Fpt University

FPT POLYTECHNIC

LẬP TRÌNH JAVA

Bài 8: Giới thiệu về Swing

www.poly.edu.vn

Điểm danh

Nhắc lại bài trước

- Khái niệm Applets
- Sự khác nhau giữa Applets và Applications
- Vòng đời của applet
- Một số phương thức của class Graphics
- Tạo một applet
- Sử dụng tham số trong Applets
- Xử lý sự kiện
- Một số interface và class xử lý sự kiện
- Một số ví dụ

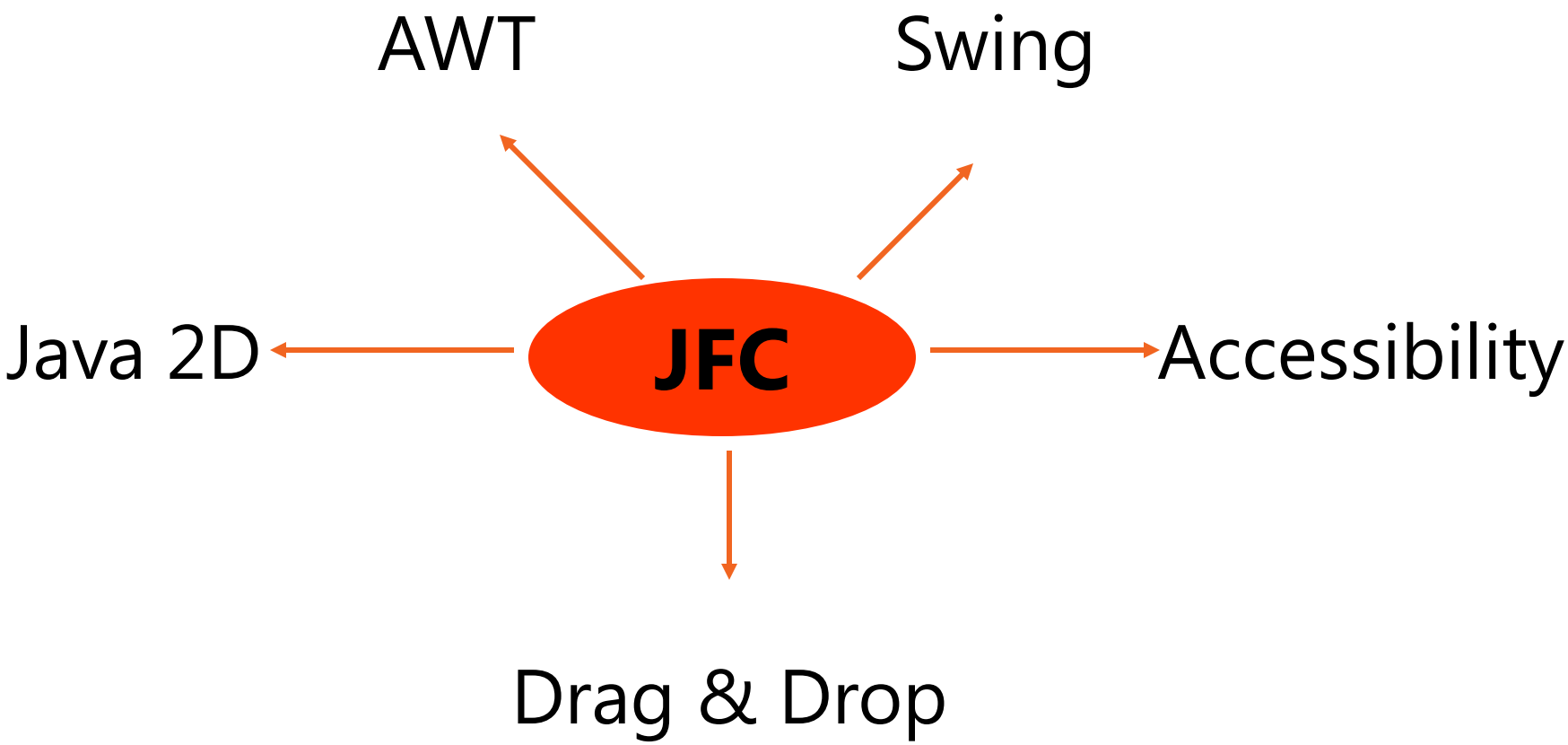
Nội dung bài học

- Java Foundation Classes (AFC)
- Abstract Windowing Toolkit (AWT)
- Swing
- Các thùng chứa -Containers
- Các hộp thoại - Dialogs
- Quản lý Layout
- Các thành phần GUI – Components
- Xử lý sự kiện

Java Foundation Classes (JFC)

- FC (Foundation Classes) là một nhóm các thư viện được thiết kế để hỗ trợ lập trình viên tạo ra các ứng dụng GUI trên Java.
- FC đơn giản hóa quá trình thiết kế và làm giảm thời gian thực hiện viết mã.
- Swing chỉ là một trong năm thư viện tạo nên JFC. JFC cũng chứa Abstract Window Toolkit (AWT),Accessibility API, 2D API và tăng cường hỗ trợ khả năng kéo thả (Drag and Drog).

Java Foundation Classes



Abstract Windowing Toolkit (AWT)

AWT chứa nhiều **class** và **method** cho phép thiết kế, quản lý cửa sổ và font chữ **trên giao diện đồ họa**.

Mục đích chính của awt là **hỗ trợ cho các ứng dụng applet** nhưng cũng được dùng để thiết kế các chương trình có **giao diện đồ họa độc lập**.

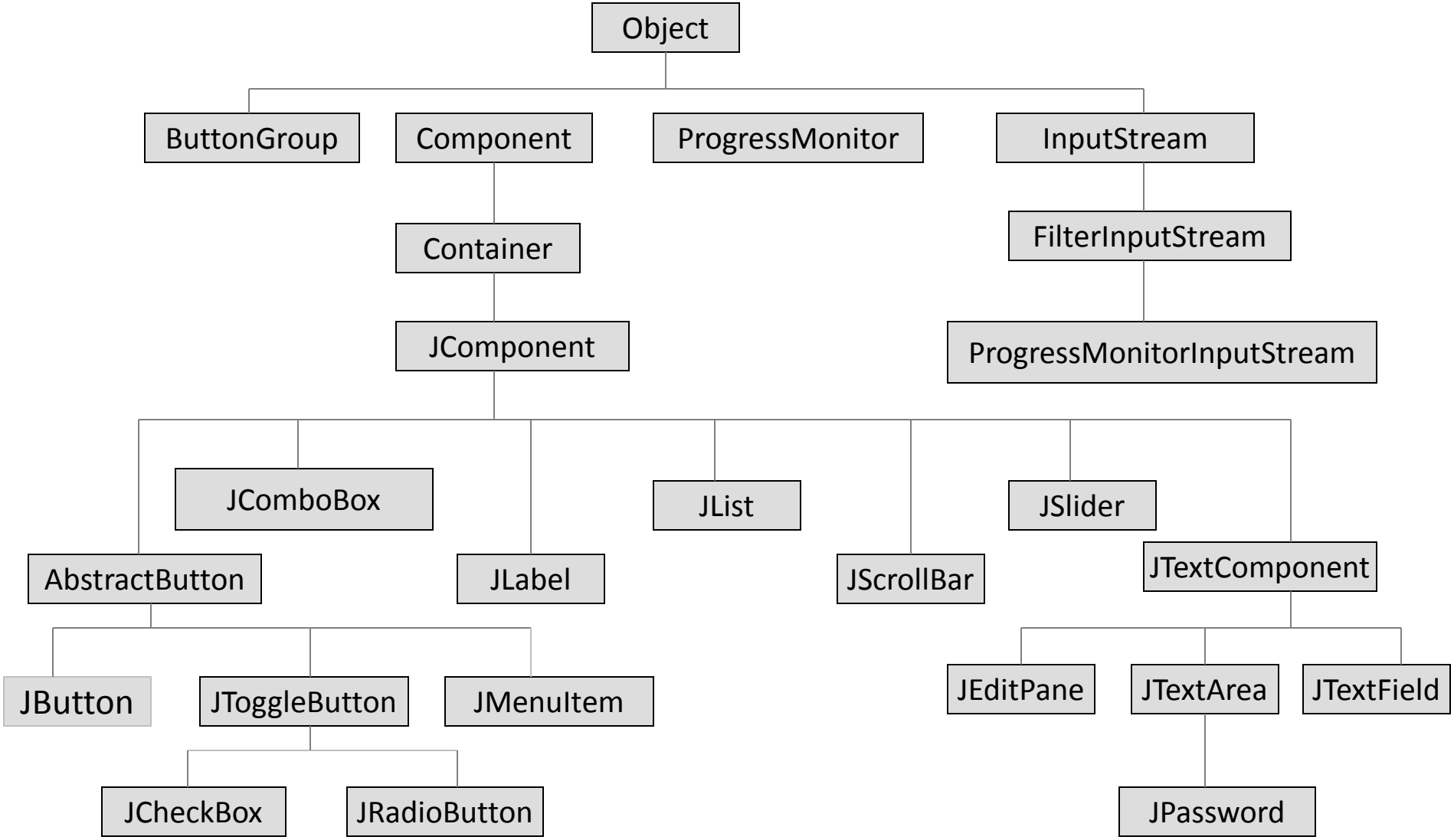
Tuy nhiên AWT có hạn chế:

- Các thành phần GUI có thể có hình dạng/hành động khác nhau trên các hệ điều hành khác nhau (heavyweight)
- Look and Feel của mỗi thành phần không thể (dễ dàng) thay đổi.

Swing

- Swing giải quyết các hạn chế liên quan đến các thành phần của AWT thông qua việc sử dụng 2 tính năng: các thành phần *lightweight* và *pluggable look and feel*.
- Các thành phần trong swing là **lightweight**: Các thành phần này được viết hoàn toàn bằng Java, do đó nó không phụ thuộc vào một hệ điều hành cụ thể nào và nó cũng rất hiệu quả và linh hoạt.
- Các class Swing có khả năng viết những cảm quan (**Look&Feels**) cho mỗi thành phần, và có thể thay đổi cảm quan vào thời điểm chạy.
- Swing có rất nhiều những thành phần mới như table, tree, slider, spinner, progress bar, internal frame và text.

Các thành phần của Swing

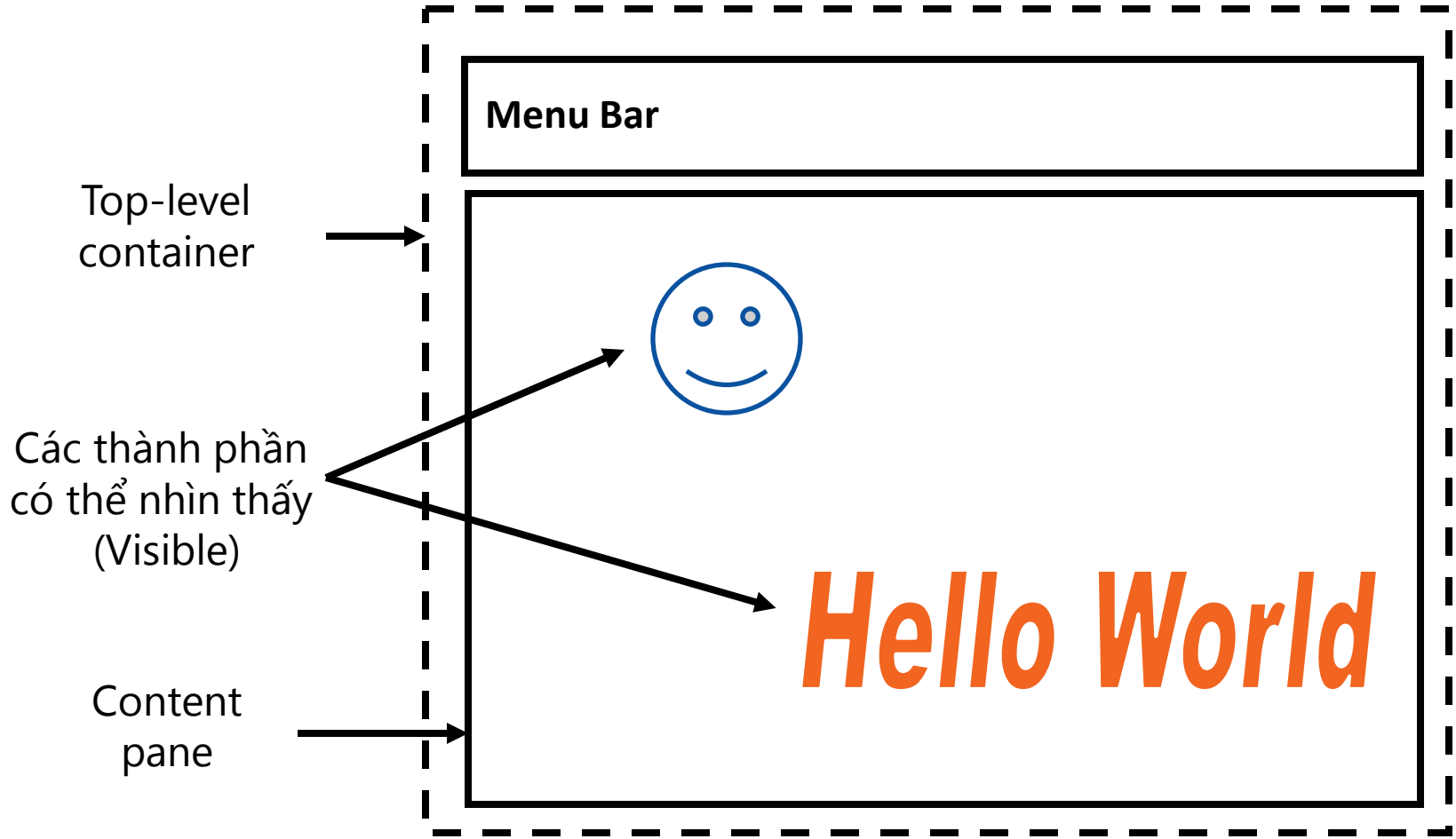


Containers

Một container là một thành phần đặc biệt có thể chứa các thành phần khác.

- **Top-level container** là một container cấp cao nhất, ở trên cùng của một hệ thống phân cấp.
- Swing cung cấp **4 container top-level container** là: JFrame, JDialog, JWindow và JApplet.
- Để hiển thị trên màn hình, mỗi một thành phần GUI phải là một phần của hệ thống phân cấp. Mỗi hệ thống phân cấp sẽ có một **top-level** là gốc. Trong đó **JFrame** hay được sử dụng nhất.

Top-Level Containers



JFrame

- JFrame Là top-level container.
- JFrame được dùng để tạo ra 1 cửa sổ.
  - `JFrame()`
  - `JFrame(String title)`
- Các thành phần khác sẽ được add vào cửa sổ JFrame đó:
  - `frame.getContentPane().add(b); //cũ`
  - `frame.add(b) //mới`

JFrame

Đóng JFrame

```
public int getDefaultCloseOperation()
public void setDefaultCloseOperation(int operation)
```

Các hằng số khi đóng 1 JFrame:

- DISPOSE_ON_CLOSE (value =1)
- DO_NOTHING_ON_CLOSE (value = 2)
- EXIT_ON_CLOSE (value = 3)
- HIDE_ON_CLOSE (value = 4) // mặc định

JFrame

Khi thêm các thành phần vào **JFrame** thì:

- Các thành phần đó được add vào **content pane** của Frame.
- Content pane sử dụng BorderLayout làm mặc định.
- Sử dụng **setSize**() hoặc **pack**() để đặt kích thước cửa sổ.
- Sử dụng **show**() hoặc **setVisible**() để hiển thị Frame.

JFrame

```
import javax.swing.*;
  class FrameTest {
   public static void main(String args[] {
     JFrame frame = new JFrame(“My Frame”);//tiêu đề
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     frame.setSize(300,200);// đặt kích thước cửa sổ
     frame.show();// hoặc setVisible(true)
  }
}
```

JPanel

- JPanel là container được dùng để nhóm các thành phần với nhau.
- JPanel sử dụng FlowLayout là mặc định.
- JPanel có các constructor:
  - `JPanel()`
  - `JPanel(LayoutManager mg)`

JPanel

```
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;

public class Main {
    public static void main(String args []) {
      JFrame frame = new JFrame("JFrame is me");
      JPanel jpan = new JPanel();
      frame.add( jpan );
      JButton btnHello = new JButton("Hello");
      jpan.add(btnHello);
      frame.setSize(300,100);
      frame.setVisible(true);
    }
}
```

JApplet

- JApplet là một container.
- javax.swing.JApplet khác với java.applet.Applet.
- Khi thêm các thành phần vào JApplet bắt buộc phải thêm vào content pane của JApplet.
  - Ví dụ: getContentPane().add(*component*);

JApplet

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

public class JAppletExample extends JApplet {
  @Override
  public void init() {
    Container content = getContentPane();
    content.setBackground(Color.white);
    content.setLayout(new FlowLayout());
    content.add(new JButton("Button 1"));
    content.add(new JButton("Button 2"));
    content.add(new JButton("Button 3"));
  }
}
```

Dialog: JOptionPane

JOptionPane cho phép bạn dễ dàng tạo ra các hộp thoại để nhập dữ liệu và hiển thị kết quả. Có rất nhiều lựa chọn khi sử dụng JOptionPane. Sau đây là cú pháp dạng đơn giản:

Giá trị	Phương thức
userInput =	JOptionPane.**showInputDialog**(component, text);
	JOptionPane.**showMessageDialog**(component, text);

Trong đó:

- *component là cửa sổ mà hộp thoại này sẽ đặt lên trên, nếu không có thì đặt là null*
- *text là xâu ký tự*

Dialog: JOptionPane

```
import javax.swing.JOptionPane;

public class JOptionPaneTest1 {
    public static void main(String[] args) {
        String ans;
        ans = JOptionPane.showInputDialog(null, "Tốc độ trên giờ?");
        double mph = Double.parseDouble(ans);
        double kph = 1.621 * mph;
        JOptionPane.showMessageDialog(null, "KPH = " + kph);
        System.exit(0);
    }
}
```

Dialog: JFileChooser

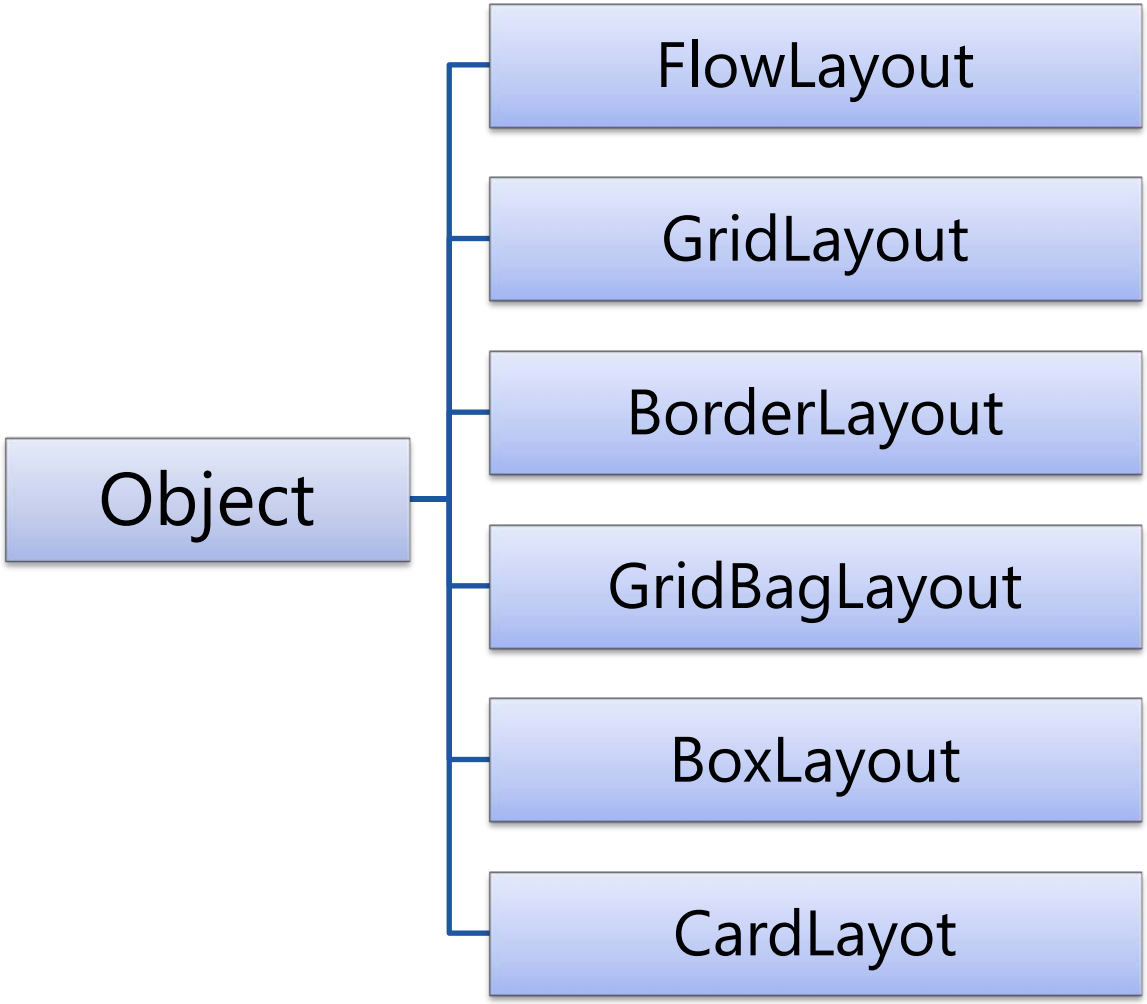
Sử dụng**JFileChooser** dùng để lựa chọn một tập tin hoặc thư mục trong ổ đĩa.

```
JFileChooser fileChooser = new JFileChooser();
int retval = fc.showOpenDialog(null);
if (retval == JFileChooser.APPROVE_OPTION) {
     File myFile = fc.getSelectedFile();
          //Các thao tác xử lý file
}
```

Cho phép chọn file/thư mục hoặc cả hai:

```
fc.setFileSelectionMode(JFileChooser.FILES_ONLY);//mặc định
fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
fc.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
```

Layout


Object
FlowLayout
GridLayout
BorderLayout
GridBagLayout
BoxLayout
CardLayot

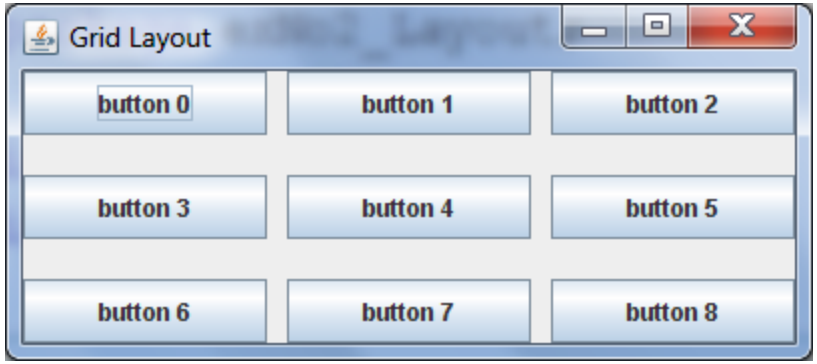
FlowLayout

- Là Layout đơn giản và là mặc định cho Panel.
- Các thành phần được sắp xếp theo hàng.
- Hướng của layout:
  - FlowLayout.LEFT
  - FlowLayout.CENTER
  - FlowLayout.RIGHT

- Khoảng cách giữa các thành phần:
  - Có giá trị mặc định = 5
  - Có thể đặt lại qua phương thức setHgap và setVgap.

GridLayout

- Là tập hợp các ô lưới có cùng kích thước
- Khoảng cách giữa các ô:
  - Mặc định = 5
  - Có thể đặt lại qua phương thức setHgap và setVgap.
- Thay đổi kích thước của các ô:
  - Các ô có thể có kích thước to hơn phụ thuộc vào kích thước của cửa sổ.



BorderLayout

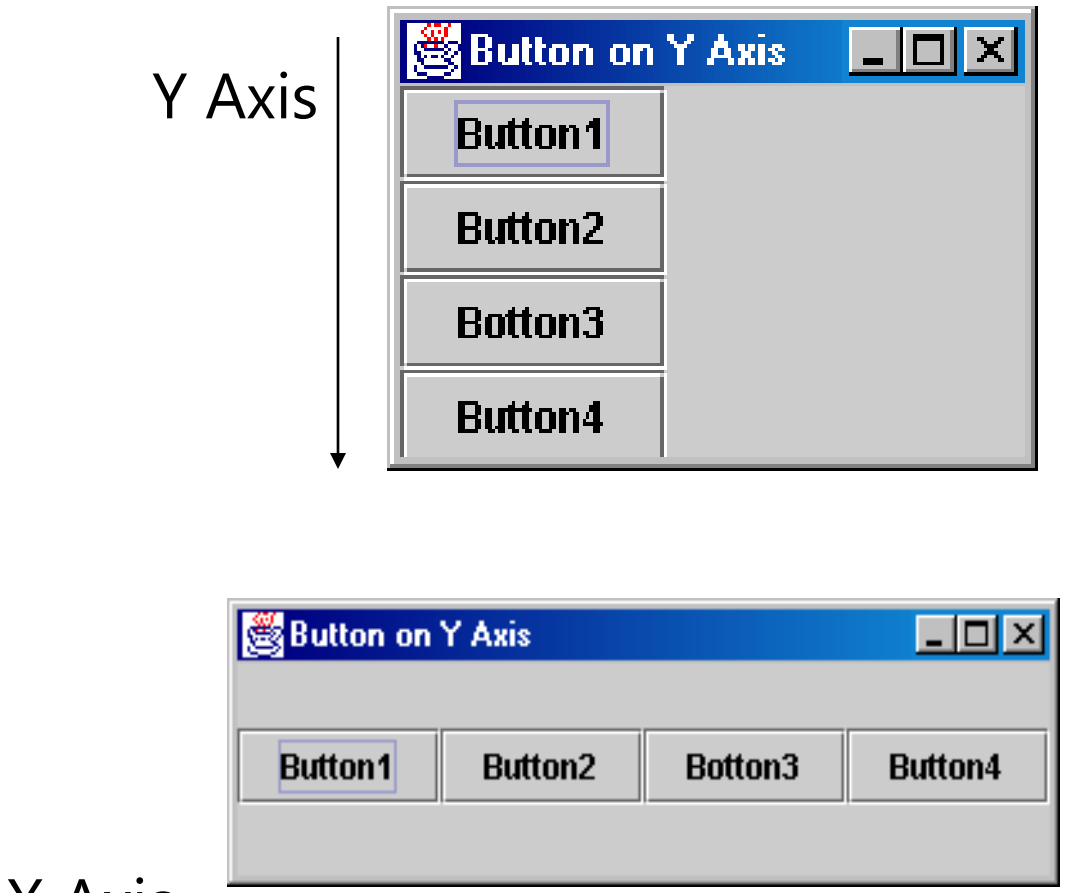
- Có 5 vùng:
  - NORTH, SOUTH, EAST, WEST, CENTER
  - Không phải tất cả 5 vùng đều được sử dụng
- Để đặt các thành phần vào một vị trí:
  - Panel.setLayout(new BorderLayout())
  - Panel.add(new JButton("Click here"), BorderLayout.NORTH)
- Đặt lại khoảng cách giữa các thành phần:
  - Có thể đặt lại qua phương thức setHgap và setVgap.
  - Sử dụng BorderLayout(int horGap, int verGap)

BoxLayout

Các thành phần được sắp xếp:

- X_AXIS: Theo chiều ngang từ trái qua phải
- Y_AXIS: Theo chiều dọc từ trên xuống dưới
- panel.setLayout(new BoxLayout(panel, BoxLayout.X_AXIS));
- panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

BoxLayout


Y Axis
Button on Y Axis
Button1
Button2
Botton3
Button4
Button on Y Axis
Button1
Button2
Botton3
Button4
X Axis

Components

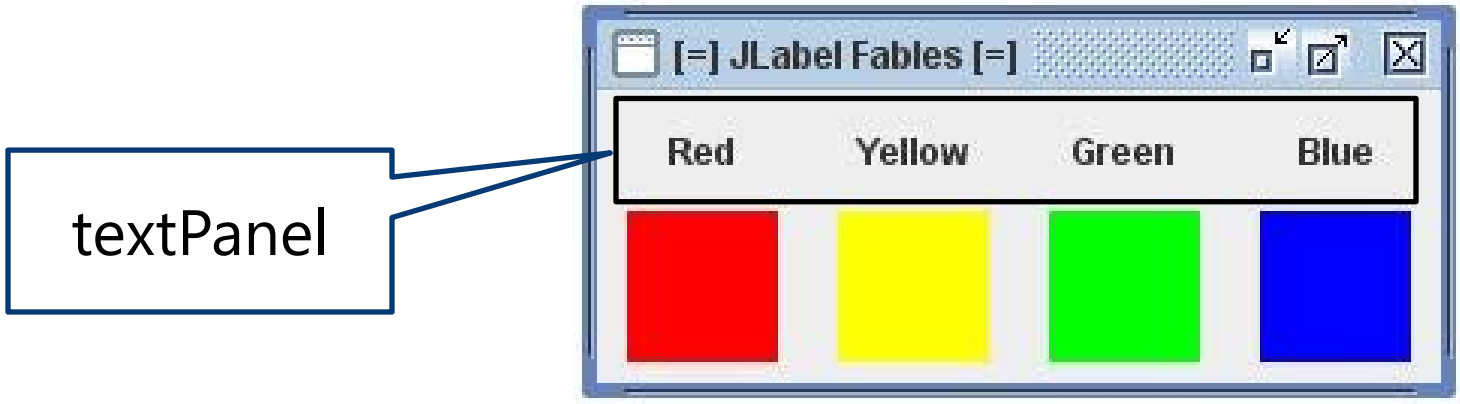
Một số component thường dùng:

JApplet**	JMenu	JSlider
JButton	JList	JSpinner
JCheckBox	JMenuBar	JTable
JColorChooser	JMenuItem	JTextArea
JComboBox	JPopupMenu	JTextField
JDialog**	JPanel	JToggleButton
JFileChooser	JProgressBar	JToolBa
JFormattedTextField	PasswordField	JTree
JFrame**	JRadioButton	JWindow **
JLabel	ScrollBar	

**: Là top-level containers.

Component: JLabel

```
JPanel textPanel = new JPanel();
JLabel redLabel = new JLabel("Red");
redLabel.setLocation(0, 0);
redLabel.setSize(50, 40);
textPanel.add(redLabel);
```



Component: JButton

Khởi tạo :

```
String text;
Icon image;
JButton btn = new JButton(text)
JButton btn = new JButton(text,image)
JButton btn = new JButton(image)
```

Phương thức:

```
ActionListener listener;
boolean b;
btn.addActionListener(listener);
btn.setEnable(b);
```

Component: JButton

Xử lý sự kiện với JButton :

Khi một button được click thì phương thức `actionPerformed()`được thực hiện với một `ActionEvent` được truyền vào.

Ví dụ:

```
JButton btn = new JButton(“Do Something”);
btn.addActionListener(new ActionListener()  {
  public void actionPerformed(ActionEvent e){
       doMyAction();// khi click vào button
  }
 }
);
```

Component: JTextField

Các phương thức:

```
JTextField txt = new JTextField(20) ;
txt.setText("Enter here");
Hoặc JTextField txt= new JTextField("Enter here", 20);
String str=txt.getText();

txt.addActionListener(listener);
txt.setEditable(true/false);
txt.setFont(font);
txt.setHorizontalAlignment(align);
txt.requestFocus();//Đặt con trỏ vào textbox
```

Component: JTextField

```
public class textFieldUpperCase extends JFrame {
    public textFieldUpperCase() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        final JTextField text = new JTextField(40);
        text.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e){
                text.setText(text.getText().toUpperCase());
            }
        });
        getContentPane().add(text, BorderLayout.NORTH);
        pack();  setVisible(true);
    }
    public static void main(String[] args) {
        new textFieldUpperCase();
    }
}
```

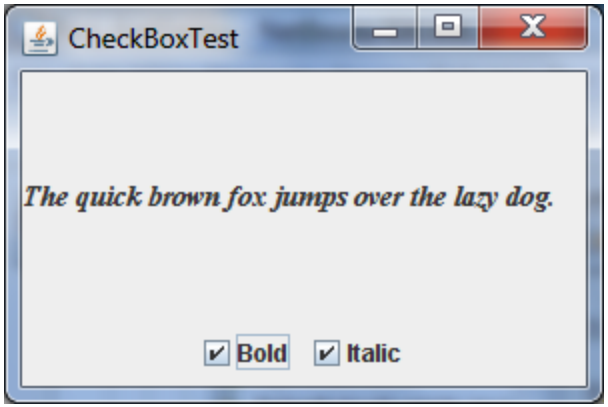
Component: JCheckBox

```
JCheckBox cb;    //một checkbox
String text;     //một giá trị text
boolean state;   //Giá trị true hoặc false

cb = new JCheckBox(text); hoặc
cb = new JCheckBox(text,state);

state = cb.isSelected();
cb.setSelected(state);
cb.addActionListener(listener);
cb.addItemListener(itemListener);
```

Component: JCheckBox

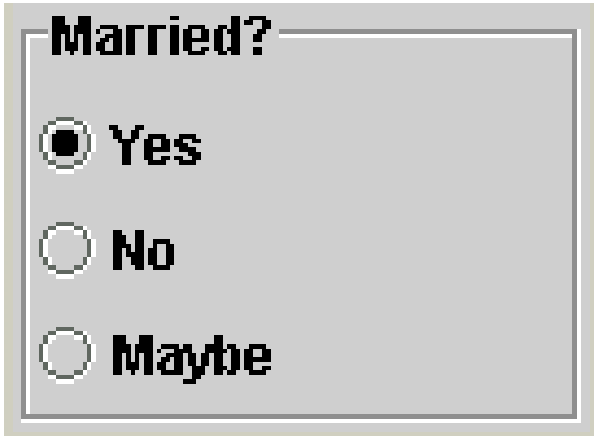
Xử lý sự kiện khi click vào Checkbox:

```
ActionListener listener = new ActionListener(){
        public void actionPerformed(ActionEvent event)
         {
            int mode = 0;
            if (bold.isSelected()) mode += Font.BOLD;
            if (italic.isSelected()) mode += Font.ITALIC;
            label.setFont(new Font("Serif", mode, FONTSIZE));

         }
     };
  bold.addActionListener(listener);
  italic.addActionListener(listener);
```



Component: JRadio

```
JRadioButton yesButton   = new JRadioButton("Yes" , true);
JRadioButton noButton    = new JRadioButton("No" , false);
JRadioButton maybeButton = new JRadioButton("Maybe", false);

ButtonGroup bgroup = new ButtonGroup();
bgroup.add(yesButton);
bgroup.add(noButton);
bgroup.add(maybeButton);

JPanel radioPanel = new JPanel();
radioPanel.setLayout(new GridLayout(3, 1));
radioPanel.add(yesButton);
radioPanel.add(noButton);
radioPanel.add(maybeButton);

radioPanel.setBorder(BorderFactory.createTitledBorder(
        BorderFactory.createEtchedBorder(), "Married?"));
```



Component: JRadio

```
boolean b;
JRadioButton rb = new JRadioButton("Sample", false);
```

Một số phương thức hay dùng:

rb.**isSelected**();	Trả về true nếu nút radio được chọn.
rb.**setSelected**(b);	Đặt nút radio là b (true/false).
rb.**addActionListener**(listener);	Thêm vào hành động listener cho nút radio. Nếu radio này được chọn thì listener sẽ hoạt động.
rb.**addItemListener**(itemlistener);	Thêm vào hành động listener cho nút radio. Nếu radio này được chọn hoặc bỏ chọn thì listener sẽ hoạt động.

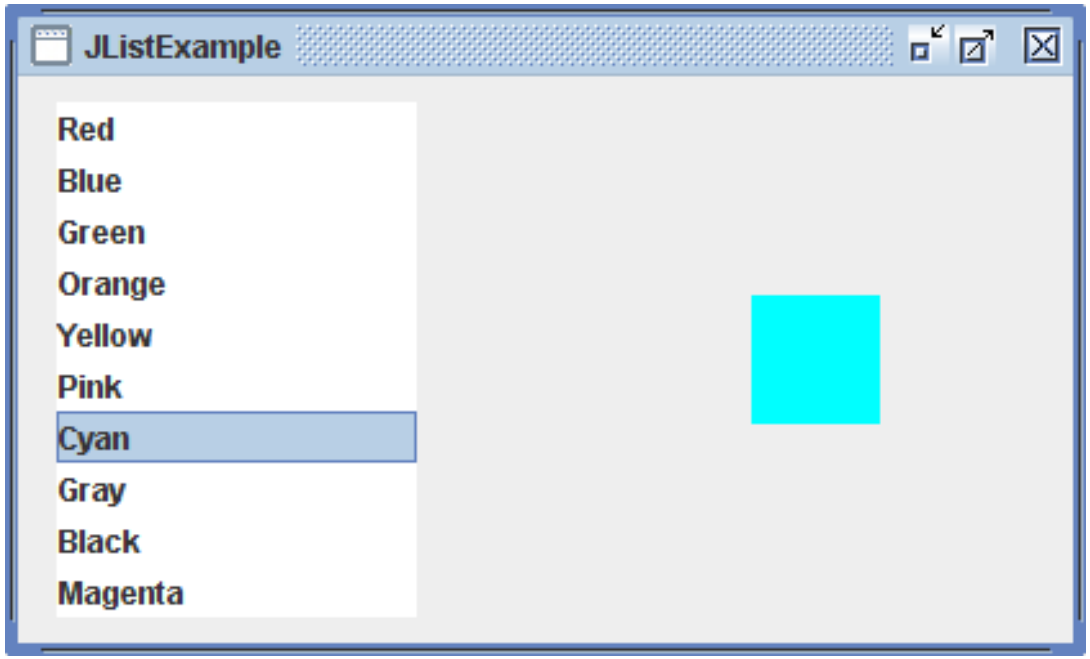
Component: JList

Cú pháp:

- `JList()`
- `JList(ListModel dataModel)`
- `Jlist(Object[] listData)`

Component: JList

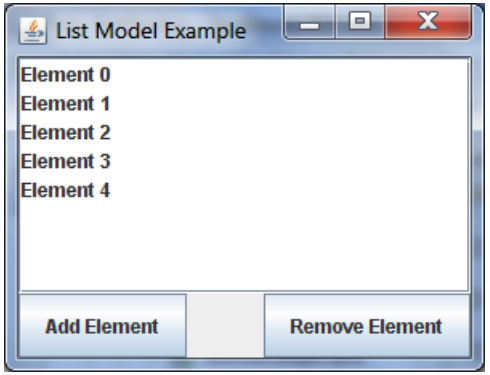
Ví dụ:




```
JList boxes;
String names[] = {"Red", "Blue", "Green", "Orange", "Yellow",
                  Pink", "Cyan", "Gray", "Black", "Magenta"};
boxes = new JList(names);
boxes.setVisibleRowCount(10);
```

Component: JList

```
JList list;
DefaultListModel model;
int count = 5;
model = new DefaultListModel();
list = new JList(model);
JButton addButton = new JButton("Add Element");
for (int i = 0; i < 10; i++)
    model.addElement("Element " + i);


addButton.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent e){
       model.addElement("Element " + counter);
       counter++;
      }
});
```



Component: JComboBox

Khởi tạo:

```
String[] dias = {"lunes", "martes", "miercoles"};
JComboBox dayChoice = new JComboBox(dias)
```

Giả sử có:

```
int index;
String item;
String obj; // obj có thể kiểu khác
```

Các phương thức:

```
String JComboBox cb = new JComboBox(...);
cb.addActionListener(listener);
```

Component: JComboBox

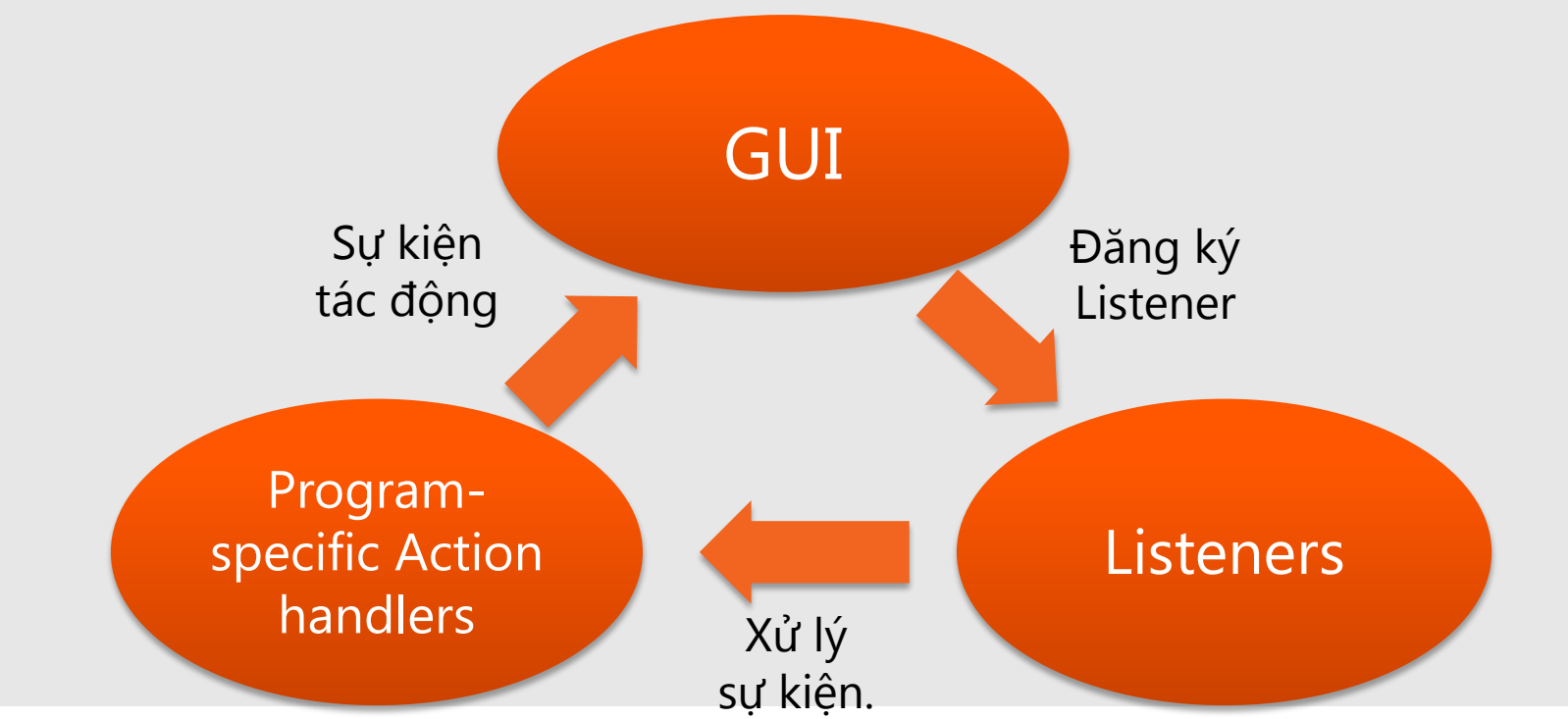
Phương thức	Ý nghĩa
`cb.setSelectedIndex(index)`	Đặt mục có chỉ số là index được chọn.
`cb.setSelectedItem(obj)`	Đặt mục obj được chọn.
`cb.add(item)`	Thêm vào một item.
`cb.insert(item,index)`	Thêm vào item sau vị trí index.
`cb.remove(item)`	Xóa mục item khỏi combo box.
`cb.remove(index)`	Xóa mục có chỉ số index ra khỏi combo box.

index = cb.**getSelectedIndex**(); ➔ Trả về chỉ số của mục được chọn.
obj = cb.**getSelectedItem**(); ➔ Trả về giá trị của mục đang được chọn.

Xử lý sự kiện

- Một sự kiện thường xảy ra khi có sự thay đổi trong giao diện người dùng đồ họa. Ví dụ như khi người dùng click chuột vào một button, click vào một mục trong combo box… và như vậy một sự kiện sẽ được kích hoạt.
- Các thành phần đồ họa (components) tạo ra các sự kiện này được gọi là ‘nguồn sự kiện’ (event source). Các nguồn sự kiện sẽ gửi đi một đối tượng sự kiện (event object).
- Các sự kiện được xử lý bởi một sự kiện lắng nghe (event listener) được gán cho nguồn sự kiện. Các thành phần khác nhau của GUI sẽ có các event listener khác nhau.

Xử lý sự kiện



- Xây dựng GUI và kết nối (còn gọi là đăng ký) tới các listeners.
- Listener sẽ thực thi (implements) các interface thích hợp với từng loại sự kiện.
- Thực hiện các lệnh (trong xử lý sự kiện) phù hợp với nguồn sự kiện.

Xử lý sự kiện

Event Object	Interface và method	Các method liên kết	Event Source
ActionEvent	interface **ActionListener** actionPerformed(ActionEvent)	addActionListener removeActionListener	JButton JCheckBox JComboBox JTextField JRadioButton
ItemEvent	interface **ItemListener** itemStateChanged(ItemEvent)	addItemListener removeItemListener	JCheckBox JRadioButton JComboBox
MouseEvent	interface **MouseListener** mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)	addMouseListener removeMouseListener	*Được tạo bởi sự kiện Mouse của bất kỳ thành phần GUI nào.*

Xử lý sự kiện

- Sử dụng Inner Class Listener có tên: Là cách thường dùng để viết các chương trình nhỏ. Nhiều component có thể dùng chung Inner Class này.
- Anonymous Inner Class Listeners: Là inner class không đặt tên, thường được dùng cho một component. Class dạng này không được linh hoạt.
- Top-level Listeners (this): Thường được sử dụng khi có một event source.

Xử lý sự kiện

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

class SomePanel extends JPanel {
private JButton myGreetingButton = new JButton("Hello");
private JTextField myGreetingField = new JTextField(20);
public SomePanel() {
      ActionListener doGreeting = new GreetingListener();
      myGreetingButton.addActionListener(doGreeting);
      myGreetingField.addActionListener(doGreeting);
      // Các lệnh khác
}
// Inner class tên là GreetingListener
private class GreetingListener implements ActionListener {
   public void actionPerformed(ActionEvent e) {
      myGreetingField.setText("Guten Tag");
      }
    }
}
```

Xử lý sự kiện

```
class myPanel extends JPanel {
. . .
   public MyPanel() {
        JButton b1 = new JButton("Hello");

        b1.addActionListener(
        new ActionListener(){
          public void actionPerformed(ActionEvent e){
             // Các lệnh xử lý sự kiện cho b1
          }
        }
       );
...
}
```

Xử lý sự kiện

```
...
class Hello extends JApplet implements ActionListener{
	JButton b;
	...
	public void init() {
		JButton b = new JButton("Hello");
		b.addActionListener(this);
		...
	}
	public void actionPerformed(ActionEvent e) {
		message = "Hi";
		repaint();
	}
	...
}
```

Tổng kết bài học

- Java Foundation Class (AFC)
- Abstract Windowing Toolkit (AWT)
- Swing
- Các thùng chứa -Containers
- Các hộp thoại - Dialogs
- Quản lý Layout
- Các thành phần GUI – Components
- Xử lý sự kiện