```
6 package helloswing;
7 import javax.swing.JFrame;
8 import javax.swing.JLabel;
9 import javax.swing.JTextArea;
10 public class JTextAreaDemo extends JFrame {
11     public JTextAreaDemo() {
12         super("JTextArea Demo");
13         setSize(300,200); // Chiều dài, rộng
14         setLocation(500,300); // Khoảng vị trí đặt trên màn hình
15         setResizable(false); // Frame không kéo thả to nhỏ được
16         JLabel lb=new JLabel("Họ và tên: ");
17         add(lb);
18         JTextArea jtahoten=new JTextArea(5,15);
19         add(jtahoten,"South",1);
20         //pack(); // setsize vừa đủ
21     }
22     public static void main(String[] args) {
23         JTextAreaDemo jta=new JTextAreaDemo();
24         jta.setVisible(true);
25     }
26 }
```





JPasswordField

```
2 package helloswing;
3 import javax.swing.JFrame;
4 import javax.swing.JLabel;
5 import javax.swing.JPasswordField;
6 public class Password extends JFrame {
7     public Password() {
8         super("Frame Demo");
9         setSize(300,200); // Chiều dài, rộng
10        setLocation(500,300); // Khoảng vị trí đặt cửa sổ
11        setResizable(false); // Frame không kéo thả t
12        JLabel lb=new JLabel("Password: ");
13        add(lb);
14        JPasswordField pas=new JPasswordField(20);
15        pas.setEchoChar('*');
16        add(pas, "East", 1);
17    }
18    public static void main(String[] args) {
19        Password pw=new Password();
20        pw.setVisible(true);
21    }
```





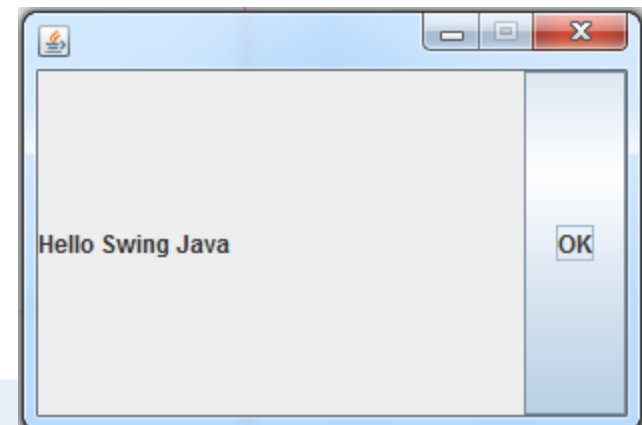
JButton

- Thể hiện chức năng nút bấm
- JButton là lớp con của lớp AbstractButton
- Đối tượng JButton bao gồm chuỗi văn bản, hình ảnh và các đường viền.
- Các hàm khởi tạo:
 - JButton();
 - JButton(Icon icon)
 - JButton(String text)
 - JButton(String text, Icon icon)
 - JButton (Action a)



Ví dụ

```
2 package helloswing;
3 import javax.swing.JFrame;
4 import javax.swing.JLabel;
5 import javax.swing.JButton;
6 public class buttonDemo extends JFrame {
7     public buttonDemo() {
8         setSize(300,200); // Chiều dài, rộng
9         setLocation(500,300); // Khoảng vị trí đặt trên màn
10        setResizable(false); // Frame không kéo thả to nhỏ
11        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //
12        JLabel lb=new JLabel("Hello Swing Java ");
13        add(lb);
14        JButton bt=new JButton("OK");
15        add(bt,"East",1);
16    }
17    public static void main(String[] args) {
18        buttonDemo bt=new buttonDemo();
19        bt.setVisible(true);
20    }
21 }
```





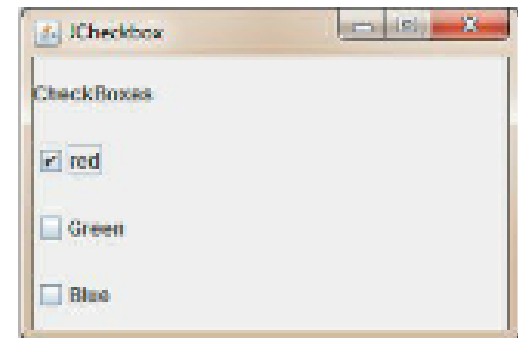
JCheckBox

- Cung cấp cho người dùng nhiều khả năng lựa chọn
- Các hàm khởi tạo:
 - JCheckBox()
 - JCheckBox(Icon icon)
 - JCheckBox(Icon icon, boolean selected)
 - JCheckBox(String text)
 - JCheckBox(String text, boolean selected)
 - JCheckBox(String text, Icon icon)
 - JCheckBox(String text, Icon icon, boolean selected)
 - JCheckBox(Action a)



Ví dụ

```
import javax.swing.*;
import java.awt.*;
class JCheckboxtest
{
    public static void main(String args[])
    {
        JLabel l=new JLabel("CheckBoxes");
        JCheckBox b1=new JCheckBox("red",true);
        JCheckBox b2=new JCheckBox("Green",false);
        JCheckBox b3=new JCheckBox("Blue",false);
        JFrame f=new JFrame("Checkbox and radiobutton");
        JPanel p=new JPanel();
        p.setLayout(new GridLayout(4,1));
        p.add(l);
        p.add(b1);
        p.add(b2);
        p.add(b3);
        f.add(p);
        f.setSize(300,200);
        f.setVisible(true);
    }
}
```





JRadioButton

- 1 tập các nút đài cho phép chỉ lựa chọn được 1 tại 1 thời điểm
- Dùng lớp ButtonGroup để tạo ra nhóm
- Các hàm dựng của lớp JRadioButton:
 - JRadioButton()
 - JRadioButton(Icon icon)
 - JRadioButton(Icon icon, boolean selected)
 - JRadioButton(String text)
 - JRadioButton(String text, boolean selected)
 - JRadioButton(String text, Icon icon)
 - JRadioButton(String text, Icon icon, boolean selected)
 - JRadioButton(Action a)



FlowLayout

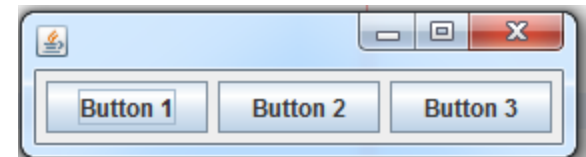
- Là một Layout manager quản lý việc sắp xếp các component từ góc trái trên đến góc phải dưới màn hình.
- Lớp FlowLayout có các hàm khởi tạo:
 - `public FlowLayout()`
 - `public FlowLayout(int alignment)`
 - `public FlowLayout(int alignment, int horizontalGap, int verticalGap)`

Trong đó: alignment có các giá trị `FlowLayout.RIGHT` hoặc `LEFT` hoặc `CENTER`. Thông số `horizontalGap`, `verticalGap` xác định số pixel đặt giữa các thành phần, thường đặt là 5.



FlowLayout

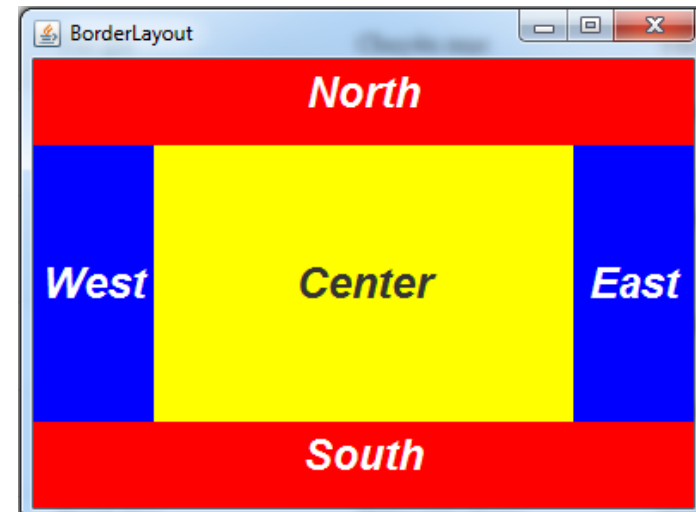
```
public class LayoutDemo extends JFrame {  
    public LayoutDemo() {  
        setSize(300,200); // Chiều dài, rộng  
        setLocation(500,300); // Khoảng vị trí đặt trên màn  
        //setResizable(false); // Frame không kéo thả to nhỏ  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        FlowLayout lb=new FlowLayout();  
        Container c= getContentPane();  
        c.setLayout( lb);  
        c.add(new JButton("Button 1"));  
        c.add(new JButton("Button 2"));  
        c.add(new JButton("Button 3"));  
    }  
  
    public static void main(String[] args) {  
        LayoutDemo lo=new LayoutDemo();  
        lo.setDefaultCloseOperation(EXIT_ON_CLOSE);  
        lo.pack();  
        lo.setVisible(true);  
    }  
}
```





BorderLayout

- Là layout manager mặc định cho Windows, JFrame và Dialog.
- Sắp xếp tối đa 5 thành phần trong 1 container:
 - North: đặt đỉnh container
 - East: đặt bên phải container
 - South: đặt phía dưới container
 - West: Đặt bên trái container
 - Center: đặt ở giữa container





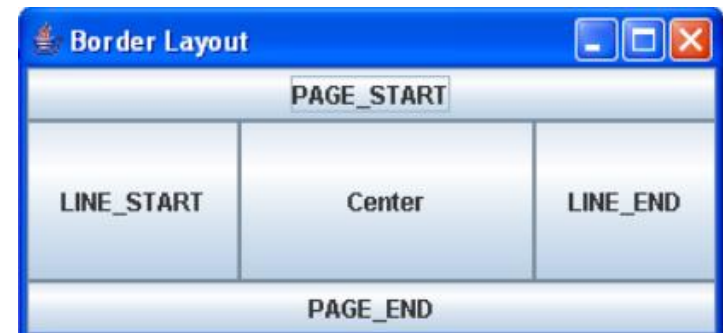
BorderLayout

- Thêm 1 thành phần vào vùng North:
`JButton b1=new JButton("Button 1");`
`setLayout (new BorderLayout());`
`add(b1, BorderLayout.NORTH);`
- Các thành phần giữ nguyên vị trí tương đối của chúng kể cả khi container bị thay đổi kích thước.



Ví dụ

```
import java.awt.*;  
import javax.swing.*;  
public class BorderLayoutTest {  
    public static void main(String args[]) {  
        JFrame frame = new JFrame("Border Layout");  
        Container contentPane = frame.getContentPane();  
        contentPane.setLayout(new BorderLayout());  
        contentPane.add(new JButton("PAGE_START"), BorderLayout.PAGE_START);  
        contentPane.add(new JButton("PAGE_END"), BorderLayout.PAGE_END);  
        contentPane.add(new JButton("LINE_END"), BorderLayout.LINE_END);  
        contentPane.add(new JButton("LINE_START"), BorderLayout.LINE_START);  
        contentPane.add(new JButton("Center"), BorderLayout.CENTER);  
        frame.setSize(300, 200);  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        frame.pack();  
        frame.show();  
    }  
}
```





GridLayout Manager

- Trợ giúp việc chia container thành ô lưới
- Mỗi ô lưới nên chứa ít nhất một thành phần
- Dùng GridLayout khi tất cả các thành phần có cùng kích thước
- Hàm khởi tạo:
 - `GridLayout g1=new GridLayout();`
 - `GridLayout g1=new GridLayout(int hang, int cot);`
 - `GridLayout g1=new GridLayout(int hang, int cot, int định dạng hàng, int định dạng cột);`



JWindow

- JWindow cũng tương tự như JFrame ngoại trừ việc JWindow không có title bar.
- Thông thường, ta thường sử dụng JWindow để hiển thị một thông điệp hoặc một màn hình splash



Các component khác

- Tham khảo tại trang web:
- <http://java.sun.com/docs/books/tutorial/uiswing/components>



Mô hình xử lý sự kiện

- Sự kiện (Event) được phát sinh khi người dùng tương tác với GUI. VD: di chuyển chuột, ấn nút, nhập văn bản, chọn menu...
- Thông tin về sự kiện được lưu trong 1 đối tượng sự kiện thuộc lớp con của lớp AWTEvent(gói java.awt.event)
- Chương trình có thể xử lý sự kiện bằng cách đặt “lắng nghe sự kiện ” trên các thành phần GUI.



Mô hình xử lý sự kiện

- Ba thành phần chính của mô hình gồm:
- **Event source:** nguồn gây ra sự kiện, thường là các thành phần GUI trong chương trình.
- **Event object:** đối tượng lưu thông tin về sự kiện đã xảy ra
- **Event listener:** đối tượng sẽ nhận được thông tin khi có sự kiện xảy ra.

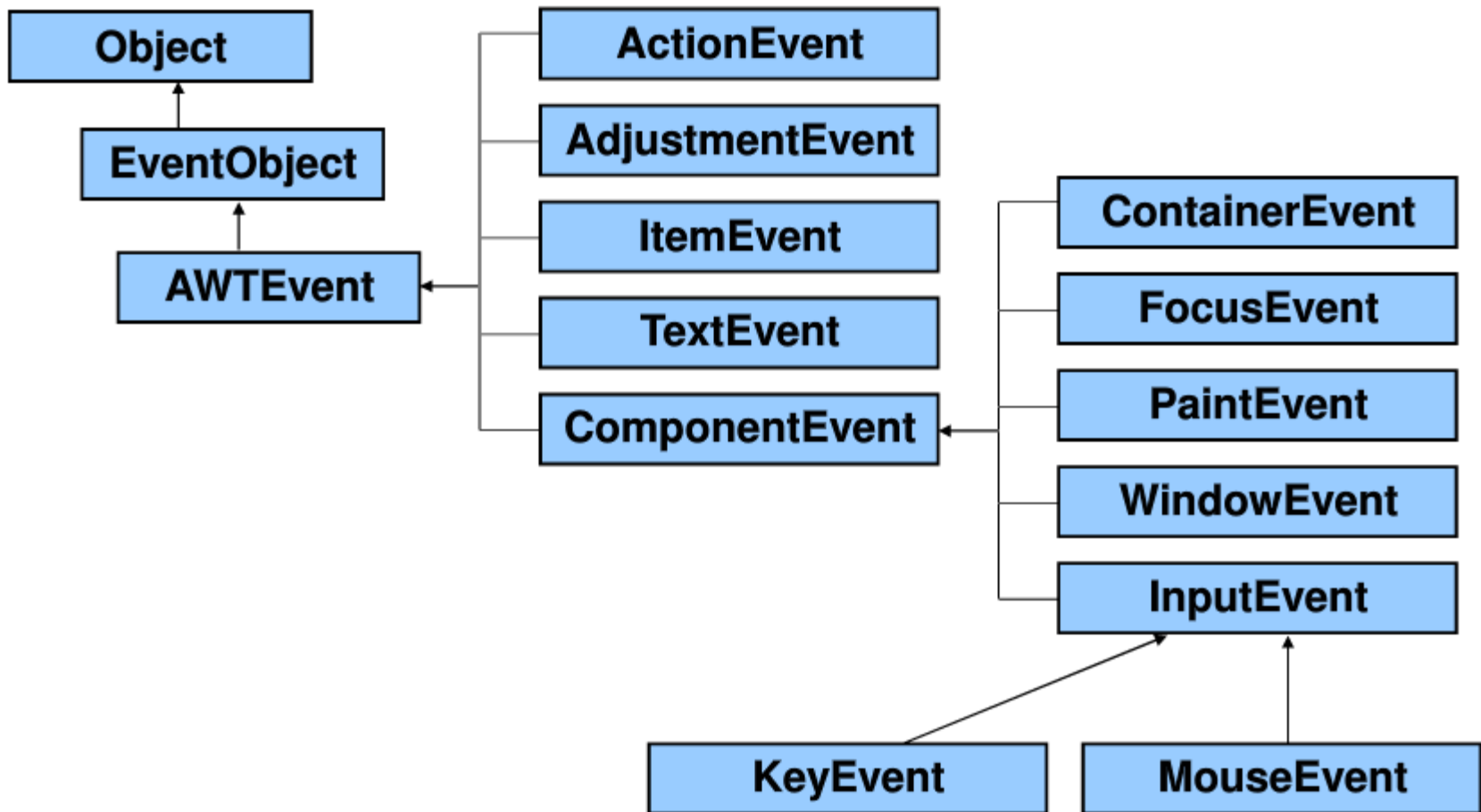


MÔ HÌNH XỬ LÝ SỰ KIỆN

- Nguồn sự kiện
 - Các lớp thành phần GUI mà người sử dụng tương tác.
 - Bạn có thể đăng ký “Listener” đáp ứng với những sự kiện nhất định
- Bộ lắng nghe (Listener)
 - Nhận đối tượng sự kiện khi được thông báo và thực hiện đáp ứng thích hợp.
 - Nhiều kiểu của bộ lắng nghe tồn tại cho các sự kiện cụ thể như MouseListener, ActionListener, KeyListener,...
 - Các giao tiếp được hiện thực và cài đặt các hành động
- Đối tượng sự kiện (Event)
 - Đóng gói thông tin về sự kiện xuất hiện
 - Các đối tượng sự kiện được gửi tới bộ lắng nghe khi sự kiện xuất hiện trên thành phần GUI



Gói java.awt.event.*





Mô hình xử lý sự kiện



- Việc thông báo sự kiện xảy ra thực chất là việc gọi một phương thức của EventListener với đối số truyền vào là EventObject.
- Các lớp con của EventListener có thể cài đặt các phương thức để xử lý sự kiện.



Mô hình xử lý sự kiện

- Lớp hiện thực giao tiếp bộ lắng nghe sự kiện (bộ xử lý sự kiện).
 - VD: class Circle extends JFrame implements ActionListener{...}
- Đăng ký bộ lắng nghe sự kiện cho nguồn sự kiện
 - VD: btExit.addActionListener(handler);
- Cài đặt phương thức xử lý sự kiện (các phương thức của giao tiếp bộ lắng nghe sự kiện).
 - VD: với bộ lắng nghe sự kiện ActionListener cần cài đặt phương thức:
 - public void actionPerformed(ActionEvent e){...}



Ví dụ

- Bước 1: Nhúng vào gói event: `import java.awt.event.*;`
- Bước 2: Cài đặt giao diện listener thích hợp
Ví dụ: `class abc extends Frame implements ActionListener`
- Bước 3: Cho đối tượng cần bắt sự kiện lắng nghe sự kiện
Ví dụ: `button.addActionListener(this)`
- Bước 4: Viết hàm để xử lý sự kiện cho tương ứng
Ví dụ :

```
public void actionPerformed(ActionEvent x)
{
}

```



Ví dụ

```
import javax.swing.*;
import java.awt.*;
public class Demo extends JFrame implements ActionListener
{
    JButton btOK;
    public Demo() {
        ...
        btOK.addActionListener(this);
        ...
    }
    public void actionPerformed (ActionEvent e) { ....}
}
```




Mô hình xử lý sự kiện

Event source	Event	Chú thích
Button	ActionEvent	Nhấn nút
Checkbox	ItemEvent	Chọn, bỏ chọn một item
Choice	ItemEvent	Chọn, bỏ chọn một item
Component	ComponentEvent	Ẩn, hiện, di chuyển
	FocusEvent	Được chọn
	MouseEvent	Tương tác chuột
	KeyEvent	Tương tác bàn phím
Container	ContainerEvent	Thêm, bớt component
List	ActionEvent	Nhấp kép chuột một item
	ItemEvent	Chọn, bỏ chọn một item



Mô hình xử lý sự kiện

Event source	Sự kiện	Chú thích
MenuItem	ActionEvent	Chọn một menu item
Scrollbar	AdjustmentEvent	Di chuyển thanh cuộn
TextComponent	TextEvent	Thay đổi văn bản
TextField	ActionEvent	Kết thúc thay đổi văn bản
Window	WindowEvent	Thay đổi cửa sổ



Mô hình xử lý sự kiện

Event Class	Listener Interface	Listener Methods
ActionEvent	ActionListener	actionPerformed()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
ComponentEvent	ComponentListener	componentHidden() componentMoved() componentResized() componentShown()
ContainerEvent	ContainerListener	componentAdded() componentRemoved()
FocusEvent	FocusListener	focusGained() focusLost()
ItemEvent	ItemListener	itemStateChanged()



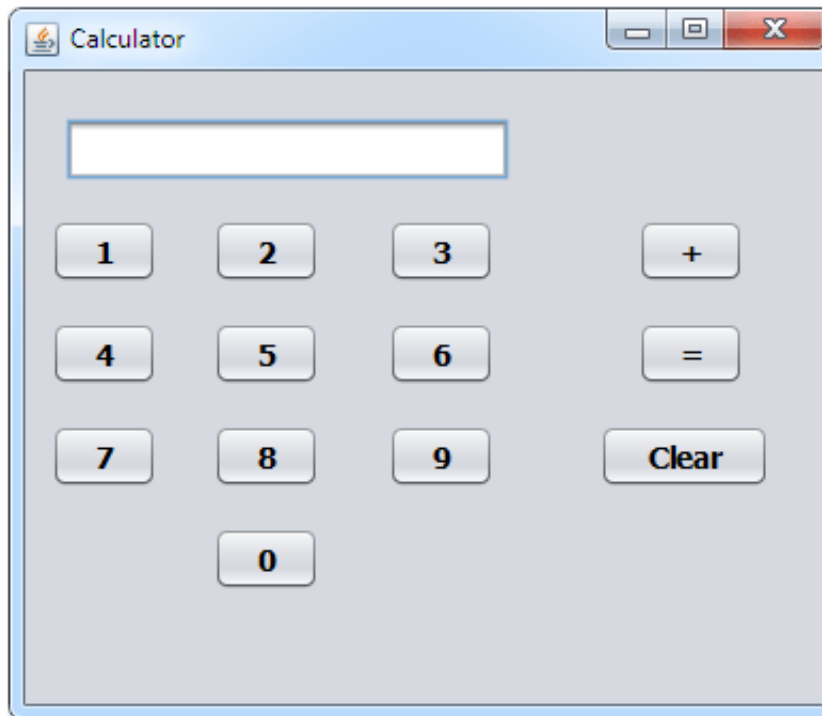
Mô hình xử lý sự kiện

Event Class	Listener Interface	Listener Methods
KeyEvent	KeyListener	keyPressed() keyReleased() keyTyped()
MouseEvent	MouseListener	mouseClicked() mousePressed() mouseReleased()
	MouseMotionListener	mouseDragged() mouseMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowClosed() windowActivated()



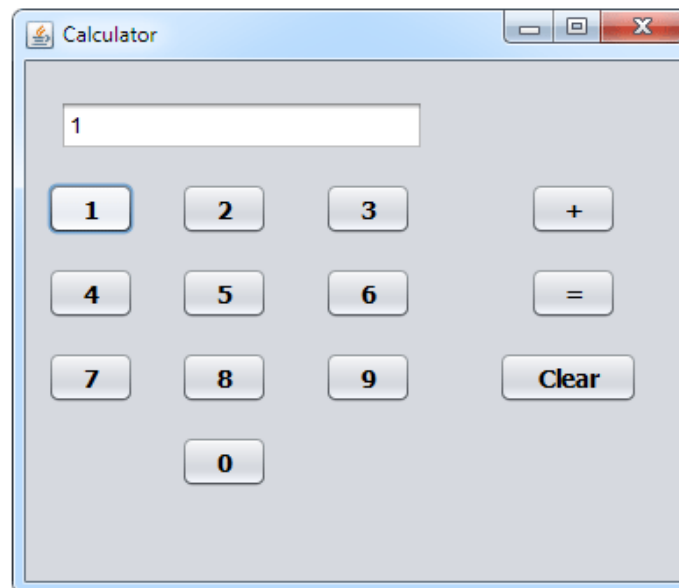
Ví dụ

Thiết kế giao diện và viết chương trình sau:





```
private void btnOneActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String btnOneText = btnOne.getText( );  
    txtDisplay.setText(btnOneText);  
  
}
```





```
private void btnOneActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String btnOneText = txtDisplay.getText() + btnOne.getText();  
    txtDisplay.setText(btnOneText);  
  
}  
  
private void btnTwoActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String btnTwoText = txtDisplay.getText() + btnTwo.getText();  
    txtDisplay.setText(btnTwoText);  
  
}  
  
private void btnThreeActionPerformed(java.awt.event.ActionEvent evt) {  
  
    String btnThreeText = txtDisplay.getText() + btnThree.getText();  
    txtDisplay.setText(btnThreeText);  
  
}  
  
private void btnFourActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnFourText = txtDisplay.getText() + btnFour.getText();  
    txtDisplay.setText(btnFourText);  
  
}  
  
private void btnFiveActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnFiveText = txtDisplay.getText() + btnFive.getText();  
    txtDisplay.setText(btnFiveText);  
  
}
```



Ví dụ

```
private void btnSixActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnSixText = txtDisplay.getText() + btnSix.getText();  
    txtDisplay.setText(btnSixText);  
}  
  
private void btnSevenActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnSevenText = txtDisplay.getText() + btnSeven.getText();  
    txtDisplay.setText(btnSevenText);  
}  
  
private void btnEightActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnEightText = txtDisplay.getText() + btnEight.getText();  
    txtDisplay.setText(btnEightText);  
}  
  
private void btnNineActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnNineText = txtDisplay.getText() + btnNine.getText();  
    txtDisplay.setText(btnNineText);  
}  
  
private void btnZeroActionPerformed(java.awt.event.ActionEvent evt) {  
    String btnZeroText = txtDisplay.getText() + btnZero.getText();  
    txtDisplay.setText(btnZeroText);  
}
```




VÍ DỤ

Xây dựng một chương trình như sau:

- Khi nhấn vào button Red hoặc button Green hoặc button Blue thì nền của cửa sổ chương trình thay đổi màu tương ứng, đồng thời label bên dưới các button cũng có câu thông báo màu tương ứng.





VÍ DỤ (file MyFirstAwt.java)

```
import java.awt.*;
import java.awt.event.*;
public class MyFirstAwt extends Frame
{
    Label status;
    Button button1 = new Button("Red");
    Button button2 = new Button("Green");
    Button button3 = new Button("Blue");

    MyFirstAwt()
    {
        this.setTitle("My First Awt"); //super("My First Awt");
        this.setLayout(new FlowLayout());
        this.add(button1);
        this.add(button2);
        this.add(button3);
        status = new Label();
        status.setText("Press any button, please!");
        this.add(status);
    }
}

//xem tiếp ở slide tiếp theo
```



VÍ DỤ (file MyFirstAwt.java) - tt

```
button1.addActionListener(new MyListener(status,this));
button2.addActionListener(new MyListener(status,this));
button3.addActionListener(new MyListener(status,this));

this.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent evt){System.exit(0);}
});

}

public static void main(String[] args)
{
    MyFirstAwt mfa = new MyFirstAwt();
    mfa.resize(300,200);
    mfa.show();
}
}
```



VÍ DỤ (file MyListener.java)

```
import java.awt.*;  
import java.awt.event.*;  
  
public class MyListener implements ActionListener  
{  
  
    Label status;  
    Component compo;  
  
    MyListener(Label status1, Component compo1)  
    {  
        this.status = status1;  
        this.compo = compo1;  
    }  
  
    //xem tiếp ở slide tiếp theo
```



VÍ DỤ (file MyListener.java) - tt

```
public void actionPerformed(ActionEvent evt)
{
    if(evt.getSource() instanceof Button)
    {
        Button temp = (Button)evt.getSource();
        status.setText("You have selected: " + temp.getLabel());
        if(temp.getLabel().equalsIgnoreCase("Red"))
        {
            compo.setBackground(new Color(255,0,0));
        }
        if(temp.getLabel().equalsIgnoreCase("Green"))
        {
            compo.setBackground(new Color(0,255,0));
        }
        if(temp.getLabel().equalsIgnoreCase("Blue"))
        {
            compo.setBackground(new Color(0,0,255));
        }
    }
}
```



Xây dựng một chương trình như sau:

- Nhập vào hai số rồi nhấn button Sum để tính tổng

The screenshot shows a Windows application window titled "My Addition Operator". Inside the window, there are three input fields and two buttons. The first input field is preceded by the text "Enter first number:". The second input field is preceded by the text "Enter second number:". The third input field is preceded by the text "The sum is:". Below the input fields, there are two buttons: "Sum" and "Exit".



VÍ DỤ (AddOperator.java)

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

public class AddOperator extends Frame implements ActionListener
{
    Label firstLabel = new Label("Enter first number:");
    Label secondLabel = new Label("Enter second number:");
    Label resultLabel = new Label("The sum is:");
    TextField firstTextField = new TextField(5);
    TextField secondTextField = new TextField(5);
    TextField resultTextField = new TextField(5);
    Button sumButton = new Button("Sum");
    Button exitButton = new Button("Exit");

    AddOperator()
    {
        this.setTitle("My Addition Operator");
        this.setLayout(null);
        sumButton.setBounds(100,150,50,30);
        this.add(sumButton);
        sumButton.addActionListener(this);
    }
    //xem tiếp ở slide tiếp theo
}
```



VÍ DỤ (AddOperator.java) - tt

```
exitButton.setBounds(200,150,50,30);
this.add(exitButton);
exitButton.addActionListener(this);
firstLabel.setBounds(50,50,130,30);
this.add(firstLabel);
secondLabel.setBounds(50,80,130,30);
this.add(secondLabel);
resultLabel.setBounds(50,110,130,30);
this.add(resultLabel);
firstTextField.setBounds(190,50,80,25);
this.add(firstTextField);
secondTextField.setBounds(190,80,80,25);
this.add(secondTextField);
resultTextField.setBounds(190,110,80,25);
this.add(resultTextField);
this.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent evt){System.exit(0);}
});
```

} //xem tiếp ở slide tiếp theo



VÍ DỤ (AddOperator.java) - tt

```
public void actionPerformed(ActionEvent evt)
{
    if(evt.getSource()==sumButton)
    {
        int firstNum = Integer.parseInt(firstTextField.getText());
        int secondNum = Integer.parseInt(secondTextField.getText());
        int resultNum = firstNum + secondNum;
        resultTextField.setText(String.valueOf(resultNum));
    }
    if(evt.getSource()==exitButton)
    {
        System.exit(0);
    }
}

public static void main(String[] args)
{
    AddOperator ao = new AddOperator();
    ao.setBounds(10,10,400,200);
    ao.setVisible(true);
}
}
```



Ví dụ

BT1: Thiết kế giao diện và viết chương trình thực hiện các phép toán $+$, $-$, $*$, $/$ 2 số tự nhiên a , b .

BT 2: Thiết kế giao diện và giải phương trình bậc 2

BT 3 : Quản lý điểm sinh viên.

XỬ LÝ SỰ KIỆN CHUỘT



XỬ LÝ SỰ KIỆN CHUỘT

- Java cung cấp hai interfaces lắng nghe (bộ lắng nghe sự kiện chuột) là `MouseListener` và `MouseMotionListener` để quản lý và xử lý các sự kiện liên quan đến thiết bị chuột.
- Những sự kiện chuột có thể “bẫy” cho bất kỳ component nào trên GUI mà dẫn xuất từ `java.awt.component`.



XỬ LÝ SỰ KIỆN CHUỘT

Các phương thức của interface `MouseListener`:

- ***`public void mousePressed(MouseEvent event)`***: được gọi khi một nút chuột được nhấn và con trỏ chuột ở trên component.
- ***`public void mouseClicked(MouseEvent event)`***: được gọi khi một nút chuột được nhấn và nhả trên component mà không di chuyển chuột.
- ***`public void mouseReleased(MouseEvent event)`***: được gọi khi một nút chuột nhả sau khi kéo rê.
- ***`public void mouseEntered(MouseEvent event)`***: được gọi khi con trỏ chuột vào trong đường biên của một component.
- ***`public void mouseExited(MouseEvent event)`***: được gọi khi con trỏ chuột ra khỏi đường biên của một component.



XỬ LÝ SỰ KIỆN CHUỘT

Các phương thức của interface MouseMotionListener:

- ***public void mouseDragged(MouseEvent event):***

phương thức này được gọi khi người dùng nhấn một nút chuột và kéo trên một component.

- ***public void mouseMoved(MouseEvent event):***

phương thức này được gọi khi di chuyển chuột trên component.

Mỗi phương thức xử lý sự kiện chuột có một tham số.

MouseEvent chứa thông tin về sự kiện chuột phát sinh chẳng hạn như: tọa độ x, y nơi sự kiện chuột xảy ra.

Những phương thức tương ứng trong các interfaces sẽ tự động được gọi khi chuột tương tác với một component.



XỬ LÝ SỰ KIỆN CHUỘT

Ví dụ. Chương trình tên MouseTracker bên dưới minh họa việc dùng những phương thức của các interfaces `MouseListener` và `MouseMotionListener` để “bắt” và xử lý các sự kiện chuột tương ứng.

```
import java.awt.*; import java.awt.event.*;
public class MouseTracker extends Frame implements MouseListener, MouseMotionListener
{
    private Label statusBar;
    public MouseTracker()
    {
        super( "Demonstrating Mouse Events" );
        statusBar = new Label();
        this.add( statusBar, BorderLayout.SOUTH );
        addMouseListener( this );
        addMouseMotionListener( this );
        setSize( 275, 100 );
        setVisible( true );
    }
    public void mouseClicked( MouseEvent event )
    {
        String str_bt = new String();
        int count = event.getClickCount();
        int mousebutton = event.getButton();
        if(mousebutton == MouseEvent.BUTTON1) str_bt = "left mouse button";
        if(mousebutton == MouseEvent.BUTTON3) str_bt = "right mouse button";
        if(mousebutton == MouseEvent.BUTTON2) str_bt = "middle mouse button";
        statusBar.setText(str_bt + " clicked at (" + event.getX() + ","
                                + event.getY() + ")" + count + " lần");
    }
} //xem ở slide kế tiếp
```



XỬ LÝ SỰ KIỆN CHUỘT

```
public void mousePressed( MouseEvent event )
{
    statusBar.setText("Pressed at [" + event.getX() + ", " + event.getY() + "]" );
}
public void mouseReleased( MouseEvent event )
{
    statusBar.setText("Released at [" + event.getX() + ", " + event.getY() + "]" );
}
public void mouseEntered( MouseEvent event )
{
    statusBar.setText( "Mouse in window" );
}
public void mouseExited( MouseEvent event )
{
    statusBar.setText( "Mouse outside window" );
}
public void mouseDragged( MouseEvent event )
{
    statusBar.setText("Dragged at [" + event.getX() + ", " + event.getY() + "]" );
}
public void mouseMoved( MouseEvent event )
{
    statusBar.setText("Moved at [" + event.getX() + ", " + event.getY() + "]" );
}
public static void main( String args[] )
{
    MouseTracker application = new MouseTracker();
}
}
```


XỬ LÝ SỰ KIỆN BÀN PHÍM



XỬ LÝ SỰ KIỆN BÀN PHÍM

- Để xử lý sự kiện bàn phím java hỗ trợ một bộ lắng nghe sự kiện đó là interface *KeyListener*. Một sự kiện bàn phím được phát sinh khi người dùng nhấn và nhả một phím trên bàn phím. Một lớp hiện thực *KeyListener* phải cài đặt các phương thức *keyPressed*, *keyReleased* và *keyTyped*. Mỗi phương thức này có một tham số là một đối tượng kiểu *KeyEvent*. *KeyEvent* là lớp con của lớp *InputEvent*.



XỬ LÝ SỰ KIỆN BÀN PHÍM

• Các phương thức của interface *KeyListener*

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



XỬ LÝ SỰ KIỆN BÀN PHÍM

Ví dụ: minh họa việc xử lý sự kiện chuột thông qua các phương thức của interface *KeyListener*. Lớp *KeyDemo* bên dưới hiện thực interface *KeyListener*, vì vậy tất cả 3 phương thức trong *KeyListener* phải được cài đặt trong chương trình.

```
// KeyDemo.java
// Demonstrating keystroke events.
// Java core packages
import java.awt.*;
import java.awt.event.*;
public class KeyDemo extends Frame implements KeyListener
{
    private String line1 = "", line2 = "";
    private String line3 = "";
    private TextArea textArea;
//xem tiếp ở slide tiếp theo
```



XỬ LÝ SỰ KIỆN BÀN PHÍM

// set up GUI

```
public KeyDemo()  
{  
    super( "Demonstrating Keystroke Events" );  
    textArea = new TextArea( 10, 15 ); // set up TextArea  
    textArea.setText( "Press any key on the keyboard..." );  
    textArea.setEnabled( false );  
    this.add( textArea );  
    addKeyListener( this ); // allow frame to process Key events  
    setSize( 350, 100 );  
    setVisible( true );  
}  
// handle press of any key  
public void keyPressed( KeyEvent event )  
{  
    line1 = "Key pressed: " +  
        event.getKeyText( event.getKeyCode() );  
    setLines2and3( event );  
} //xem tiếp ở slide tiếp theo
```



XỬ LÝ SỰ KIỆN BÀN PHÍM

```
// handle release of any key
public void keyReleased( KeyEvent event )
{
    line1 = "Key released: " + event.getKeyText( event.getKeyCode() );
    setLines2and3( event );
}

// handle press of an action key
public void keyTyped( KeyEvent event )
{
    line1 = "Key typed: " + event.getKeyChar();
    setLines2and3( event );
}

//xem tiếp ở slide tiếp theo
```



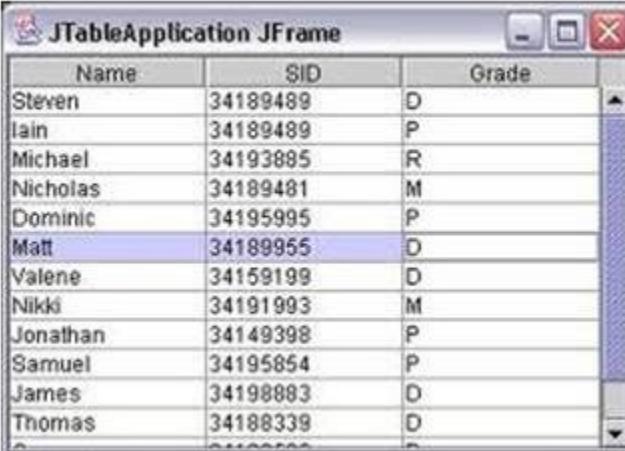
XỬ LÝ SỰ KIỆN BÀN PHÍM

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



JTable

- Là component cho phép hiển thị thông tin dưới dạng bảng.
- Dùng để hiển thị những thông tin phức tạp, khối lượng thông tin nhiều.
- VD: danh sách nhân viên, bảng điểm sinh viên, danh sách hàng hoá...



Name	SID	Grade
Steven	34189489	D
Iain	34189489	P
Michael	34193895	R
Nicholas	34189481	M
Dominic	34195995	P
Matt	34189955	D
Valene	34159199	D
Nikki	34191993	M
Jonathan	34149398	P
Samuel	34195854	P
James	34198883	D
Thomas	34188339	D



JTable

- Bước 1: Khai báo đối tượng Jtable
- Bước 2: Xây dựng class có tên TableModel thừa kế từ lớp trừu tượng AbstractTableModel để đọc thông tin.
 - Có thể định nghĩa 1 số phương thức để sử dụng cho các thuộc tính cơ bản của đối tượng JTable: phương thức cho phép đọc số cột, số hàng:
getColumnCount(); getRowCount(); getValueAt(int row, int col); getColumnName(int col);...
- Bước 3: Tạo 1 đối tượng của lớp TableModel vừa định nghĩa ở trên, sau đó gắn vào Jtable đã tạo thông qua setModel của đối tượng Jtable.



JTable

TableModel

- ◆ `getRowCount() : int`
- ◆ `getColumnCount() : int`
- ◆ `getValueAt(row : int, column : int) : Object`
- ◆ `getColumnName(column : int) : String`
- ◆ `getColumnClass(column : int) : Class`
- ◆ `isCellEditable(row : int, column : int) : boolean`
- ◆ `addTableModelListener(listener : TableModelListener) : void`
- ◆ `removeTableModelListener(listener : TableModelListener) : void`



JTable

Tạo mới 1 project: new/ Project và chọn Java Desktop Application.





JTable

- Chọn new/ File và tạo 1 Dialog Form cho giao diện Swing GUI Form.





JTable

Bước 1: Chọn biểu tượng Table trong nhóm Swing control trên thanh công cụ Pallete.





JTable

Định vị trí và đặt tên cho Jtable này. Kết quả là:

Title 1	Title 2	Title 3	Title 4



JTable

```
public class rsTableModel extends AbstractTableModel {
    //-- Khai báo 2 Vector Object để sử dụng cho JTable
    private Vector colHeaders;      //-- Chứa thông tin là tên của các Field dùng làm ColumnHeader
    private Vector tbData;         //-- Phần chứa dữ liệu của JTable
    /**
    * Constructor truyền vào 1 ResultSet để gán dữ liệu cho JTable có Model tham chiếu tới
    * @param rset - ResultSet chứa dữ liệu đọc từ Database
    * @throws SQLException
    */
    public rsTableModel(ResultSet rsData) throws SQLException {
        ResultSetMetaData rsMeta = rsData.getMetaData(); //-- Đọc MetaData của ResultSet
        int count = rsMeta.getColumnCount(); //-- Xác định số Field trong ResultSet

        colHeaders = new Vector(count);
        tbData = new Vector();
        //-- Lập tương ứng với số phần tử của columnHeaders để lấy tên của từng cột trong bảng
        for (int i = 1; i <= count; i++) {
            colHeaders.addElement(rsMeta.getColumnName(i));
        }
        //-- Lập từ Record đầu tiên đến cuối ResultSet để lấy dữ liệu và Add vào Table
        while (rsData.next()) {
            //-- Khai báo 1 Object Vector có tên là rowData để chứa dữ liệu tương ứng với mỗi dòng đọc từ Table
            Vector dataRow = new Vector(count);
            //-- Lập dựa theo số cột của bảng để lấy thông tin của từng đối tượng
            for (int i = 1; i <= count; i++) {
                dataRow.addElement(rsData.getObject(i));
            }
            tbData.addElement(dataRow);
        }
    }
}
```



JTable

```
/* Định nghĩa phương thức lấy số cột của JTable  
/* @return  
*/  
public int getColumnCount() {  
    return colHeaders.size();  
}  
/**  
/* Định nghĩa phương thức lấy số hàng có trong JTable  
/* @return  
*/  
public int getRowCount() {  
    return tbData.size();  
}  
/**  
/* Định nghĩa phương thức lấy dữ liệu tại 1 ô nào đó của bảng  
/* @param row - Vị trí của hàng chứa ô dữ liệu cần lấy  
/* @param column - Vị trí của cột chứa ô dữ liệu cần lấy  
/* @return Giá trị trả về cho nơi gọi là 1 Object chứa giá trị, để dùng phải Cast lại  
*/  
public Object getValueAt(int row, int column) {  
    Vector rowData = (Vector) (tbData.elementAt(row));  
    return rowData.elementAt(column);  
}  
/**  
/* Định nghĩa lại hàm lấy tên của cột thứ n trong JTable  
/* @param column - Vị trí của cột muốn lấy tên  
/* @return - Kết quả trả về là tên của cột ở dạng chuỗi ký tự  
*/  
@Override  
public String getColumnName(int column) {  
    return (String) (colHeaders.elementAt(column));  
}  
}
```




JTable

```
16  /**
17  *
18  * @author Nguyen Mai Huy
19  */
20  public class dsNguoiQuen extends javax.swing.JDialog {
21
22      /** Creates new form dsNguoiQuen */
23      public dsNguoiQuen(java.awt.Frame parent, boolean modal) {
24          super(parent, modal);
25          initComponents();
26          this.napDuLieu();
27      }
28
29      private void napDuLieu() {
30          try {
31              connectSQL kn = new connectSQL();
32              String strQ = "Select maNQ, hoNQ, tenNQ, gioiTinh, ";
33                  strQ += "ngaySinhNQ, soDD, eMailNQ, diaChiNQ from ttNguoiQuen";
34              ResultSet kq = kn.getData(strQ);
35              this.bangDL.setModel(new rsTableModel(kq));
36          } catch (Exception e) {
37              e.printStackTrace();
38          }
39      }
```



JTable

Tạo 1 MenuItem trong hệ thống chương trình chính, sau đó tạo phương thức actionPerformed như sau:

```
262 private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {  
263     // TODO add your handling code here:  
264     dsNguoiQuen ds=new dsNguoiQuen(null, true);  
265     ds.setVisible(true);  
266 }
```



JTable

Kết quả:

maNQ	hoNQ	tenNQ	gioiTinh	ngaySinhNQ	soDD	eMailNQ	diaChiNQ
001	Nguyễn Minh	Tân	true	1973-12-28 0...	0912234567	tan.mailtemp@gmail.com	Phú Nhuận
002	Hoàng	Tùng	true	1972-02-04 0...	0918888888	htungnttc@gmail.com	Bình Thạnh
003	Nguyễn Hoàng	Dung	true	1973-07-24 0...	0913222222	hoangdung247@gmail.com	Gò Vấp
006	Lê Phẩm	Trọng	true	1970-04-01 0...	0913232323	lptrong@yahoo.com	Hóc Môn