VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

UNIVERSITY OF INFORMATION TECHNOLOGY

Chapter 4

GUI

Lecturer: MSc. Kiet Van Nguyen

Faculty of Information Science and Engineering

University of Information Technology, VNU-HCM

Today's Lecture

- Introduction to Java's GUI
- AWT GUI
- Swing GUI

Introduction to Java's GUI

- Three sets of Java APIs for graphics programming: AWT (Abstract Windowing Toolkit), Swing and JavaFX.
  - AWT API was introduced in JDK 1.0.
  - Swing API, a much more comprehensive set of graphics libraries that enhances the AWT, introduced in JDK 1.1.
  - The latest JavaFX, which was integrated into JDK 8, is meant to replace Swing.

AWT

- Consists of 12 packages of 370 classes (Swing is even bigger, with 18 packages of 737 classes as of JDK 8).
- Only 2 packages - java.awt and java.awt.event - are commonly-used.

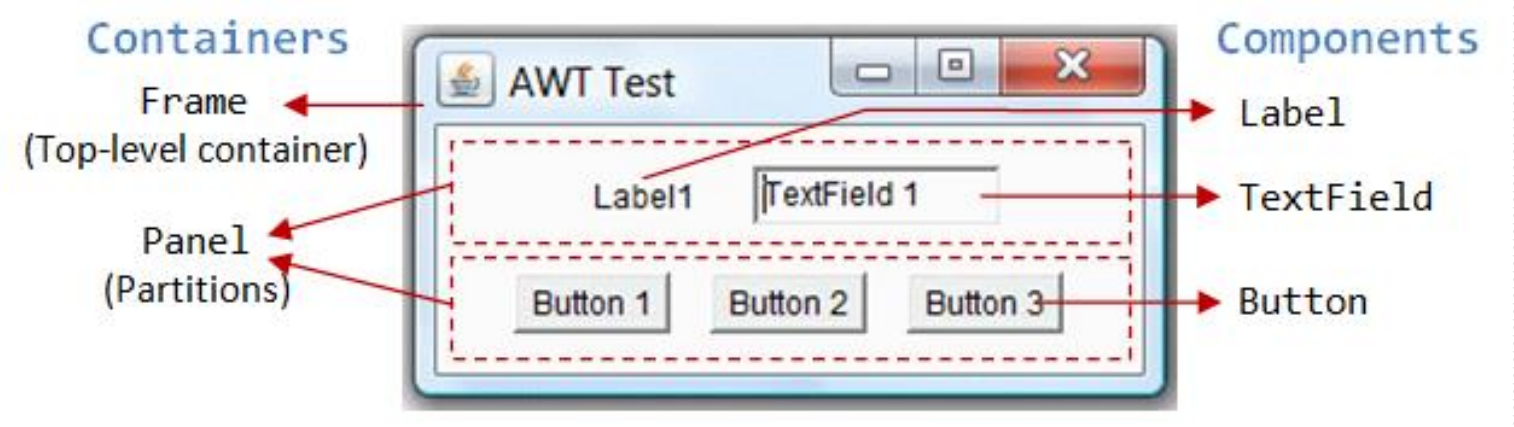
AWT (Cont.)

- The java.awt package contains the *core* AWT graphics classes:
  - GUI Component classes, such as Button, TextField, and Label.
  - GUI Container classes, such as Frame and Panel.
  - Layout managers, such as FlowLayout, BorderLayout and GridLayout.
  - Custom graphics classes, such as Graphics, Color and Font.
- The java.awt.event package supports event handling:
  - Event classes, such as ActionEvent, MouseEvent, KeyEvent and WindowEvent,
  - Event Listener Interfaces, such as ActionListener, MouseListener, MouseMotionListener, KeyListener and WindowListener,
  - Event Listener Adapter classes, such as MouseAdapter, KeyAdapter, and WindowAdapter.

Containers and Components

- Two types of GUI elements:
  - ***Component***: Components are elementary GUI entities, such as Button, Label, and TextField.
  - ***Container***: Containers, such as Frame and Panel, are used to *hold components in a specific layout* (such as FlowLayout or GridLayout). A container can also hold sub-containers.

Containers and Components (Cont.)



Containers and Components (Cont.)

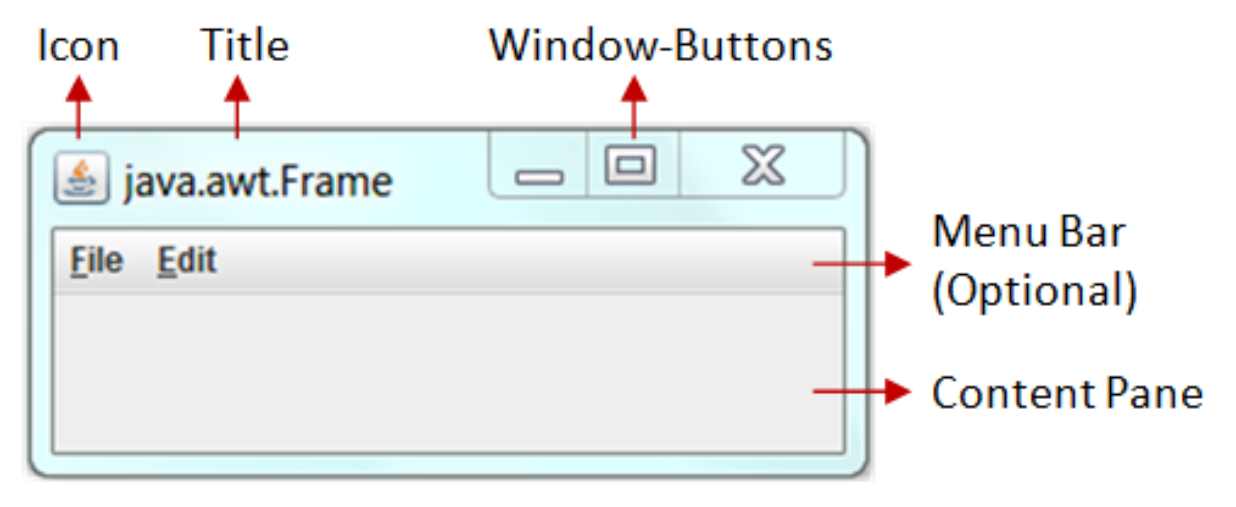
```
Panel pnl = new Panel();          // Panel is a container
Button btn = new Button("Press"); // Button is a component
pnl.add(btn);                     // The Panel container adds a Button component
```

- GUI components are also called *controls* (e.g., Microsoft ActiveX Control), *widgets* (e.g., Eclipse's Standard Widget Toolkit, Google Web Toolkit).

AWT Container Classes

- **Top-Level Containers:** Frame, Dialog and Applet
- **Secondary Containers:** Panel and ScrollPane

Example: Top-Level Containers



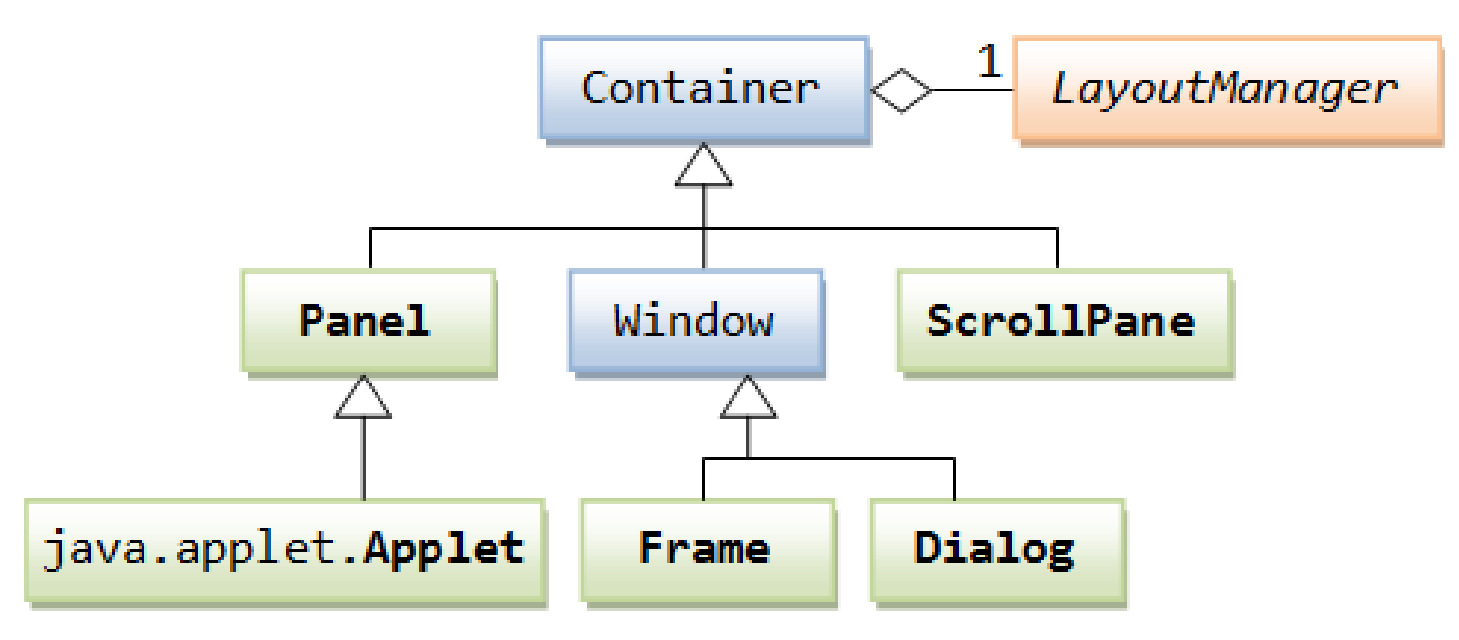
Code: Top-Level Containers

```
import java.awt.Frame;  // Using Frame class in package java.awt

// A GUI program is written as a subclass of Frame - the top-level container
// This subclass inherits all properties from Frame, e.g., title, icon, buttons, content-pane
public class MyGUIProgram extends Frame {

   // private variables
   ......

   // Constructor to setup the GUI components
   public MyGUIProgram() { ...... }

   // methods
   ......
   ......

   // The entry main() method
   public static void main(String[] args) {
      // Invoke the constructor (to setup the GUI) by allocating an instance
      new MyGUIProgram();
   }
}
```

Secondary Containers

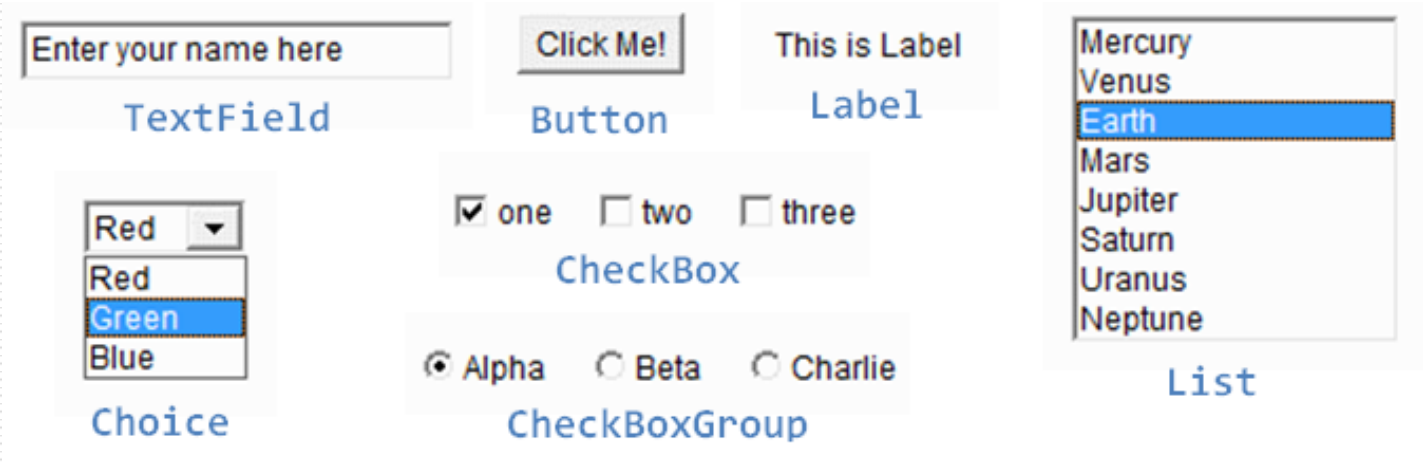
- Secondary containers are placed inside a top-level container or another secondary container. AWT provides these secondary containers:
  - ✓ **Panel**: a rectangular box used to *layout* a set of related GUI components in pattern such as grid or flow.
  - ✓ **ScrollPane**: provides automatic horizontal and/or vertical scrolling for a single child component.
  - ✓ others.

Hierarchy of the AWT Container Classes



AWT Component Classes

- Providing many ready-made and reusable GUI components in package java.awt.
- Button, TextField, Label, Checkbox, CheckboxGroup (radio buttons), List, and Choice



AWT GUI Component: java.awt.Label

- *java.awt.Label* provides a descriptive text string.
- Constructors:

```
public Label(String strLabel, int alignment); // Construct a Label with the given text String, of the text alignment
public Label(String strLabel);                // Construct a Label with the given text String
public Label();                               // Construct an initially empty Label
```

- Constants (final static fields)

```
public static final LEFT;    // Label.LEFT
public static final RIGHT;   // Label.RIGHT
public static final CENTER;  // Label.CENTER
```

- Public Methods:

```
// Examples
public String getText();
public void setText(String strLabel);
public int getAlignment();
public void setAlignment(int alignment);  // Label.LEFT, Label.RIGHT, Label.CENTER
```

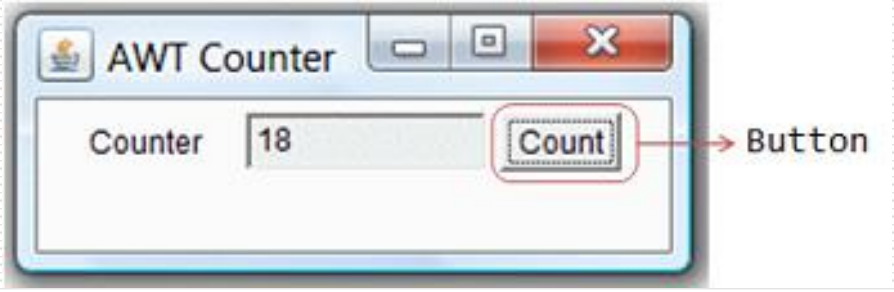
AWT GUI Component: java.awt.Label

- **For example,**

```
Label lblInput;                      // Declare an Label instance called lblInput
lblInput = new Label("Enter ID");    // Construct by invoking a constructor via the new operator
add(lblInput);                       // this.add(lblInput) - "this" is typically a subclass of Frame
lblInput.setText("Enter password"); // Modify the Label's text string
lblInput.getText();                  // Retrieve the Label's text string
```

AWT GUI Component: java.awt.Button

- A java.awt.Button is a GUI component that triggers a certain programmed *action* upon clicking.



AWT GUI Component: java.awt.Button

- Constructors

```
public Button(String btnLabel);
   // Construct a Button with the given label
public Button();
   // Construct a Button with empty label
```

- Public Methods

```
public String getLabel();
   // Get the label of this Button instance
public void setLabel(String btnLabel);
   // Set the label of this Button instance
public void setEnable(boolean enable);
   // Enable or disable this Button. Disabled Button cannot be clicked.
```

- Event: Clicking a button fires a so-called ActionEvent.

AWT GUI Component: java.awt.Button

- **Example,**

```
Button btnColor = new Button("Red"); // Declare and allocate a Button instance called btnColor
add(btnColor);                       // "this" Container adds the Button
...
btnColor.setLabel("Green");          // Change the button's label
btnColor.getLabel();                 // Read the button's label
...
add(Button("Blue"));    // Create an anonymous Button. It CANNOT be referenced later
```

AWT GUI Component: java.awt.TextField

- A java.awt.TextField is single-line text box for users to enter texts.
- There is a multiple-line text box called TextArea.

AWT GUI Component: java.awt.TextField

- Constructors

```
public TextField(String initialText, int columns);
   // Construct a TextField instance with the given initial text string with the number of columns.
public TextField(String initialText);
   // Construct a TextField instance with the given initial text string.
public TextField(int columns);
   // Construct a TextField instance with the number of columns.
```

- Public Methods

```
public String getText();
   // Get the current text on this TextField instance
public void setText(String strText);
   // Set the display text on this TextField instance
public void setEditable(boolean editable);
   // Set this TextField to editable (read/write) or non-editable (read-only)
```

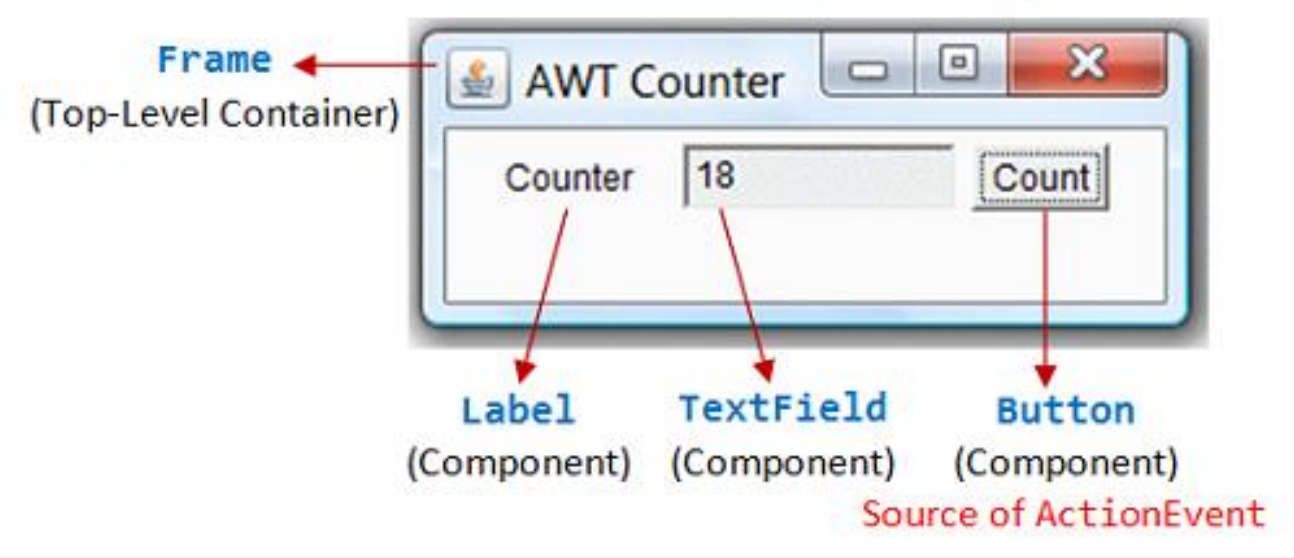
- Event: Hitting the "ENTER" key on a TextField fires a ActionEvent, and triggers a certain programmed action.

AWT GUI Component: java.awt.TextField

- **Example,**

```
TextField tfInput = new TextField(30); // Declare and allocate an TextField instance called tfInput
add(tfInput);                          // "this" Container adds the TextField
TextField tfResult = new TextField();  // Declare and allocate an TextField instance called tfResult
tfResult.setEditable(false) ;          // Set to read-only
add(tfResult);                         // "this" Container adds the TextField
......
// Read an int from TextField "tfInput", square it, and display on "tfResult".
// getText() returns a String, need to convert to int
int number = Integer.parseInt(tfInput.getText());
number *= number;
// setText() requires a String, need to convert the int number to String.
tfResult.setText(number + "");
```

Example : AWTCounter



Example: AWTCounter

```
1  import java.awt.*;        // Using AWT container and component classes
2  import java.awt.event.*;  // Using AWT event classes and listener interfaces
3
4  // An AWT program inherits from the top-level container java.awt.Frame
5  public class AWTCounter extends Frame implements ActionListener {
6     private Label lblCount;    // Declare a Label component
7     private TextField tfCount; // Declare a TextField component
8     private Button btnCount;   // Declare a Button component
9     private int count = 0;     // Counter's value
10
11    // Constructor to setup GUI components and event handlers
12    public AWTCounter () {
13       setLayout(new FlowLayout());
14          // "super" Frame, which is a Container, sets its layout to FlowLayout to arrange
15          // the components from left-to-right, and flow to next row from top-to-bottom.
16
17       lblCount = new Label("Counter");  // construct the Label component
18       add(lblCount);                    // "super" Frame container adds Label component
19
20       tfCount = new TextField(count + "", 10); // construct the TextField component with initial text
21       tfCount.setEditable(false);       // set to read-only
22       add(tfCount);                     // "super" Frame container adds TextField component
23
24       btnCount = new Button("Count");   // construct the Button component
25       add(btnCount);                    // "super" Frame container adds Button component
26
27       btnCount.addActionListener(this);
```

Example: AWTCounter (Cont.)

```
33          setTitle("AWT Counter");  // "super" Frame sets its title
34          setSize(250, 100);        // "super" Frame sets its initial window size
35
36          // For inspecting the Container/Components objects
37          // System.out.println(this);
38          // System.out.println(lblCount);
39          // System.out.println(tfCount);
40          // System.out.println(btnCount);
41
42          setVisible(true);         // "super" Frame shows
43
44          // System.out.println(this);
45          // System.out.println(lblCount);
46          // System.out.println(tfCount);
47          // System.out.println(btnCount);
48       }
49
50       // The entry main() method
51       public static void main(String[] args) {
52          // Invoke the constructor to setup the GUI, by allocating an instance
53          AWTCounter app = new AWTCounter();
54             // or simply "new AWTCounter();" for an anonymous instance
55       }
56
57       // ActionEvent handler - Called back upon button-click.
58       @Override
59       public void actionPerformed(ActionEvent evt) {
60          ++count; // Increase the counter value
61          // Display the counter value on the TextField tfCount
62          tfCount.setText(count + ""); // Convert int to String
63       }
64    }
```

AWT Event-Handling

- **Event-Driven** programming model for event-handling.
- **A piece of event-handling codes is executed** when an event was fired in response to an user input. For example, clicking a mouse button or hitting the ENTER key in a text field.

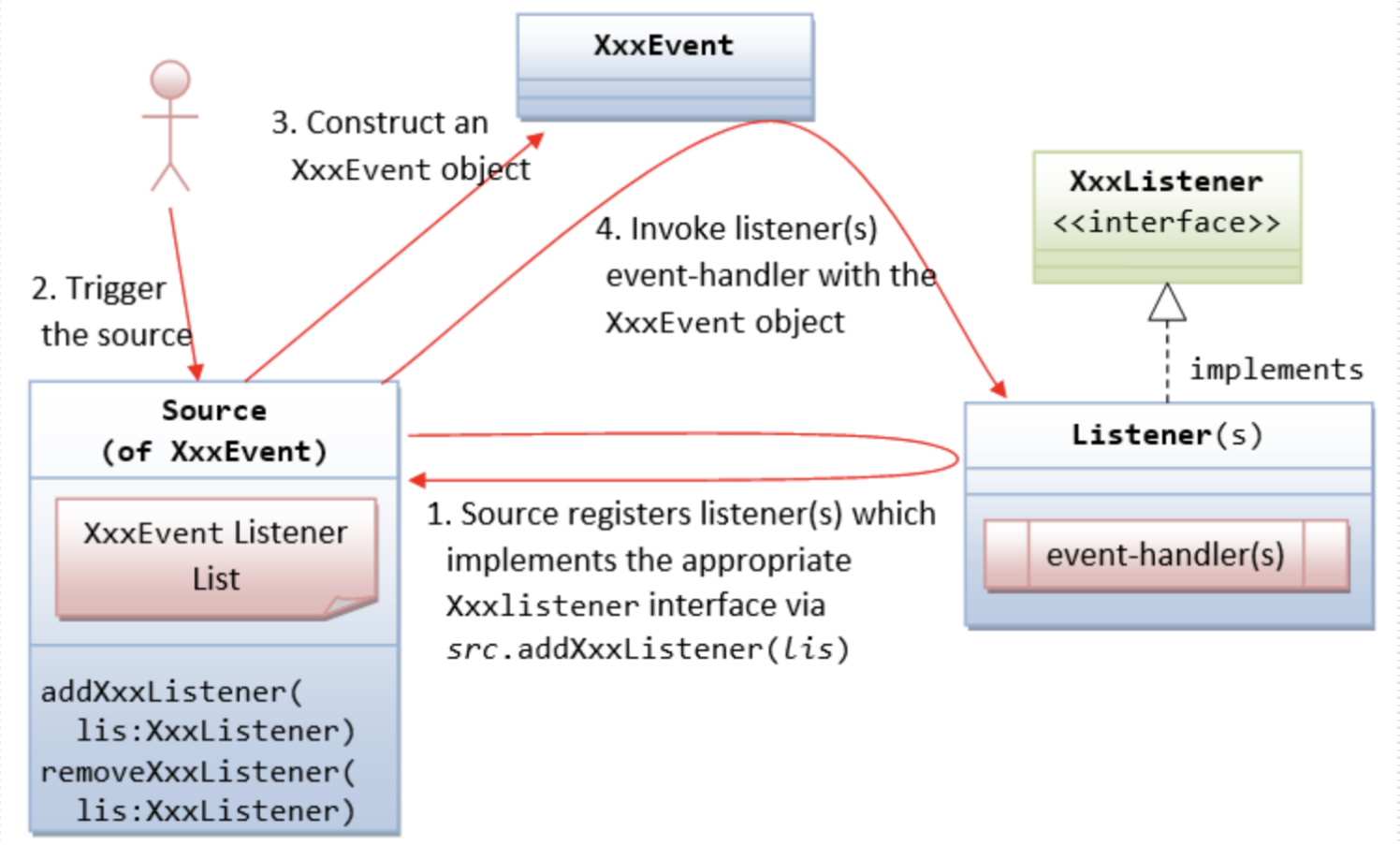
AWT Event-Handling

❖ Source, Event and Listener Objects

- package java.awt.event.
- Three objects: *source*, *listener* and an *event* .
- S***ource*** (Button and TextField): interact with the user.
- ***Event*** will be messaged to all the *registered* ***listener*** object(s), and an appropriate event-handler method of the listener(s) is called-back to provide the response.

Triggering ***a source*** *fires* ***an event*** *to all its* ***listener****(s), and invoke an appropriate event handler of the listener(s).*

AWT Event-Handling



AWT Event-Handling

Step 1: The source object registers its listener(s) for a certain type of event.

Step 2: The source is triggered by a user.

Step 3: The source create a XxxEvent object, which encapsulates the necessary information about the activation. For example, the (x, y) position of the mouse pointer, the text entered, etc.

Step 4: Finally, for each of the XxxEvent listeners in the listener list, the source invokes the appropriate handler on the listener(s), which provides the programmed response.

Event và Listener

Source	Event	Listener
Window, Frame, …	WindowEvent	WindowListener
Button, MenuItem, …	ActionEvent	ActionListener
TextComponent, …	TextEvent	TextListener
List, …	ActionEvent	ActionListener
…	ItemEvent	ItemListener
	ComponentEvent	ComponentListener
	MouseEvent	MouseListener
		MouseMotionListener
	KeyEvent	KeyListener

Layout Manager

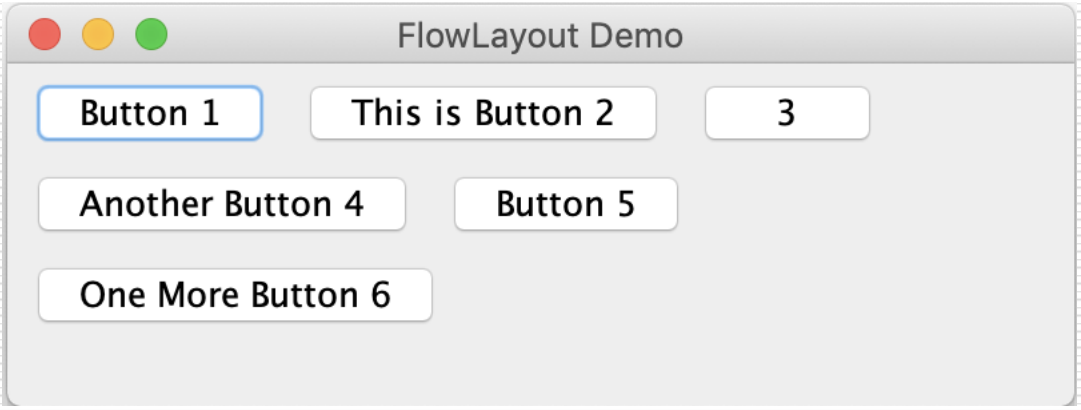
- **Various types of layout:**
  - Flow Layout
  - Border Layout
  - Grid Layout
  - GridBag Layout
  - Null Layout
  - …
- *setLayout( ): set layout.*

FlowLayout

- Default layout of Applet và Panel.
- Các thành phần được sắp xếp từ góc trái trên → góc phải dưới của màn hình.
- Các constructor:

```
FlowLayout layout = new FlowLayout();
FlowLayout layout = new
FlowLayout(FlowLayout.RIGHT);
// Căn lề bên phải
```

FlowLayout (tiếp theo)

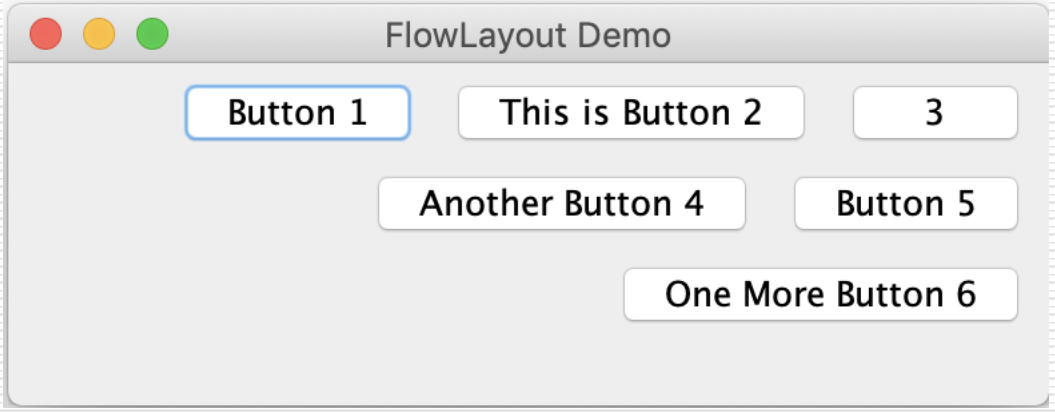


Căn trái

(Right Alignment)

Căn phải

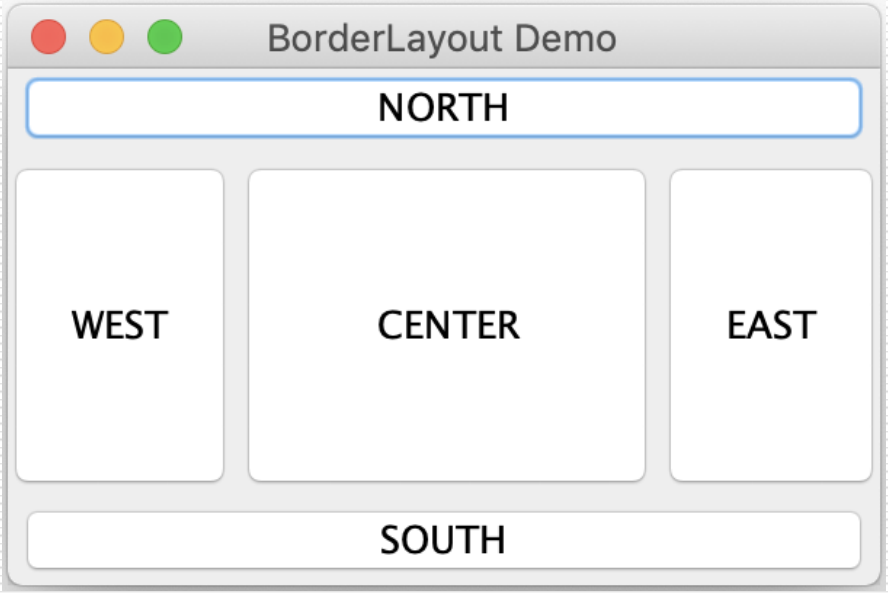
(Right Alignment)



BorderLayout

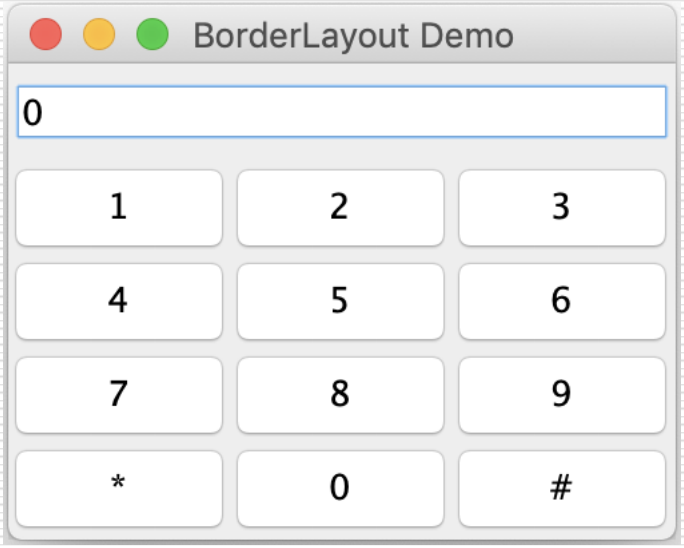
- Là trình quản lý layout mặc định cho Window, Frame và Dialog.
- Trình quản lý này có thể sắp xếp đến 5 thành phần theo 5 hướng NORTH, EAST, SOUTH, WEST và CENTER của container.
- Ví dụ: Để thêm một thành phần vào vùng North của container
  - `Button b1= new Button("North Button");`
  - `setLayout(new BorderLayout( ));`
  - `add(b1, BorderLayout.NORTH);`

BorderLayout (tiếp theo)



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




```
GridLayout layout = new GridLayout(no. of rows, no. of columns);
containerObj.setLayout(layout);
```

Các Layout khác

- BoxLayout
- GridBagLayout
- NullLayout

→ Sinh viên tìm hiểu thêm.

Exercise

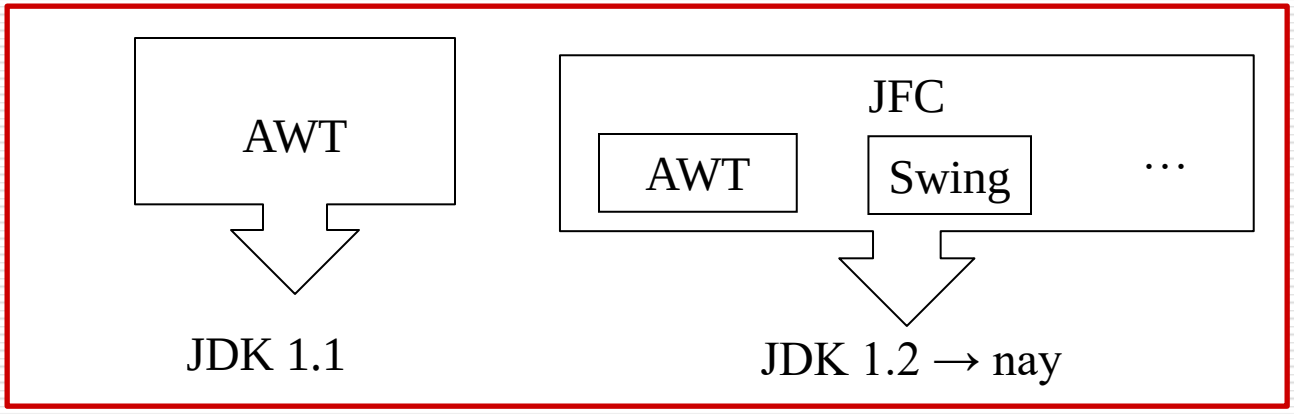
Ex 1: So sánh lập trình giao diện (GUI): Java và C#.

Ex 2: Thiết kế và xây dựng một máy tính có khả năng thực hiện các phép toán cơ bản.

Swing

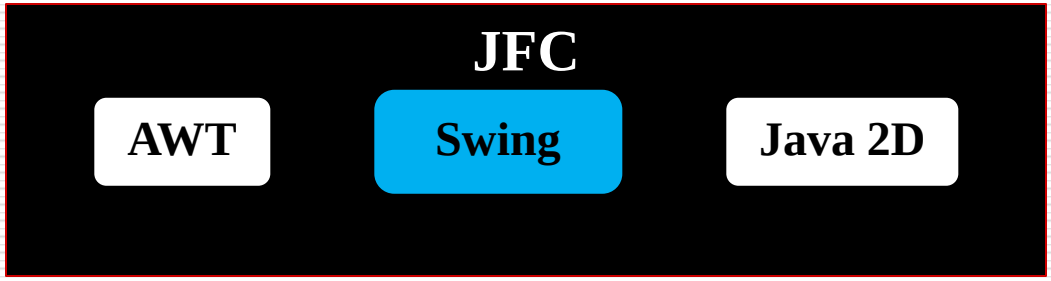
Giới thiệu

- **AWT**
  - Các phiên bản trước JDK 1.2.
- **Swing**
  - Từ phiên bản JDK 1.2.
  - Nhiều chức năng hơn AWT.



Java Swing

- Là một thành phần của **Java Foundation Classes** (JFC, 4/1997).

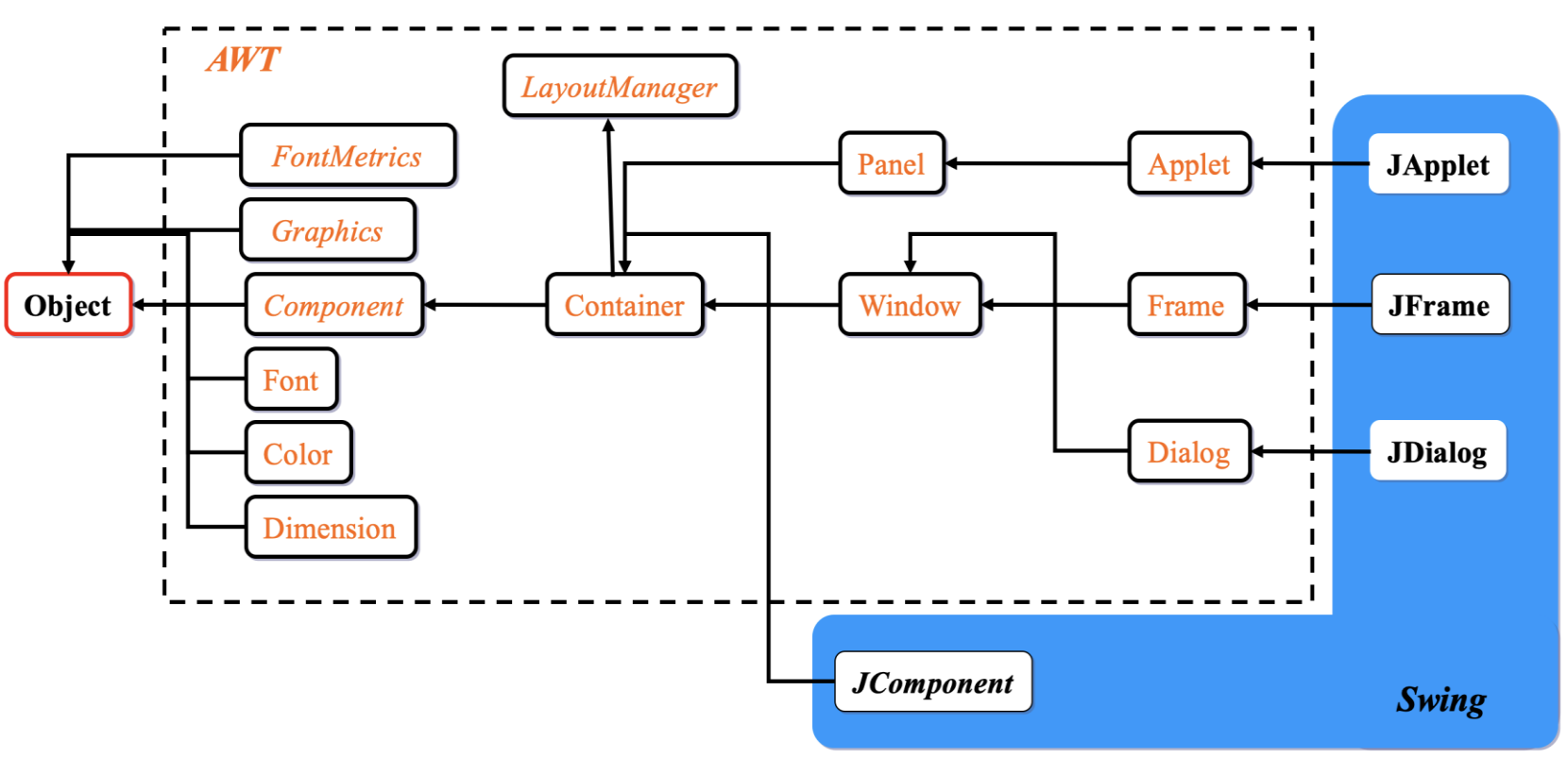


- Cung cấp một **tập hợp các thành phần giao diện** (GUI Components).
- Được sử dụng để **tạo chương trình trên máy tính**.
- GUI Components: Bảng (JTable), Nhãn (JLabel), Nút (JButton), vv.

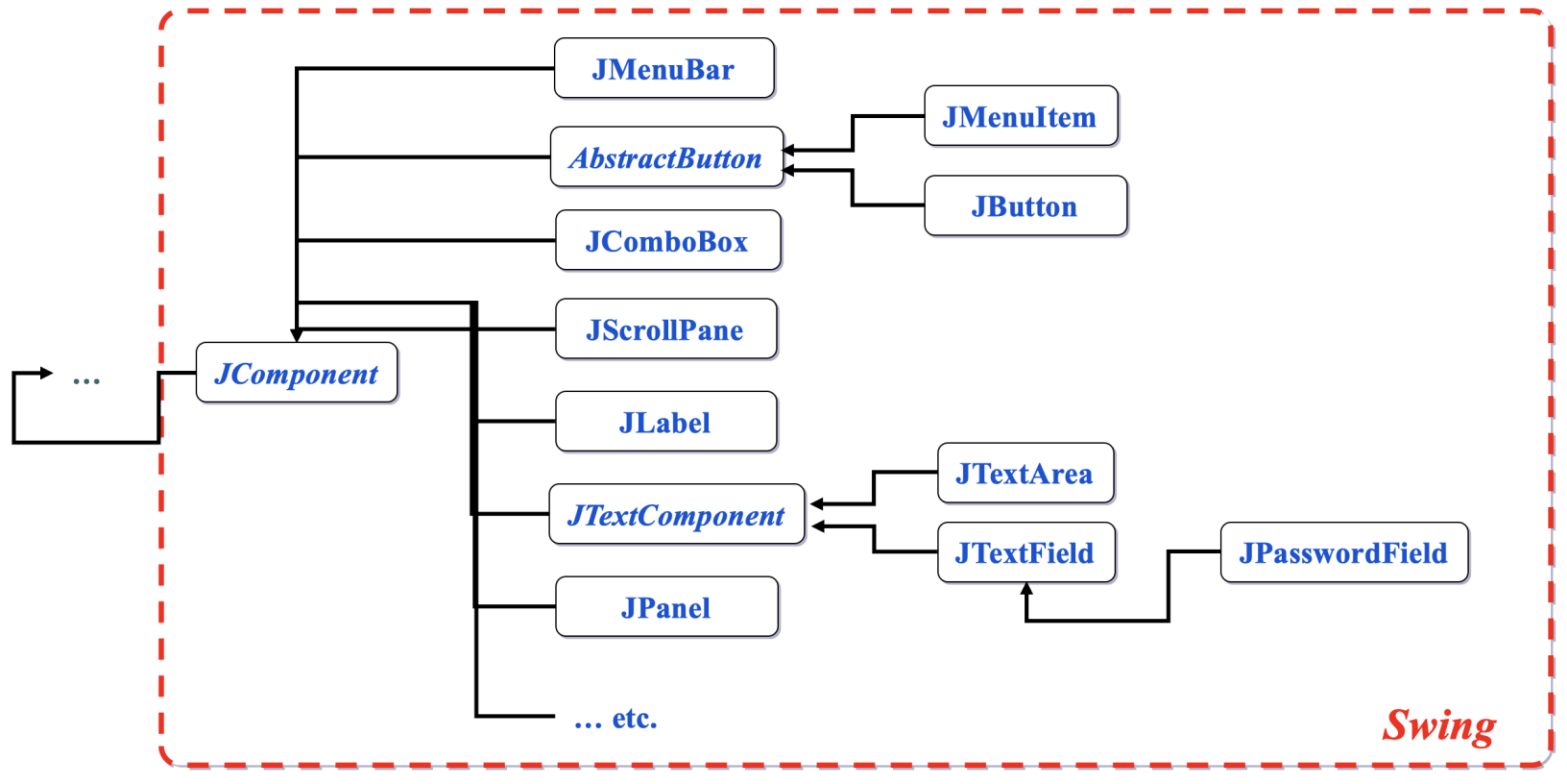
Các gói GUI (Packages)

AWT	Swing
• java.awt • java.awt.color • java.awt.font • java.awt.image • java.awt.event • …	• javax.swing • javax.swing.color • javax.swing.font • javax.swing.image • javax.swing.event • …

Java GUI Hierarchy



Java GUI Hierarchy (tiếp theo)



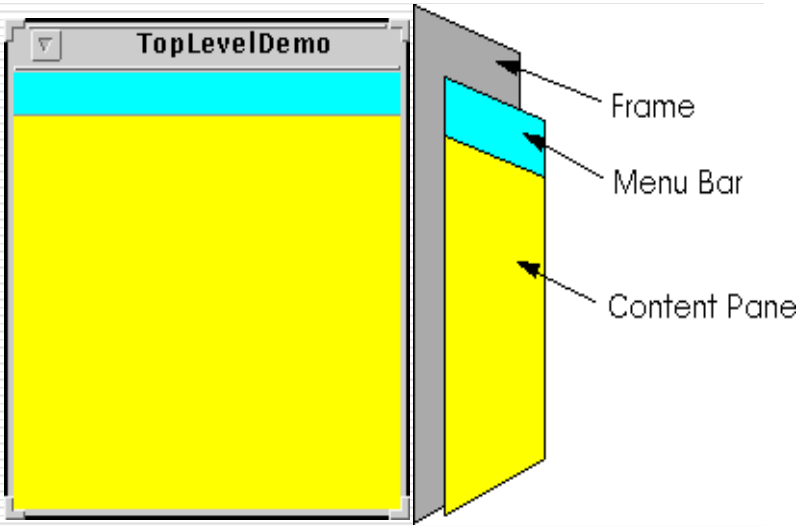
Principles for building GUI in Java

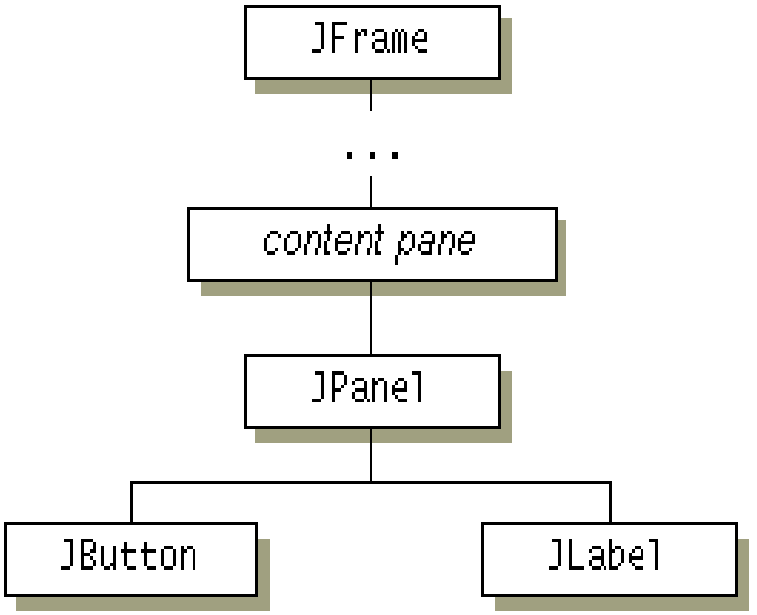
1. Choosing a certain container: JFrame, JWindow, JDialog, …
2. Creating components: JButton, JTextarea, JLabel, …
3. Putting these components into containers.
4. Arranging components (Layout).
5. Adding event handlers (Listener).

Containers

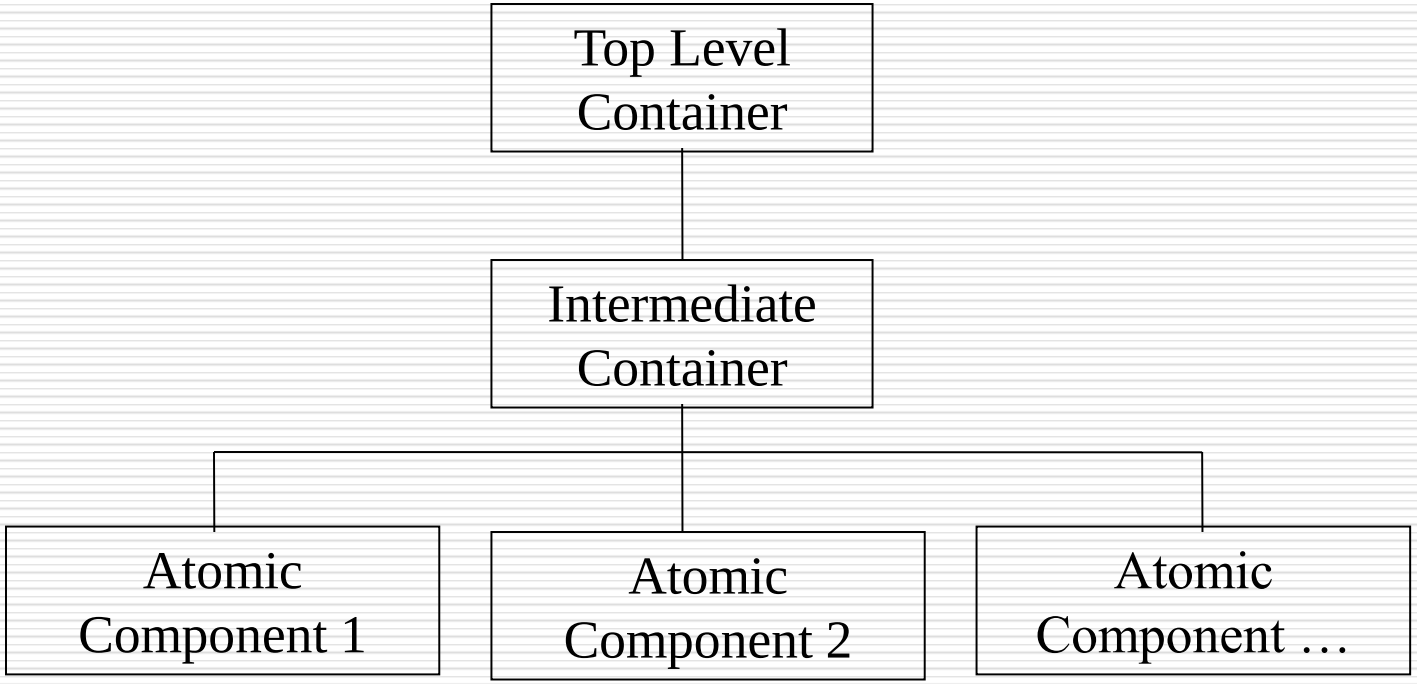
- Là thành phần có thể chứa những thành phần khác: JFrame, JPanel, JDialog, JScrollPane, …
- Giao diện GUI được trưng bày trên một vật chứa (container) có tổ chức phân cấp.
- Container cấp cao quản lý gián tiếp các container trung gian cho phép tổ chức giao diện theo chiều sâu.

Containment Hierarchy



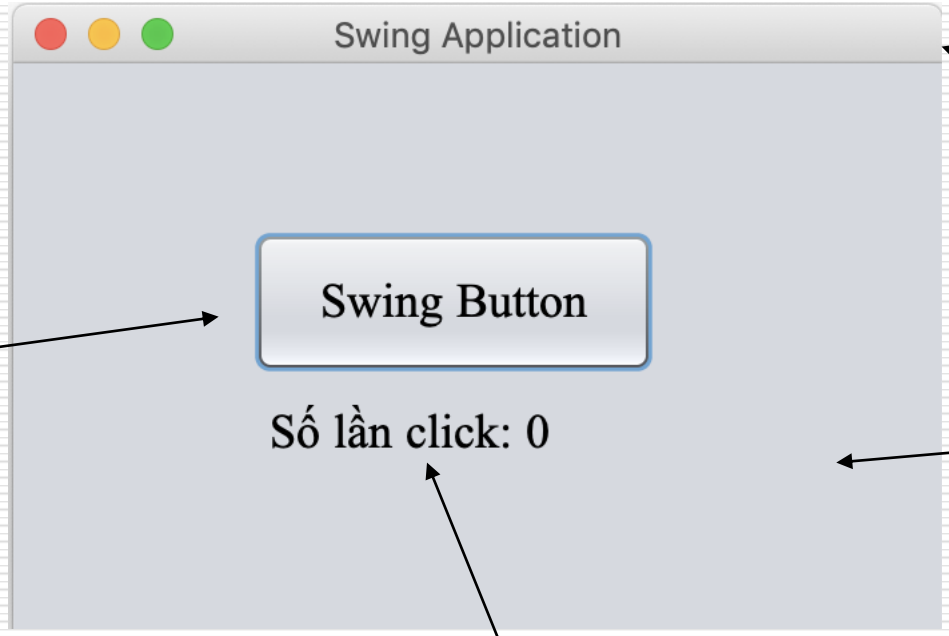


Containment Hierarchy (tiếp theo)



Containment Hierarchy (tiếp theo)

- **Ví dụ:**



Top Level

(JFrame)

Atomic

(JButton)

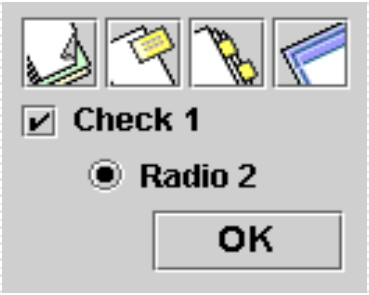
Intermediate

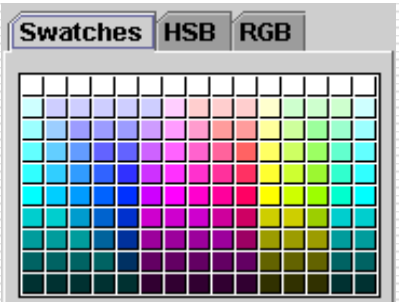
(JPanel)

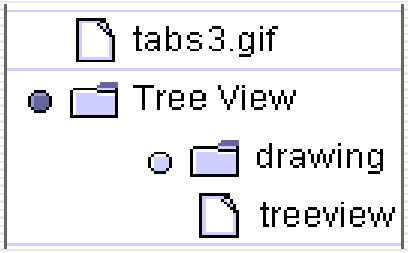
Atomic

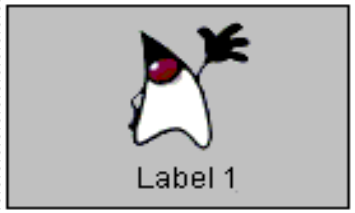
(JLabel)

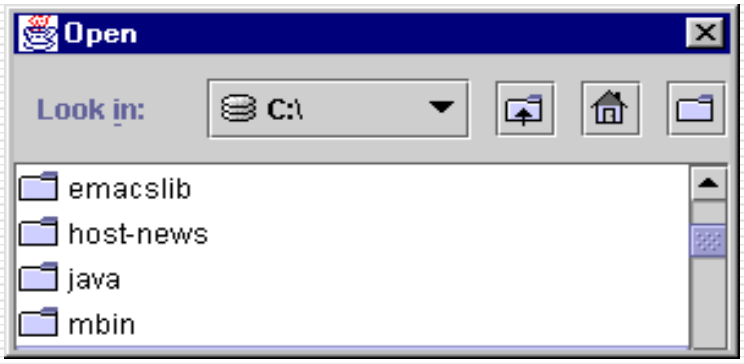
Swing Components

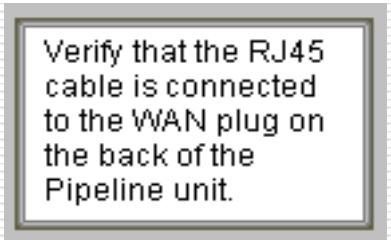


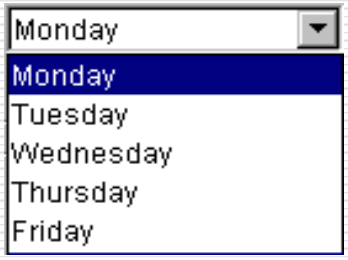












Class JComponent

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, JLayer, JLayeredPane, JList, JMenuBar, JOptionPane, JPanel, JPopupMenu, JProgressBar, JRootPane, JScrollBar, JScrollPane, JSeparator, JSlider, JSpinner, JSplitPane, JTabbedPane, JTable, JTableHeader, JTextComponent, JToolBar, JToolTip, JTree, JViewport

Các phương thức phổ biến

- Phương thức dùng để gắn components

```
objectName.add(…);
```

- Phương thức dùng để lấy thuộc tính

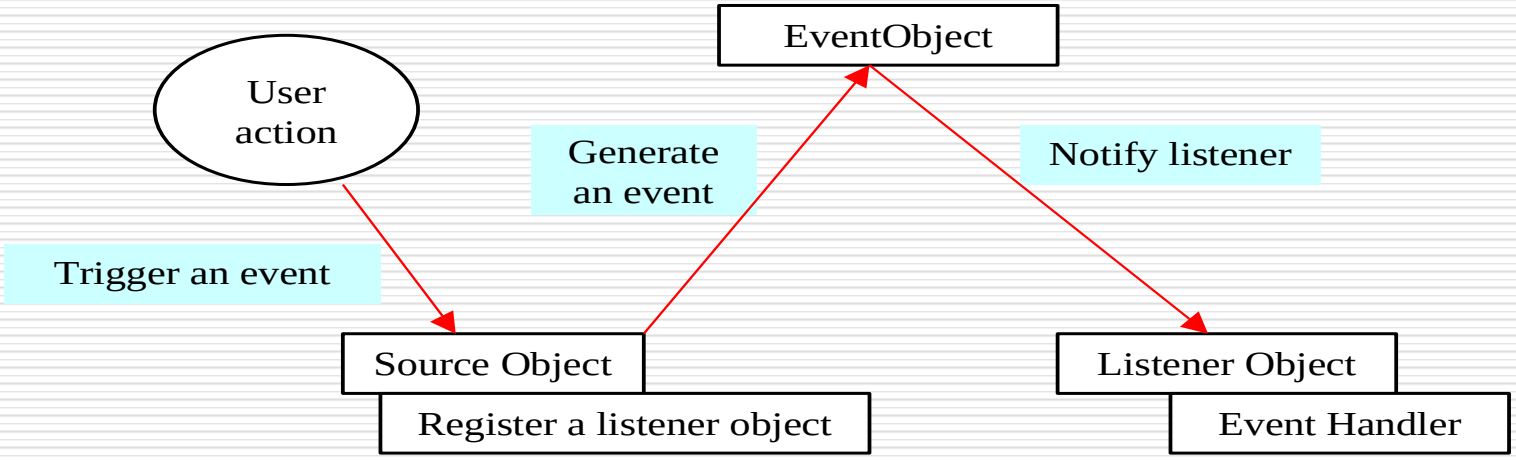
```
objectName.getyyy( );
```

- Phương thức dùng để gán thuộc tính

```
objectName.setyyy( );
```

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)



Bài tập trên lớp

- Tạo mini app cho nhập vào hai số a và b. Yêu cầu: xuất ra màn hình là tổng của hai số a và b.

Event và Listener

Đối tượng	Event	Listener
Window, Frame, …	WindowEvent	WindowListener
Button, MenuItem, …	**ActionEvent**	ActionListener
TextComponent, …	TextEvent	TextListener
List, …	ActionEvent	ActionListener
…	ItemEvent	ItemListener
	ComponentEvent	ComponentListener
	MouseEvent	MouseListener
		MouseMotionListener
	KeyEvent	KeyListener

JLabel

- Là một thành phần có chức năng hiển thị thông tin dưới dạng văn bản.

Các constructor	Các thuộc tính

`JLabel()`
`JLabel(String text)`
`JLabel(String text,int hAlignment)`
`JLabel(Icon icon)`
`JLabel(Icon icon, int hAlignment)`
`JLabel(String text,Icon icon,int hAlignment)`
	`text`
	`icon`
	`horizontalAlignment`
	`verticalAlignment`

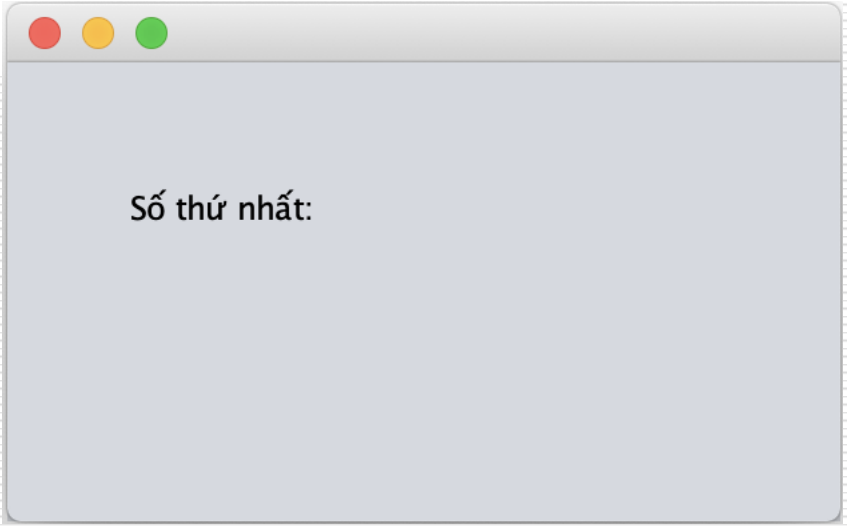
JLabel (tiếp theo)

- Các phương thức thông dụng:
  - `getText()`: Trả về chuỗi ký tự trong JLabel.
  - `setText(String text)`: Đặt chuỗi ký tự trong JLabel.
  - `setEditable(boolean editable)`: Cho phép hoặc vô hiệu hóa soạn thảo trong JLabel. Mặc định, `editable` là `true`.
  - `setForeground(Color foreground)` : Thiết lập màu chữ trong JLabel.

JLabel (tiếp theo)

- Ví dụ:

```
//Khởi tạo JLabel với nội dung "Số thứ nhất:"
JLabel lblSoThuNhat = new JLabel("Số thứ nhất:");
//lblSoThuNhat với vị trí (50 50) và chiều dài - cao (100, 20)
lblSoThuNhat.setBounds(50, 50, 100, 20);
//add lblSoThuNhat vào JFrame this
this.add(lblSoThuNhat);
```



JTextField

- Là ô nhập dữ liệu dạng văn bản trên một dòng

Các constructor	Các thuộc tính

`JTextField()`
`JTextField(int columns)`
`JTextField(String text)`
`JTextField(String text, int columns)`
	• `text`
	• `horizontalAlignment`
	• `editable`
	• `columns`

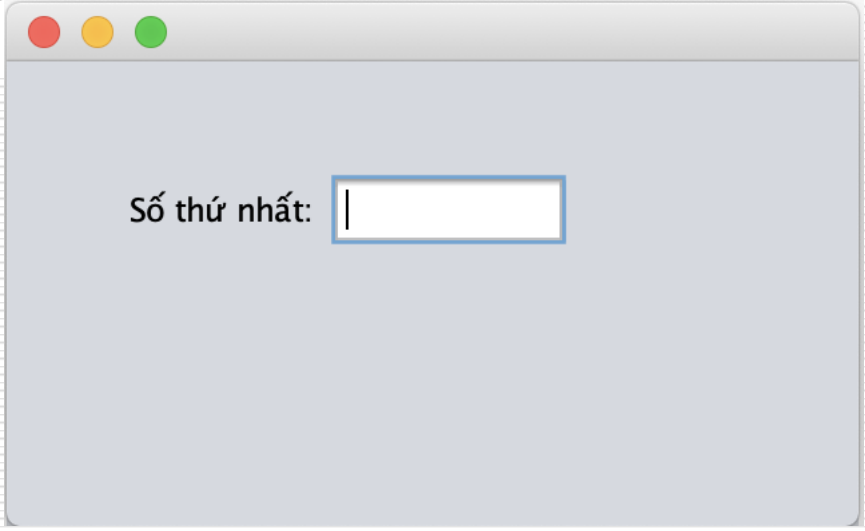
JTextField (tiếp theo)

- Các phương thức thông dụng:
  - `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.

JTextField (tiếp theo)

- Ví dụ:

```
//Khởi tạo JTextField
JTextField txtSoThuNhat = new JTextField("");
//txtSoThuNhat với vị trí (150, 50) và chiều dài - cao (100, 20)
txtSoThuNhat.setBounds(150, 50, 100, 20);
//add txtSoThuNhat vào JFrame this
this.add(txtSoThuNhat);
```



JButton

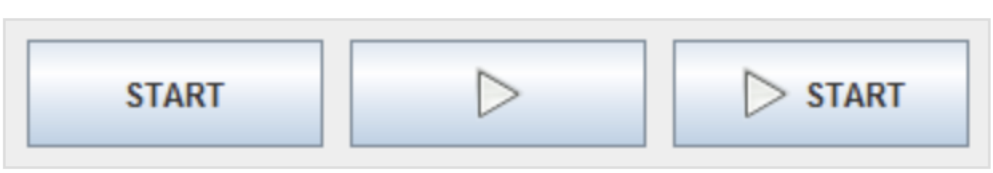
- Là một thành phần tạo ra một sự kiện hành động khi được kích chuột.

Các constructor	Các thuộc tính

`JButton()`
`JButton(String text)`
`JButton(String text, Icon icon)`
`JButton(Icon icon)`
	`text`
	`icon`
	`mnemonic`
	`horizontalAlignment`
	`verticalAlignment`
	`horizontalTextPosition`
	`verticalTextPosition`

ImageIcon

- JComponents (such as Jlabel and JButton) support a text label and an image icon.



ImageIcon (Cont.)

```
// Construct an ImageIcon from an image filename
String imgFilename = "images/duke.png";
      // Can use an absolute filename such as "c:/project/images/nought.gif"
ImageIcon iconDuke = new ImageIcon(imgFilename);

// OR
// Construct an ImageIcon via an image URL (in the form of file://path/filename)
ImageIcon iconDuke = null;
String imgFilename = "images/duke.png";
java.net.URL imgURL = getClass().getClassLoader().getResource(imgFilename);
      // Filename always relative to the root of the project (i.e., bin)
      // can access resource in a JAR file
if (imgURL != null) {
   iconDuke = new ImageIcon(imgURL);
} else {
   System.err.println("Couldn't find file: " + imgFilename);
}
```

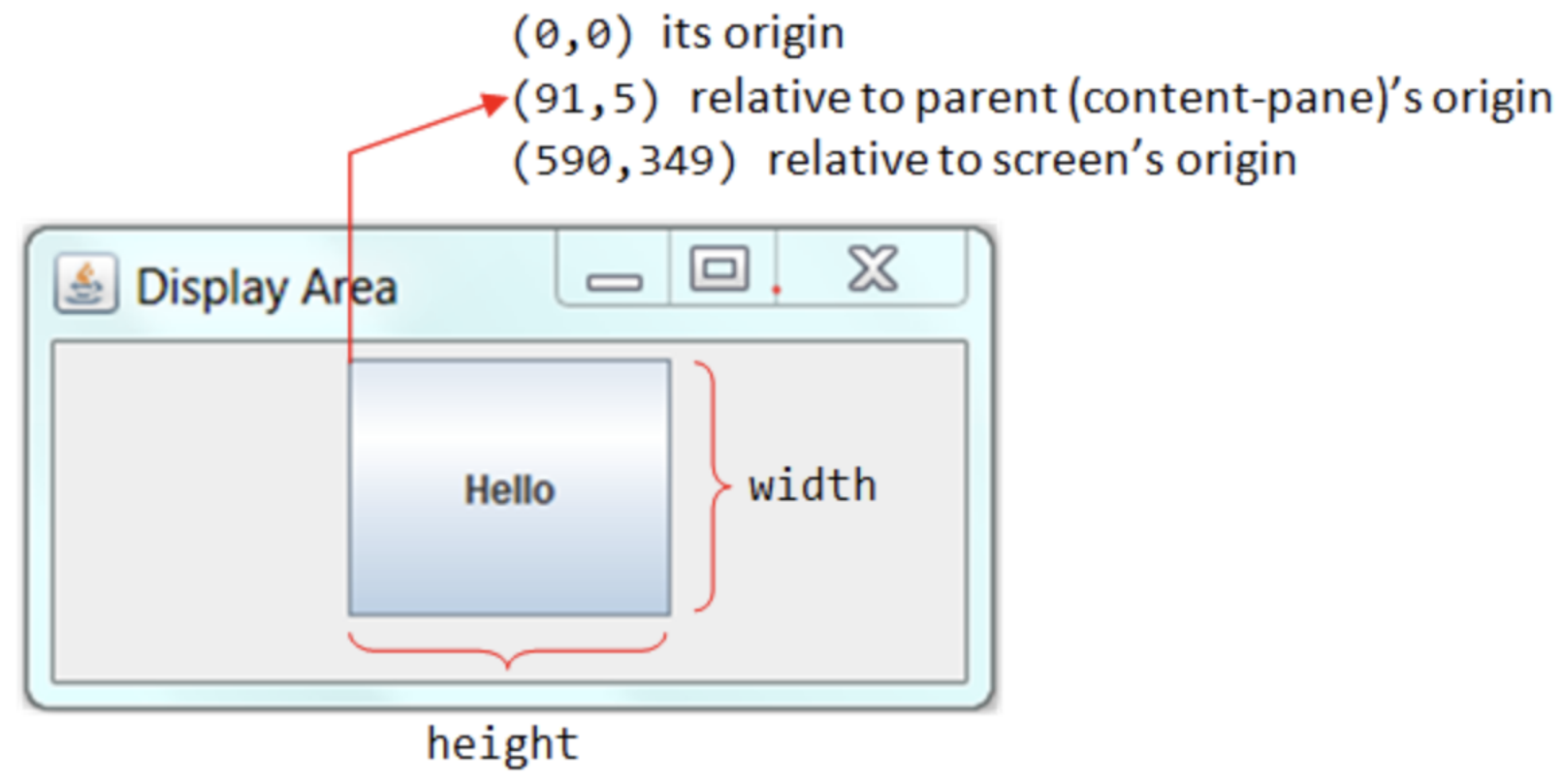
Display Area

```
1  import java.awt.*;
2  import javax.swing.*;
3
4  public class TestSize {
5     public static void main(String[] args) {
6        JFrame frame = new JFrame("Display Area");
7        Container cp = frame.getContentPane();
8        cp.setLayout(new FlowLayout());
9        JButton btnHello = new JButton("Hello");
10       btnHello.setPreferredSize(new Dimension(100, 80));
11       cp.add(btnHello);
12
13       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14       frame.setSize(300, 150);  // or pack() the components
15       frame.setLocationRelativeTo(null);  // center the application window
16       frame.setVisible(true);             // show it
17
18       System.out.println(btnHello.getSize());
19       System.out.println(btnHello.getLocation());
20       System.out.println(btnHello.getLocationOnScreen());
21
22       System.out.println(cp.getSize());
23       System.out.println(cp.getLocation());
24       System.out.println(cp.getLocationOnScreen());
25
26       System.out.println(frame.getSize());
27       System.out.println(frame.getLocation());
28       System.out.println(frame.getLocationOnScreen());
29    }
30 }
```

Display Area (Cont.)

```
java.awt.Dimension[width=100,height=80]   // JButton's getSize()
java.awt.Point[x=91,y=5]                  // JButton's getLocation()
java.awt.Point[x=590,y=349]               // JButton's getLocationOnScreen()

java.awt.Dimension[width=282,height=105]  // ContentPane's getSize()
java.awt.Point[x=0,y=0]                   // ContentPane's getLocation()
java.awt.Point[x=499,y=344]               // ContentPane's getLocationOnScreen()

java.awt.Dimension[width=300,height=150]  // JFrame's getSize()
java.awt.Point[x=490,y=308]               // JFrame's getLocation()
java.awt.Point[x=490,y=308]               // JFrame's getLocationOnScreen()
```

Display Area (Cont.)



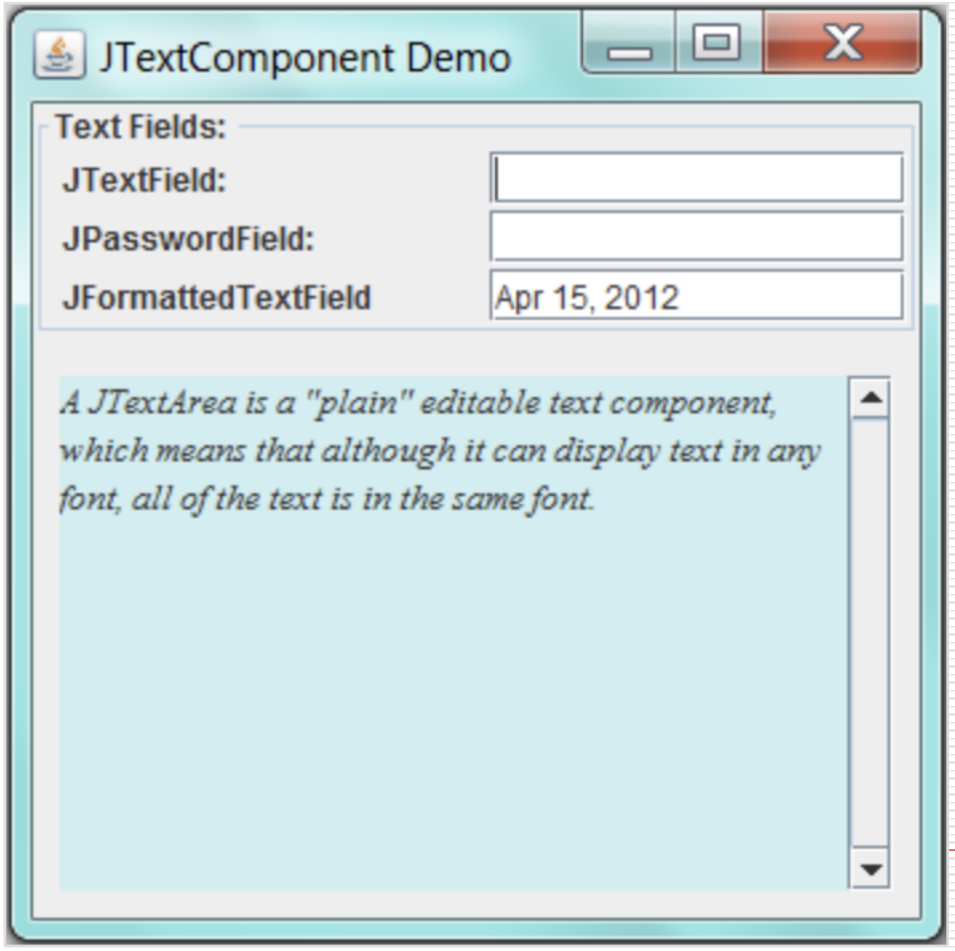
Display Area (Cont.)

- public void setSize(int width, int height)
- public void setLocation(int x, int y)
- public void setBounds(int x, int y, int width, int height)
- public void setSize(Dimension dim)
- public void setLocation(Point origin)

Component groups

- **Text Components:** JTextField, JTextArea, JEditorPane.
- **Buttons and ComboBox:** JButton, JCheckBox, JRadioButton, JComboBox.
- **Menu-Bar:** JMenuBar, JMenu, JMenuItem
- **JOptionPane: I**nteracting with the User

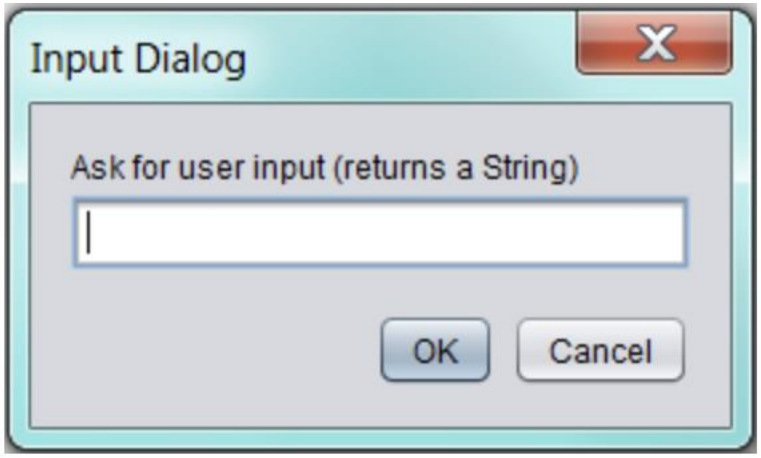
Text Components: JTextField, JTextArea, JEditorPane.

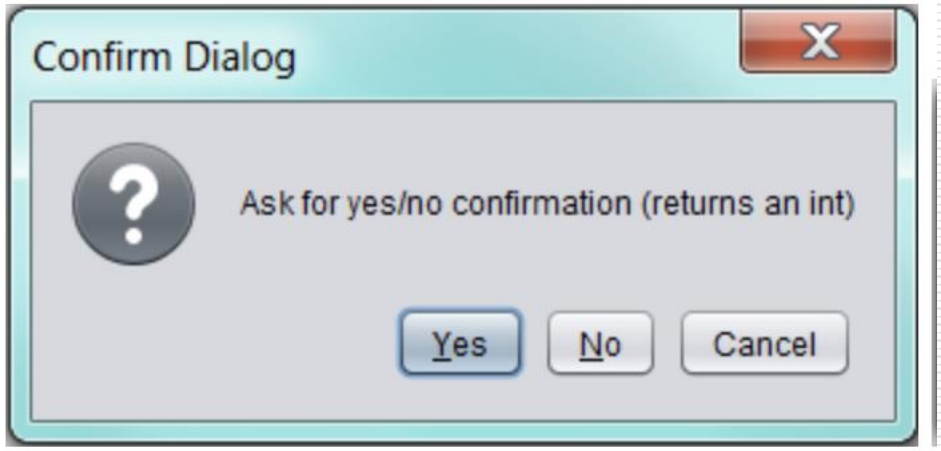


Input, Confirm and Message Dialogs

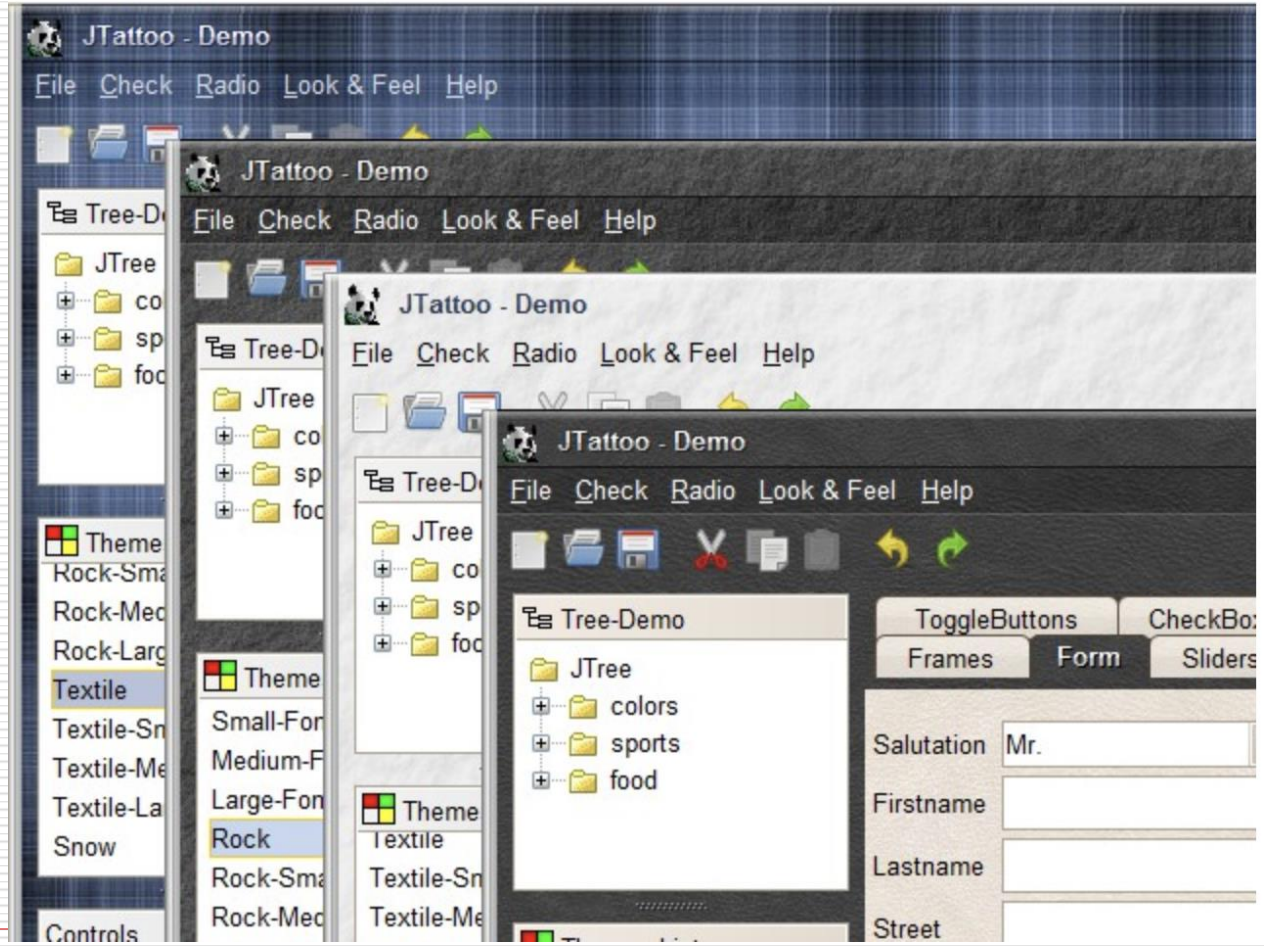
```
1  import javax.swing.*;
2  public class JOptionPaneTest {
3     public static void main(String[] args) {
4        // JOptionPane does not have to run under a Swing Application (extends JFrame).
5        // It can run directly under main().
6        String inStr = JOptionPane.showInputDialog(null, "Ask for user input (returns a String)",
7              "Input Dialog", JOptionPane.PLAIN_MESSAGE);
8        System.out.println("You have entered " + inStr);
9        JOptionPane.showMessageDialog(null, "Display a message (returns void)!",
10             "Message Dialog", JOptionPane.PLAIN_MESSAGE);
11       int answer = JOptionPane.showConfirmDialog(null, "Ask for confirmation (returns an int)",
12             "Confirm Dialog", JOptionPane.YES_NO_CANCEL_OPTION);
13       switch (answer) {
14          case JOptionPane.YES_OPTION:
15             System.out.println("You clicked YES"); break;
16          case JOptionPane.NO_OPTION:
17             System.out.println("You clicked NO"); break;
18          case JOptionPane.CANCEL_OPTION:
19             System.out.println("You clicked Cancel"); break;
20       }
21    }
22 }
```

Input, Confirm and Message Dialogs (Cont.)





Look and Feel



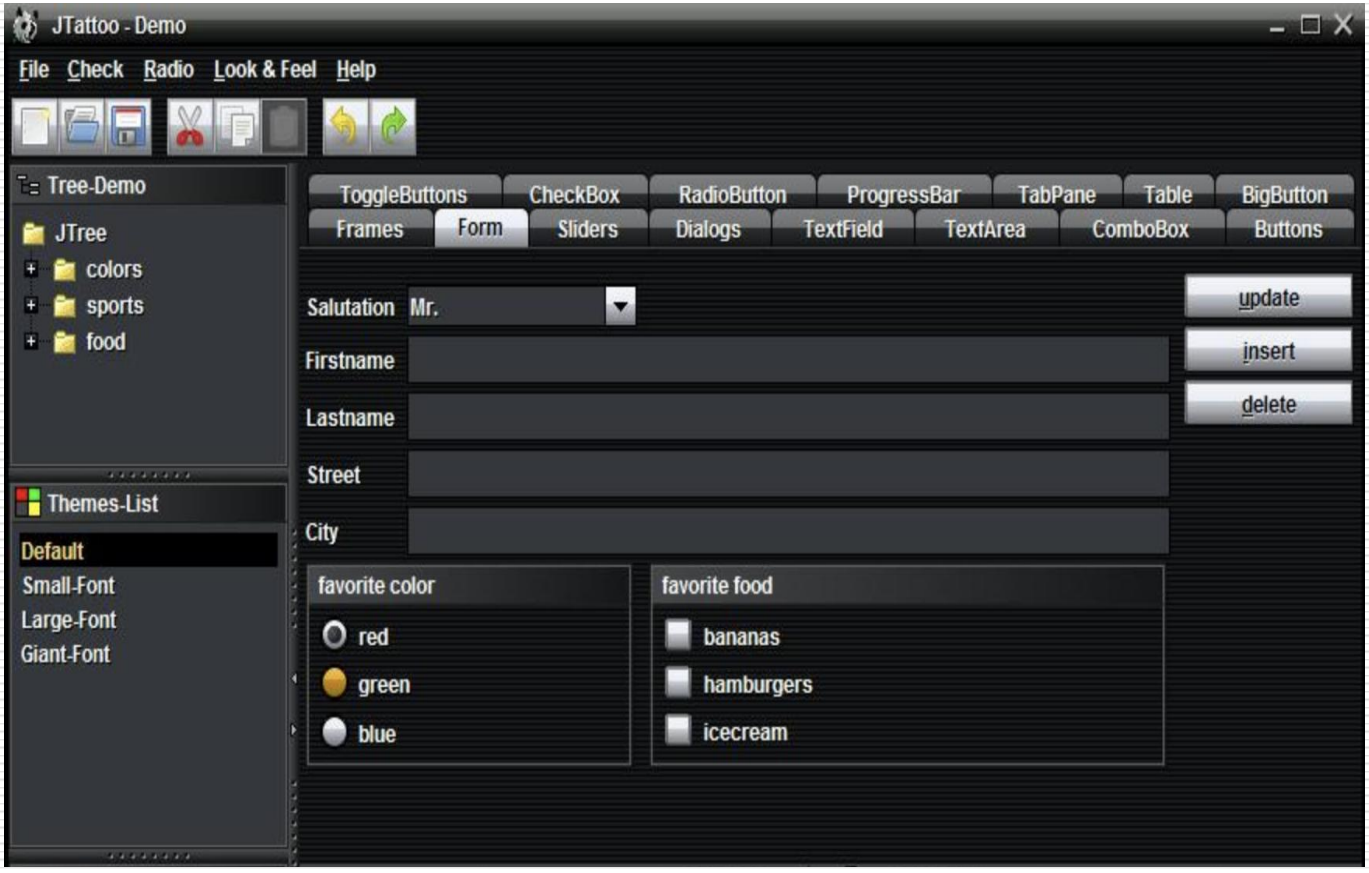
Look and Feel (Cont.)

- UIManager.setLookAndFeel(String *className*)
- **For example,**

Set NoireLookAndFeel in **Jtatoo**

UIManager.setLookAndFeel("com.jtattoo.plaf.noire.NoireLookAndFeel");

Look and Feel (Cont.)



Q&A

Thank you!