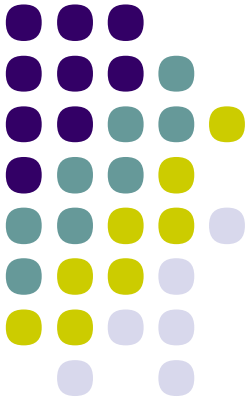
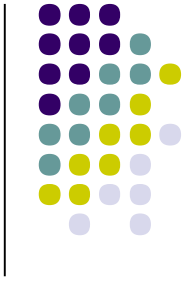
Introduce Graphic User Interface





Nội dung

- Giới thiệu AWT và Swing
- Xây dựng Java GUI cơ bản
- Cơ chế kiểm soát sự kiện người dùng

Giới thiệu về AWT

- AWT (Abstract Window Toolkit) (java.awt.*) cung cấp một tập hợp các lớp dùng để viết giao diện người dùng dạng đồ họa
- Bộ khung (framework) GUI cũ cho Java (Java 1.1)

Giới thiệu về AWT

- Đặc điểm:
  - Bao gồm tập hợp các lớp ngang hàng, tức là giao diện lập trình ứng dụng cho các tính năng cửa sổ hiện có được cung cấp bởi hệ điều hành.
  - AWT cung cấp hai mô hình xử lý biến cố:
    - Mô hình thừa hưởng (mô hình phân cấp)
    - Mô hình ủy quyền
  - AWT cung cấp các lớp chứa (container) và các thành phần (component) để đơn giản hóa việc xây dựng các chương trình.
  - AWT quản lý bố cục theo các sơ đồ tổ chức khác nhau.

Giới thiệu về AWT

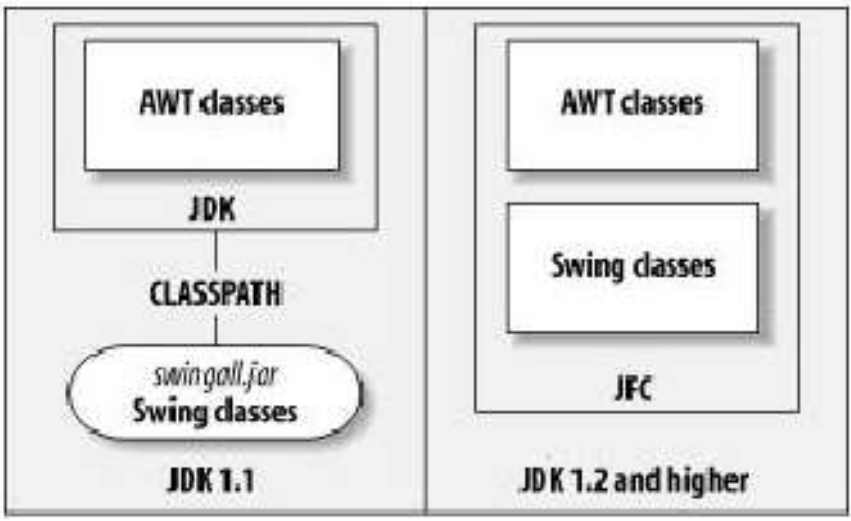
- Hạn chế:
  - Chiếm nhiều tài nguyên hệ thống (heavyweight object)
  - Khó mở rộng (không có các công nghệ hỗ trợ)
  - Một số dựa vào các bản sao mã bản ngữ (native code)
  - Gặp các vấn đề độc lập hệ nền
    - Phụ thuộc vào các thành phần GUI của hệ điều hành

JFC (Java Foundation Classes)

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

Giới thiệu về SWING

- Swing (javax.swing.*)
  - Bộ khung GUI mới được giới thiệu đầu tiên trong Java 1.2
  - Bao gồm tất cả những đặc tính của AWT cộng với nhiều đặc tính tiên tiến khác
  - Thuần Java, các thành phần nhẹ (lightweight) (không dựa vào mã bản ngữ)
  - Kiến trúc cảm quan (look and feel)



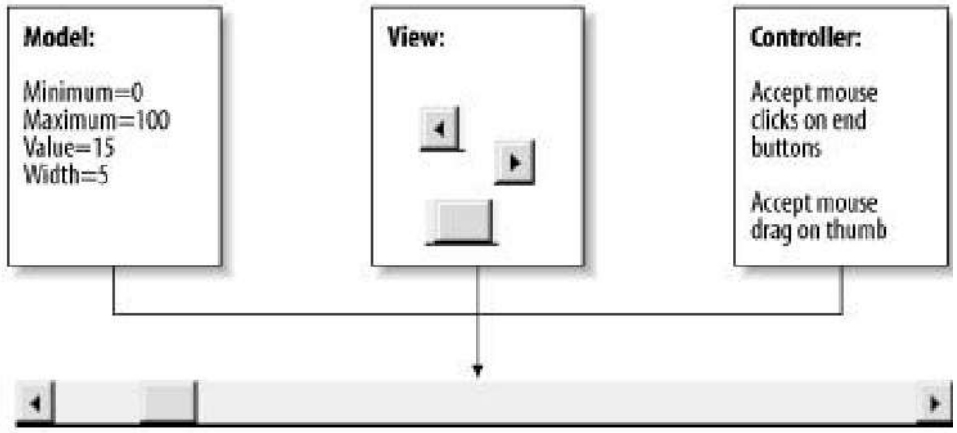
Giới thiệu về SWING

- Các ưu điểm của Swing:
  - Các thành phần của Swing chiếm ít tài nguyên hệ thống hơn vì chúng không ngang hàng riêng trong hệ điều hành.
  - Hỗ trợ khái niệm "`pluggable look-and-feel`", cung cấp thêm nhiều diện mạo để người dùng lựa chọn
  - Hỗ trợ các công nghệ nhập xuất mới: tiếng nói và thao tác không mouse
  - Dễ dàng mở rộng:
    - Button hỗ trợ cả văn bản và đồ họa
    - Sử dụng HTML trong Label
    - …

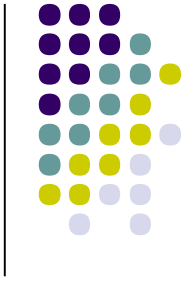
Kiến trúc MVC của Swing

- Swing sử dụng kiến trúc MVC để xây dựng các thành phần, chương trình của mình, MVC chia mỗi thành phần giao diện thành 3 phần;
  - Model
    - Phần chứa nội dung trạng thái của các thành phần GUI
    - Mỗi kiểu thành phần GUI có 1 model khác nhau
  - View
    - Thể hiện trực quan thành phần GUI
  - Controller
    - Quản lý cách thức tương tác giữa các thành phần GUI với các sự kiện người dùng: click chuột, nhập phím…

Kiến trúc MVC của Swing



Kiến trúc MVC thông thường



Model

The MODEL passes its data to the VIEW for rendering

View

The COTROLLER updates the MODEL based on events receiced

The VIEW determines which events passed to the CONTROLLER

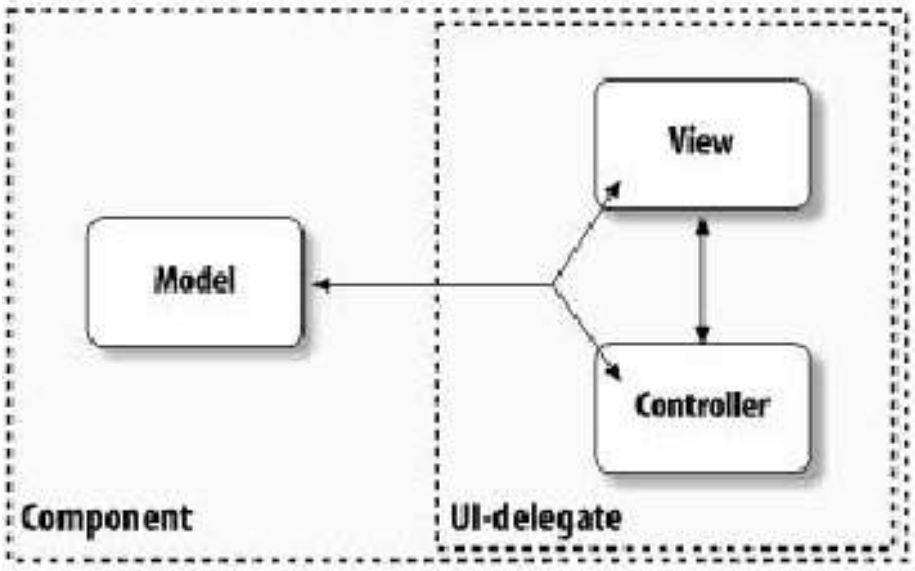
Controller

MVC trong Swing

- Swing hiện thực 1 mô hình `MVC` khá đơn giản còn được gọi là `model-delegate`
- Mô hình này nối `View` và `Controller` thành 1 đối tượng duy nhất gọi là `UI-delegate`
- UI-delegate làm 2 nhiệm vụ:
  - Hiển thị thành phần Swing lên màn hình
  - Quản lý các sự kiện

MVC trong Swing

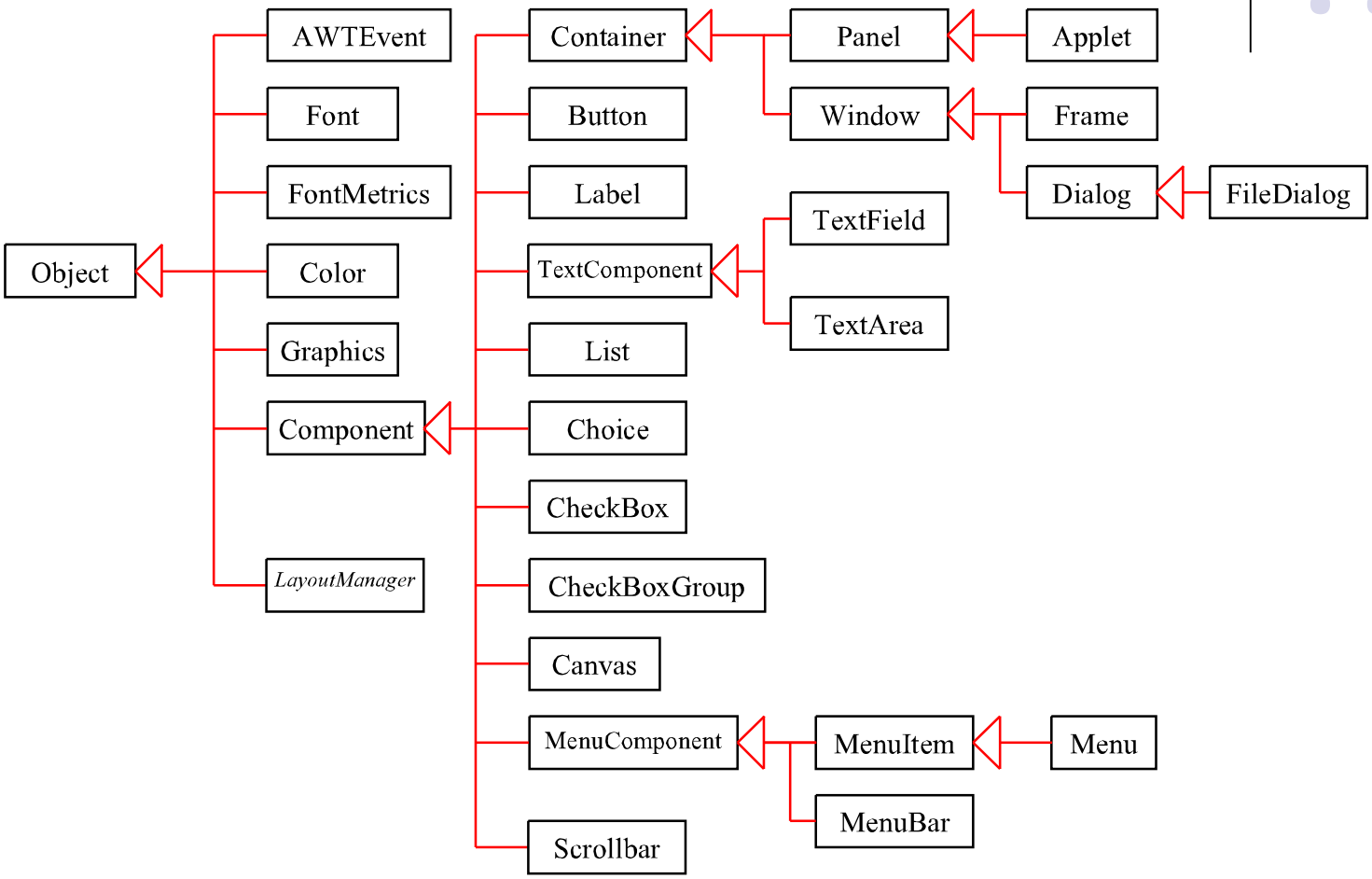
- Mỗi thành phần GUI gồm 2 phần:
  - Model
  - UI-delegate



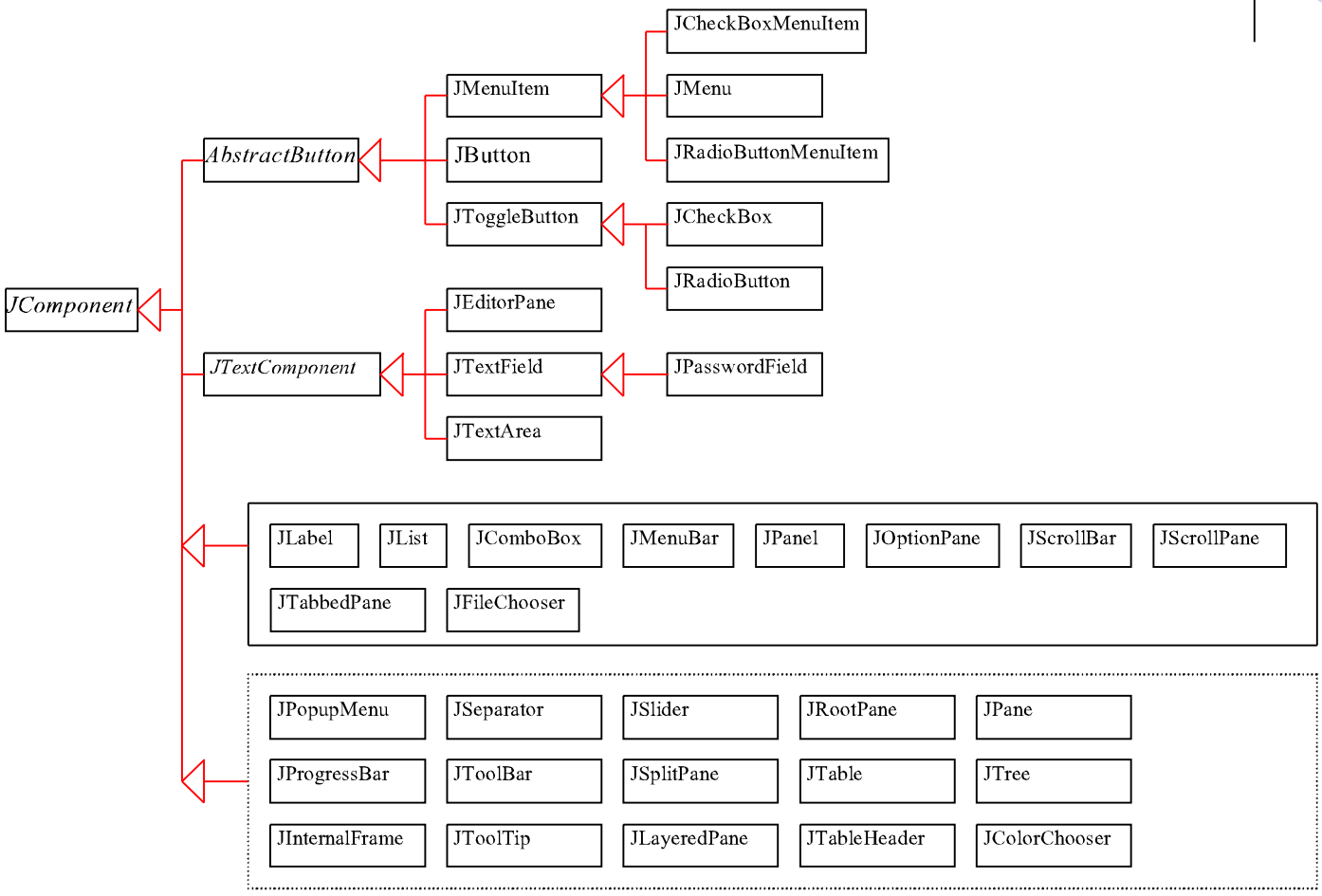
Giới thiệu Java GUI

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

GUI Class Hierarchy (AWT)



GUI Class Hierarchy (Swing)



Ví dụ: Tạo cửa sổ với Swing

- Ứngng dụng HelloWorld cơ bản
- Tạo một Cửa sổ với “HelloWorldString” trong thanh tiêu đề và hiển thị label “Hello World”

```java
import javax.swing.*;

public class HelloWorldSwing {
    private static void createAndShowGUI() {
        // Create and set up the window.
        JFrame frame = new JFrame("HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Add the ubiquitous "Hello World" label.
        JLabel label = new JLabel("Hello World");
        frame.getContentPane().add(label);

        // Display the window.
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        // Schedule a job for the event-dispatching thread:
        // creating and showing this application's GUI.
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

Cơ bản về thiết kế GUI

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

Component

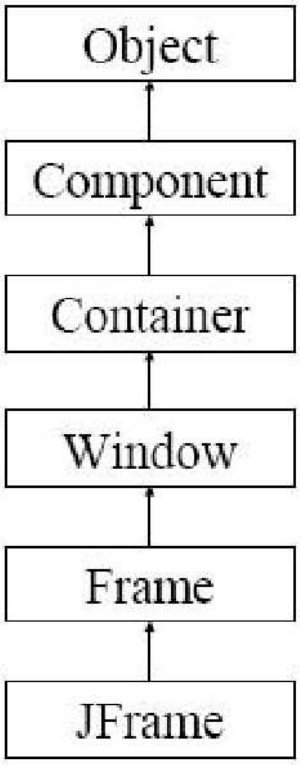
- Là các đối tượng có biểu diễn đồ họa được hiển thị lên màn hình mà người dùng tương tác được
- Ví dụ: nút nhấn, checkbox, scrollbar

Container

- Đối tượng vật chứa hay những đối tượng có khả năng quản lý và nhóm các đối tượng khác lại
- Một số đối tượng container trong java:
  - Panel
    - Đối tượng khung chứa đơn giản nhất dùng để nhóm các đối tượng, thành phần con lại với nhau
    - Một Panel có thể chứa 1 Panel khác
  - ScrollPanes
    - Tương tự Panel nhưng có thêm 2 thanh trượt giúp ta tổ chức và xem các đối tượng lớn
  - Dialogs
    - Là một cửa sổ dạng hộp thoại
    - Dùng để đưa ra các thông báo, lấy dữ liệu nhập từ người dùng

Container

- Các đối tượng container trong Java:
  - Frame, JFrame
    - Là một cửa sổ Windows ở mức trên cùng gồm tiêu đề và đường biên như các ứng dụng Windows thông thường khác
    - Thường được dùng để tạo ra cửa sổ chính cho các ứng dụng khác
  - Applet: Web Applet
  - JWindow: Không có thanh tiêu đề hay các nút điều khiển.



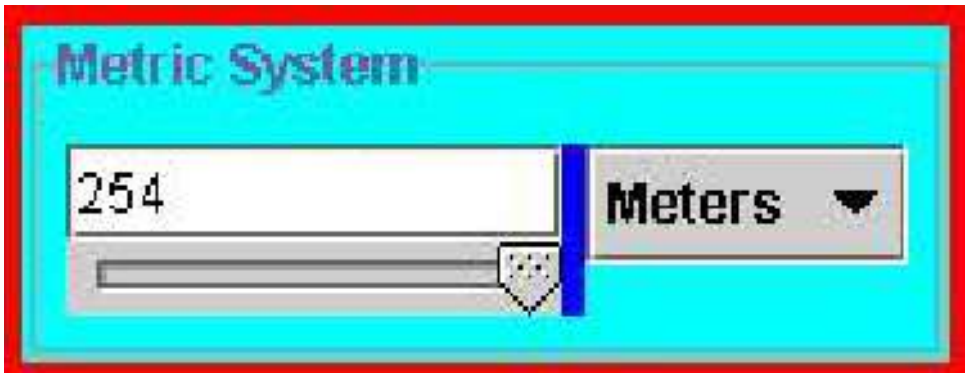
Container

- Top-level component: là thành phần trên cùng của bất kì Swing containment hierarchy nào.

Dialog **Frame** **Applet**

Container

- Intermediate containers: là thành phần đơn thuần dùng để chứa các component khác



Panel

Scroll pane

Container

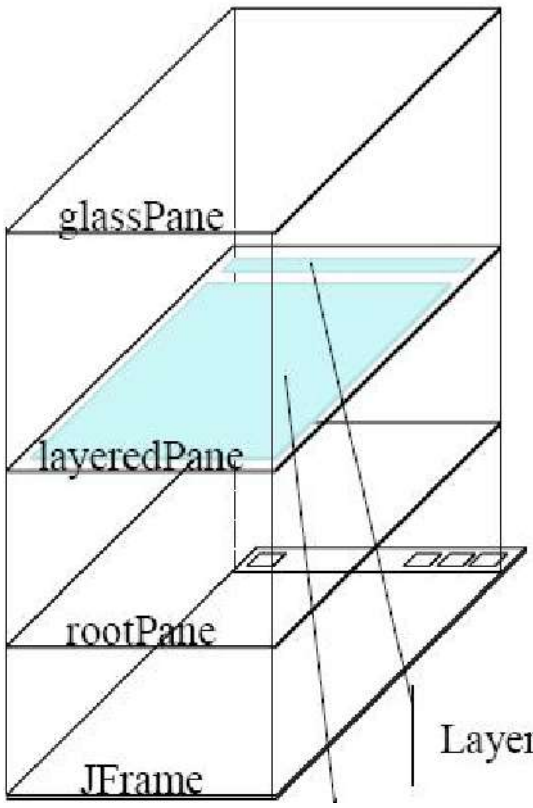
- Special-Purpose Containers là các thành phần chứa trung gian đặc biệt

Internal Frame

Layered pane

Cấu trúc JFrame

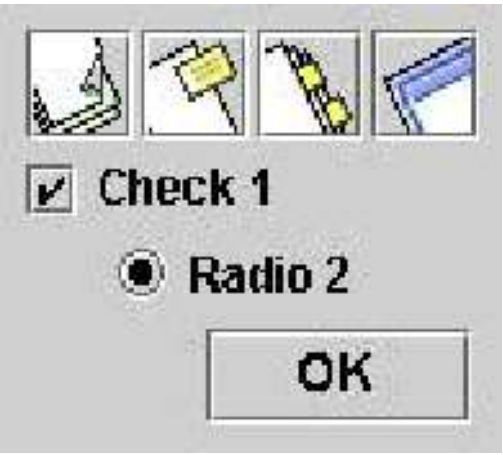
- Khung chứa đa tầng
- Hầu hết mọi thứ đặt vào trong khung nội dung (content panel)
  - getContentPane()
- Sử dụng glassPane cho Popup menus, một số hoạt cảnh,…
- Các phương thức
  - getRootPane()
  - getLayeredPane()
  - getContentPane()
  - getGlassPane()

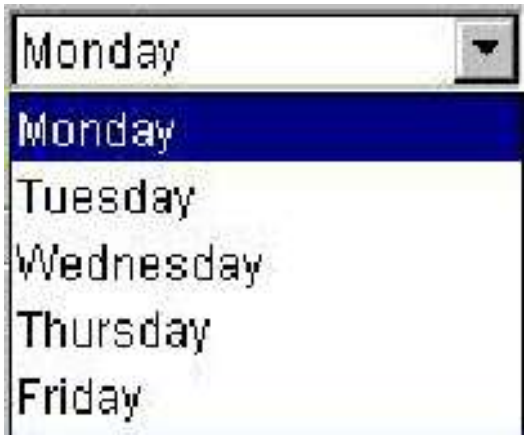


LayeredPane chứa contentPane

Các thành phần điều khiển cơ bản

- Dùng để nhận dữ liệu từ người dùng

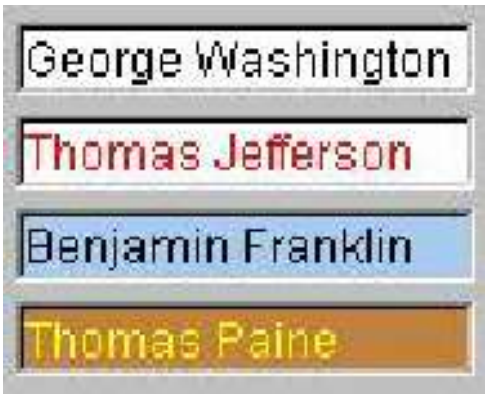




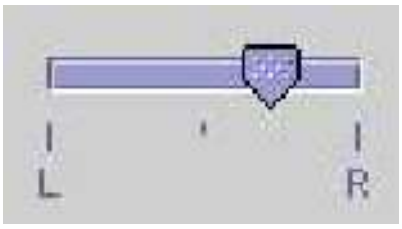
Buttons **Combo Box** **List**

Các thành phần điều khiển cơ bản

Menu



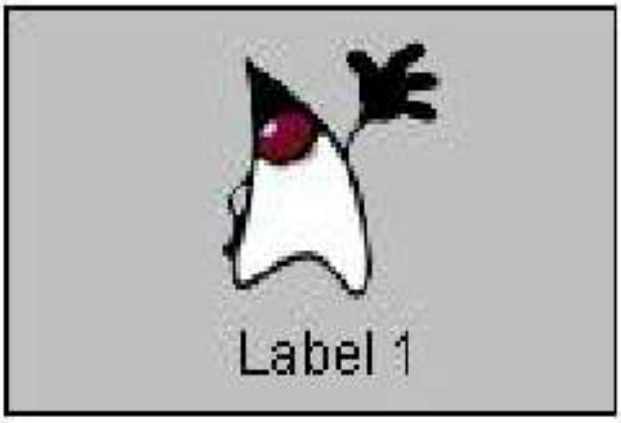
Text fields



Slider

Các thành phần thuần hiển thị thông tin

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



Label

Tool tip

Progress Bar

Các thành phần sửa chữa định dạng

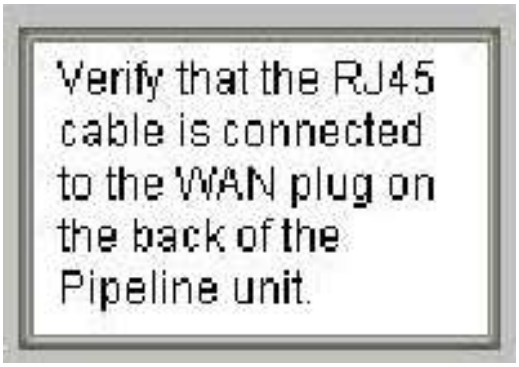
- Dùng để hiển thị các thông tin định dạng
- Cho phép người dùng lựa chọn định dạng

Color Chooser

File Chooser

Các thành phần hiển thị thông tin đã định dạng

- Cho phép người sử dụng sửa đổi thông tin



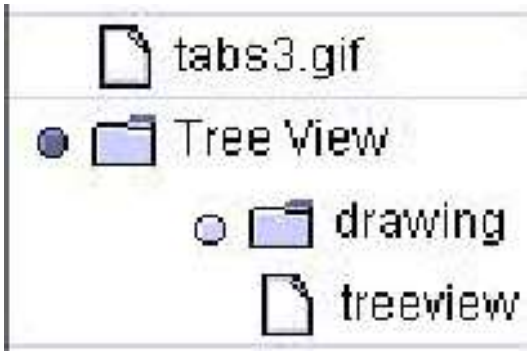


Table **Text** **Tree**

Quản lý bố cục (Layout manager)

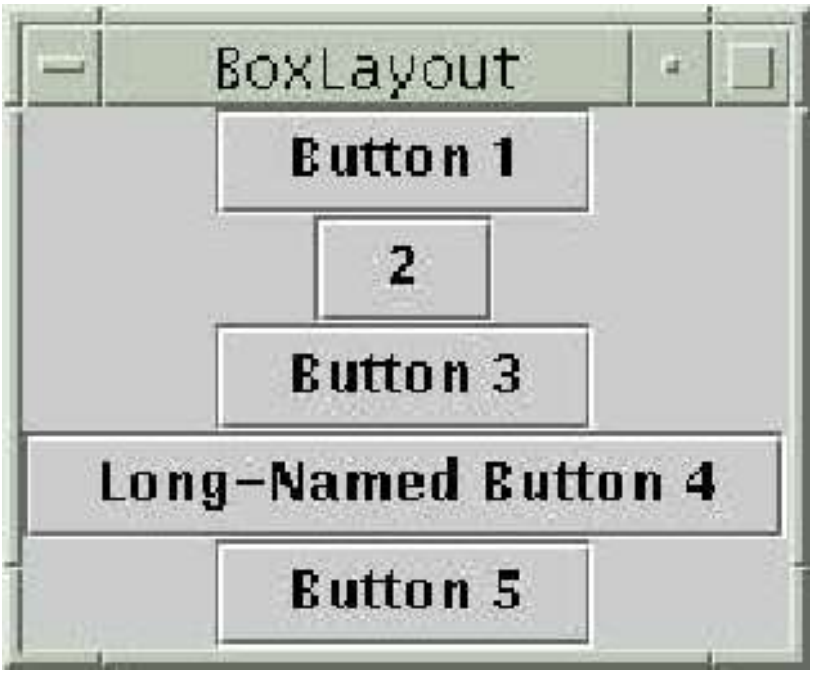
- Dùng để xác định kích thước và vị trí của các thành phần GUI
- Mỗi thành phần sẽ có 1 Layout manager mặc định
- Các Layout manager Java hỗ trợ:
  - BorderLayout
  - BoxLayout
  - CardLayout
  - FlowLayout
  - GridBagLayout
  - GridLayout
  - OverlayLayout
  - …

BorderLayout

- Chia thành 5 phần: NORTH, WEST, CENTER, EAST, SOUTH

BoxLayout

- Đưa các thành phần vào thành từng dòng hoặc từng cột



CardLayout

- Cho phép hiện thị nhiều component khác nhau tại nhiều thời điểm khác nhai

FlowLayout

- Thêm các đối tượng tuần tự từ trái sang phải

GridLayout

- Thêm các đối tượng tuần từ trái sang phải, từ trên xuống dưới vào các ô đã định sẵn

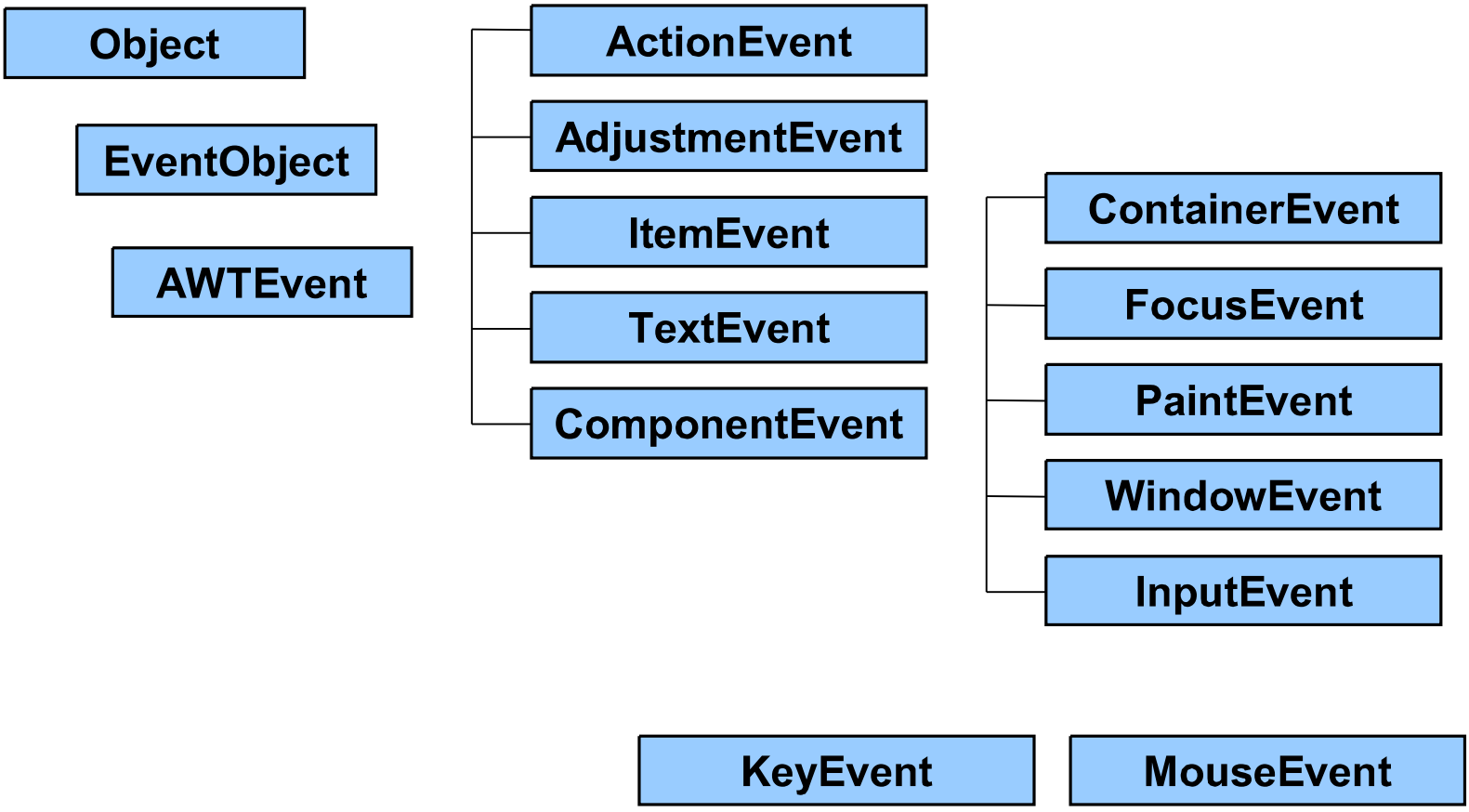
GridBagLayout

- Thêm các đối tượng vào các ô lưới đã định sẵn, nhưng cho phép người dùng mở rộng chỗ chứa cho các component (không chỉ 1 ô)

Bộ xử lý sự kiện (Event Handler)

- GUI là hệ thống hướng sự kiện (event-driven)
  - Chuột nhấn và chuyển động, nút nhấn và văn bản nhập thông qua bàn phím, nhấn vào các mục menu,…
  - Thao tác mong muốn sinh ra một hành động trên mỗi các sự kiện này
- Gói java.awt.event.*, java.swing.event.*

Gói java.awt.event.*



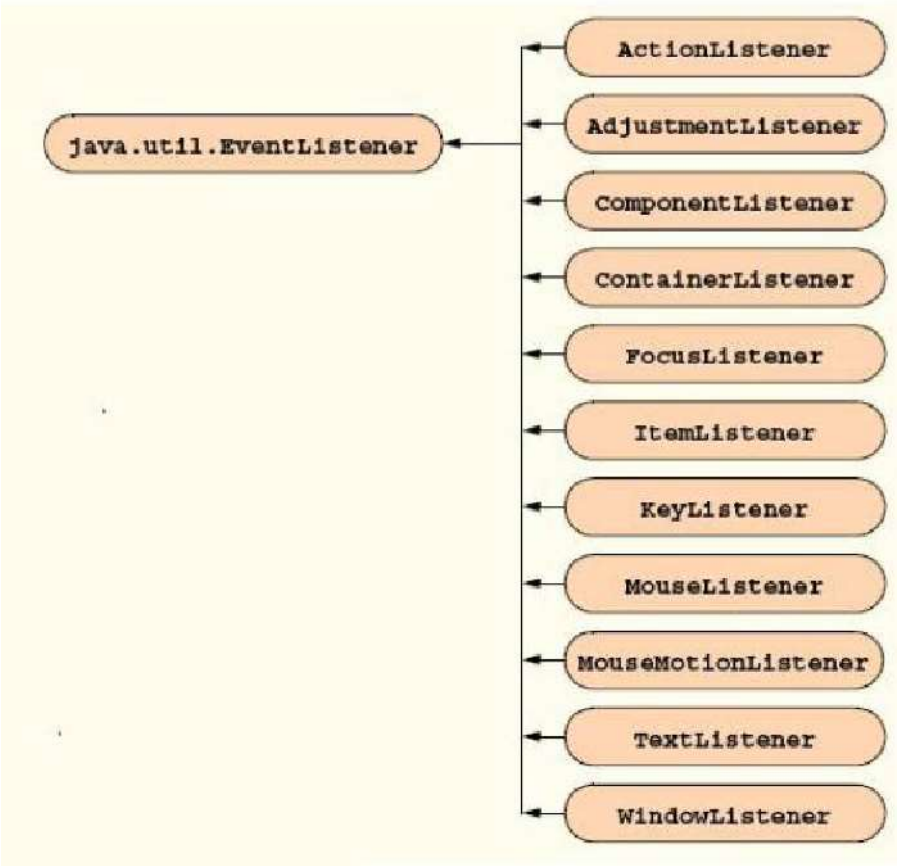
Các đối tượng trong xử lý sự kiện

- 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

Mô hình xử lý sự kiện

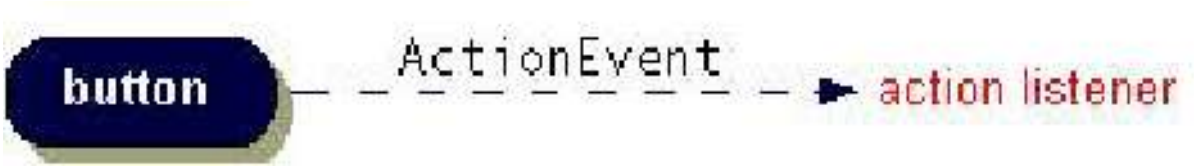
- Lớp hiện thực giao tiếp bộ lắng nghe sự kiện (bộ xử lý sự kiện).
  - Ví dụ: class Circle extends JFrame implements ActionListener {…}
- Đăng ký bộ lắng nghe sự kiện cho nguồn sự kiện
  - Ví dụ: btCancel.addActionListener(handler);
- Cài đặt phương thức xử lý sự kiện (các phương thức của giao tiếp bộ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

Một số bộ lắng nghe sự kiện



Ví dụ

- Một ActionListener được hiện thực và đối tượng lắng nghe được đăng ký với một thành phần JButton.
- Khi nút được nhấn, một sự kiện tự động được phát sinh và phương thức tương ứng cài đặt trong ActionListener được gọi (actionPerformed)



Ví dụ

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

public class MyGUIEvent extends JFrame
                        implements ActionListener
{
    JButton btOk;
    public MyGUIEvent(){
        ....
        btOk.addActionListener(this);
        ....
    }
    public void actionPerformed(ActionEvent e){
        // Todo
    }
}
```

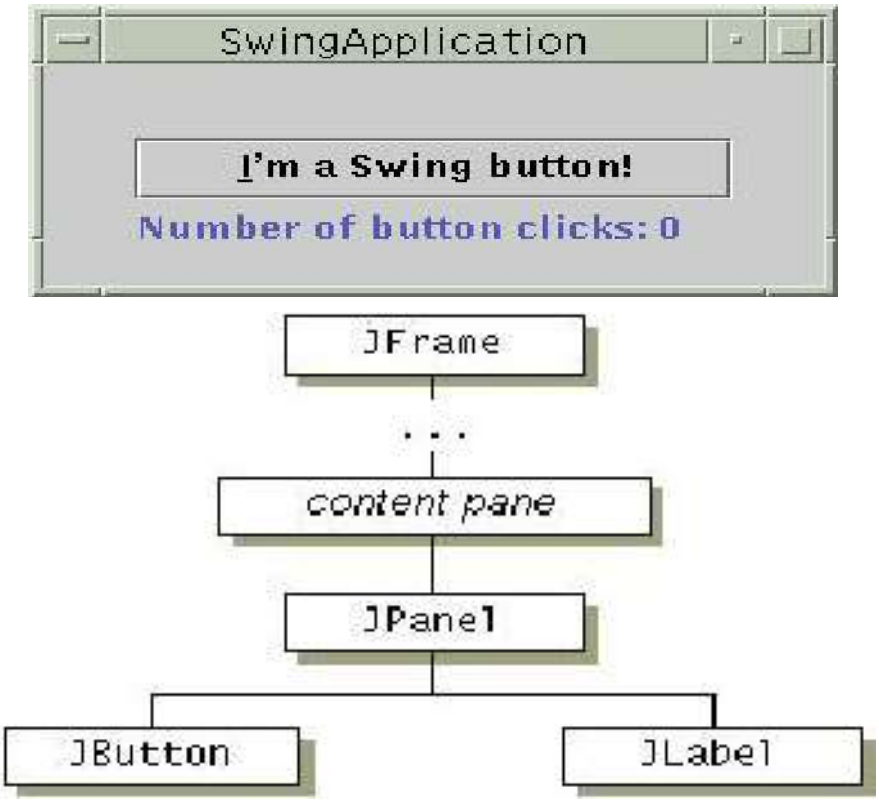
Ví dụ

```
import javax.swing.*;
import javax.awt.event.*;
public class MyGUIEvent extends JFrame
{
    JButton btOk;
    public MyGUIEvent(){
        ....
        btOk.addActionListener( new ActionListener() {
            public void actionPerformed(ActionEvent e){
                // Todo
            }
        });
        ....
    }
}
```

Painting

- Các component có thể cần hiển thị lại mình trên giao diện vì các lý do sau:
  - Thiết lập các trạng thái khác với mặc định
  - Phản ứng với các tương tác khác
- Các repaint: bắt đầu repaint với thành phần cao nhất cần repaint đi xuống cho tới hết cây phân cấp thành phần.
- Các thành phần thường sẽ repaint mỗi khi cần thiết
  - VD: khi bạn gọi setText()

Ví dụ

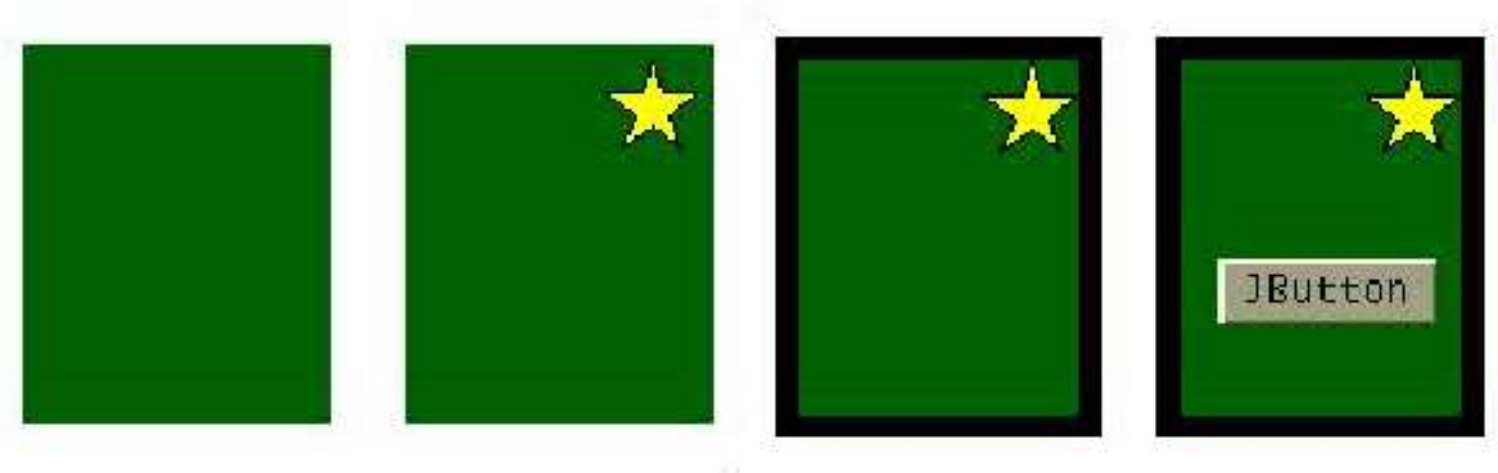

SwingApplication
I'm a Swing button!
Number of button clicks: 0
JFrame
. . .
content pane
JPanel
JButton
JLabel

Ví dụ

- JFrame sẽ repaint đầu tiên
- Content pane sẽ repaint background của nó rồi báo cho JPanel vẽ lại
- JPanel repaint background của nó sau đó vẽ lại đường biên ➔ báo cho các thành phần con vẽ lại
- JButton sẽ vẽ lại background của nó rồi sửa lại đoạn text mà nó chứa
- JLabel sẽ repaint đoạn text của nó.

Tuần tự triệu gọi của repaint

1. Background 2. Custom 3. Border 4. Children



Một số tính năng đặc biệt

- Trừ `top-level containers`, các thành phần bắt đầu bằng chữ *J* đều có 1 số tính năng đặc biệt sau:
  - Tooltips
  - Border
  - Look and feel

JComponent: tool tips

- Bằng cách dùng phương thức `setToolTipText`, bạn có thể cung cấp các gợi ý trợ giúp cho người sử dụng
- Khi con trỏ lướt qua vùng của component➔ `tooltip` sẽ hiển thị

JComponent: border

- Phương thức `setBorder` cho phép người sử dụng chỉ định đường biên xung quanh component
- Bạn có thể sử dụng lớp `BorderFactory` để tạo ra 1 số border thường gặp


```
JPanel pane = new JPanel();
pane.setBorder(BorderFactory.createLineBorder(Color.black));
```

JComponent: look and feel

- Việc hiển thị của các component phụ thuộc vào `ComponentUI` bên dưới
- Bạn có thể dùng phương thức `UIManager.setLookAndFeel` để thay đổi cách hiển thị của các thành phần.

```
public static void main(String[] args) {
    try { UIManager.setLookAndFeel(
        UIManager.getCrossPlatformLookAndFeelClassName());
    }
    catch (Exception e) { }
    new SwingApplication(); //Create and show the GUI.
}
```

JComponent: look and feel

- Bạn cũng có thể dùng các "`look and feel`" của các hệ nền khác sử dụng cú pháp sau

```
UIManager.setLookAndFeel(
    "com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
```

- Nếu bạn thiết lập "`look and feel`" trước khi bất cứ thành phần UI nào được tạo ra thì chương trình sẽ cố gắng thiết lập "`look and feel`" theo thông số bạn truyền vào
  - Nếu không được sẽ lấy giá trị mặc định trong file `swing.properties`

JComponent: look and feel

- Nếu bạn thiết lập lại "`look and feel`" sau khi đã có thành phần UI tạo ra thì bạn sẽ làm như sau để cập nhật "`look and feel`" cho các thành phần nay:

```
UIManager.setLookAndFeel(lnfName);
SwingUtilities.updateComponentTreeUI(frame);
```

Summary