

BÀI GIẢNG JAVA

BÙI MINH CƯỜNG & NGUYỄN TRỌNG PHÚC

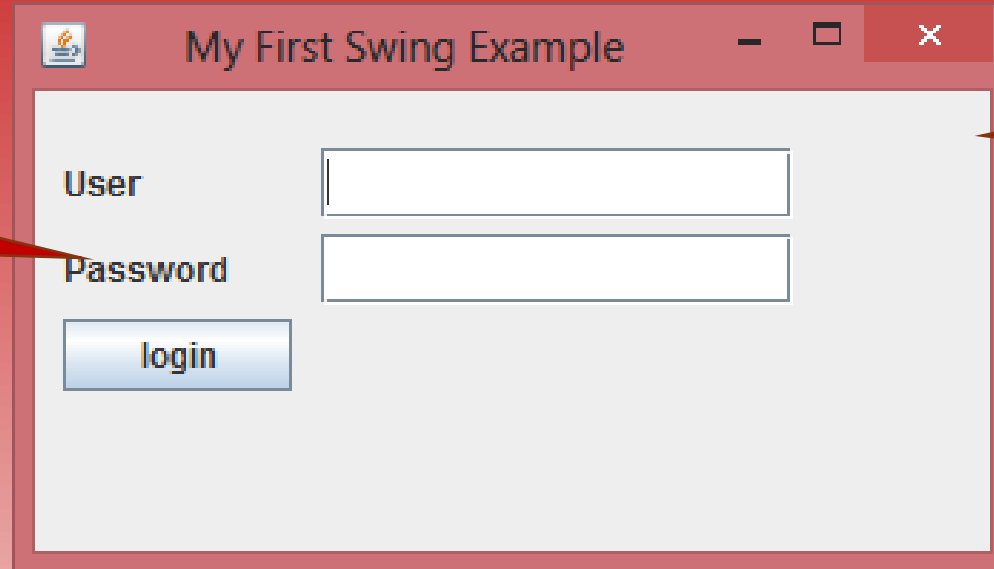
Khoa CNTT - ĐẠI HỌC GTVT

GUI Và Event

Giới thiệu

Khái niệm

component



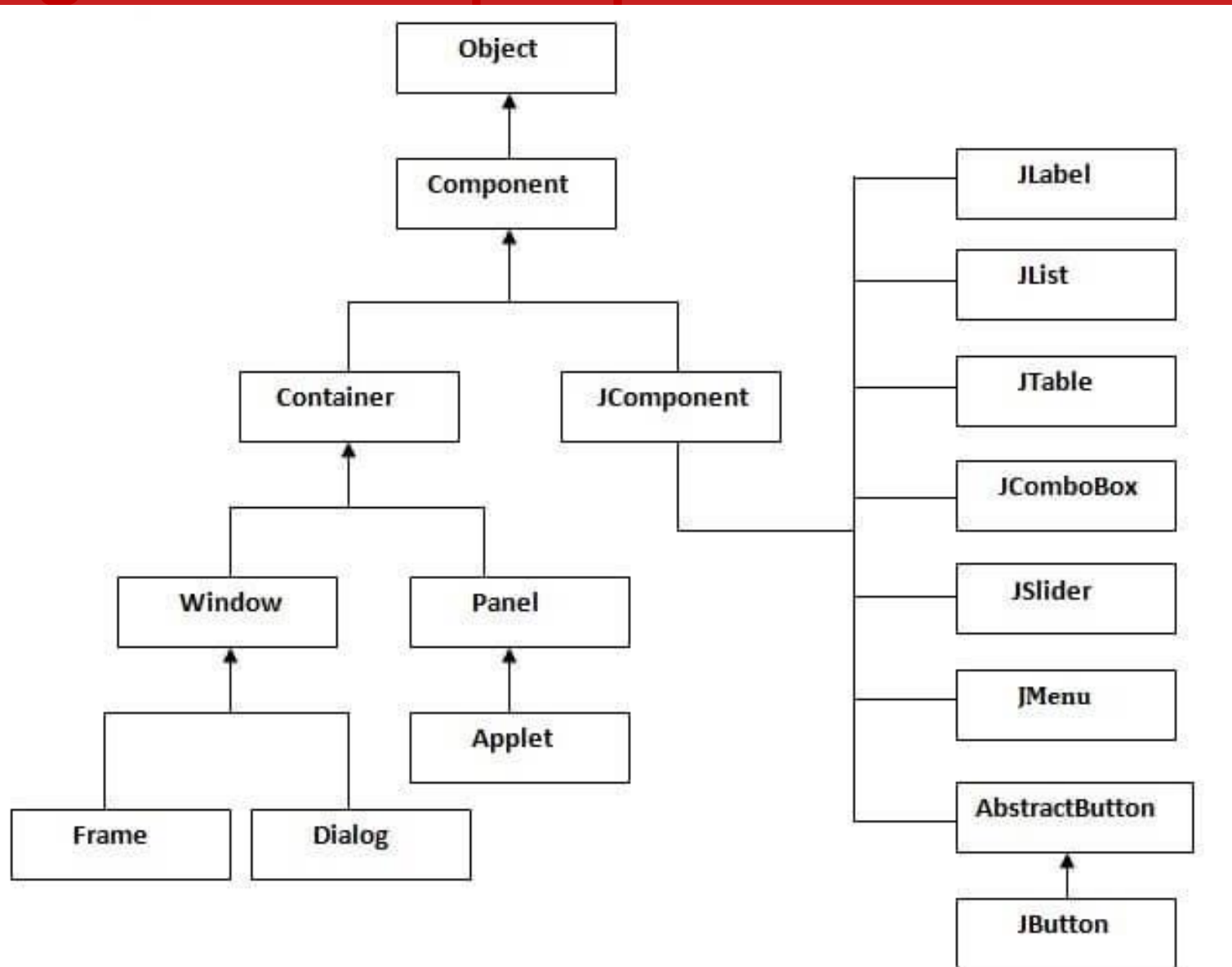
container

GUI = Containers + Components

Java Swing

- Java Swing là một phần của Java Foundation Classes (JFC) được sử dụng để tạo các ứng dụng window-based. Nó được xây dựng trên API AWT (Abstract Windowing Toolkit) và được viết hoàn toàn bằng Java.
- Không giống như AWT, Java Swing cung cấp các thành phần không phụ thuộc vào nền tảng và nhẹ hơn.
- Gói javax.swing cung cấp các lớp cho java swing API như JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser///

Java swing – Phân cấp lớp



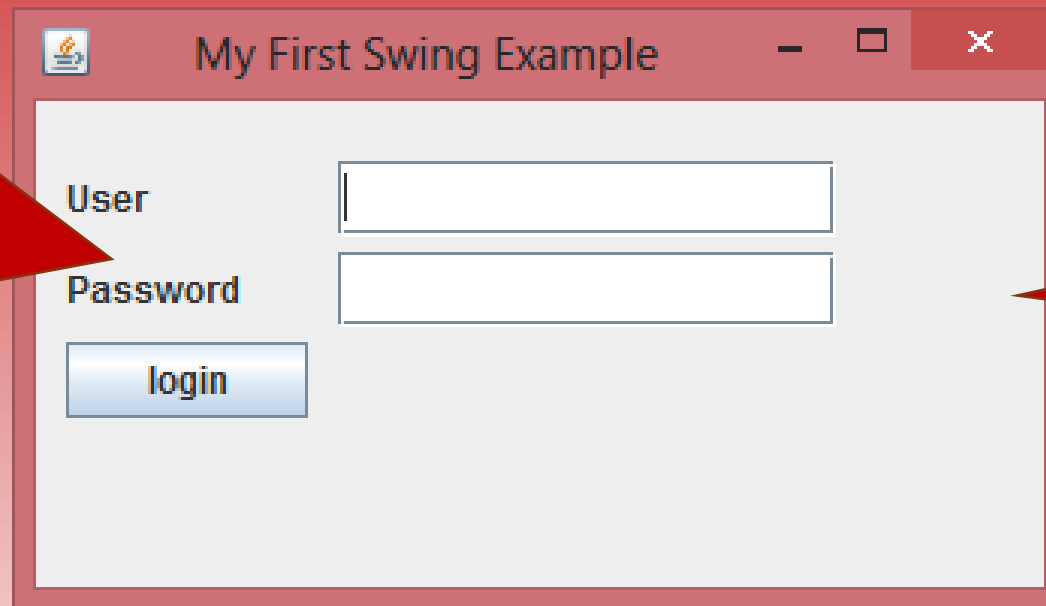
Đưa component vào GUI

- Tạo 1 đối tượng component phù hợp.
- Xác định hình thức bên ngoài lúc đầu của component.
- Định vị component này trên GUI.
- Thêm component này vào GUI.

Đưa component vào GUI

- Ví dụ

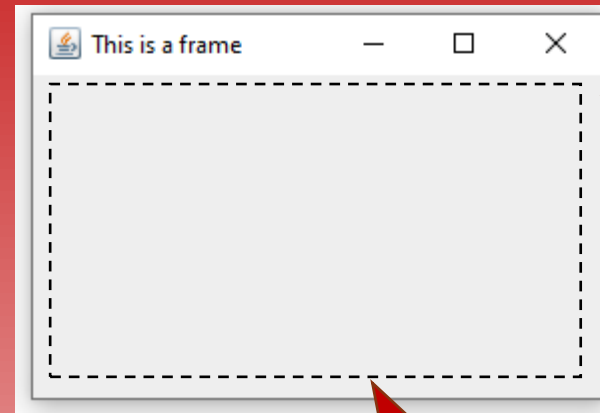
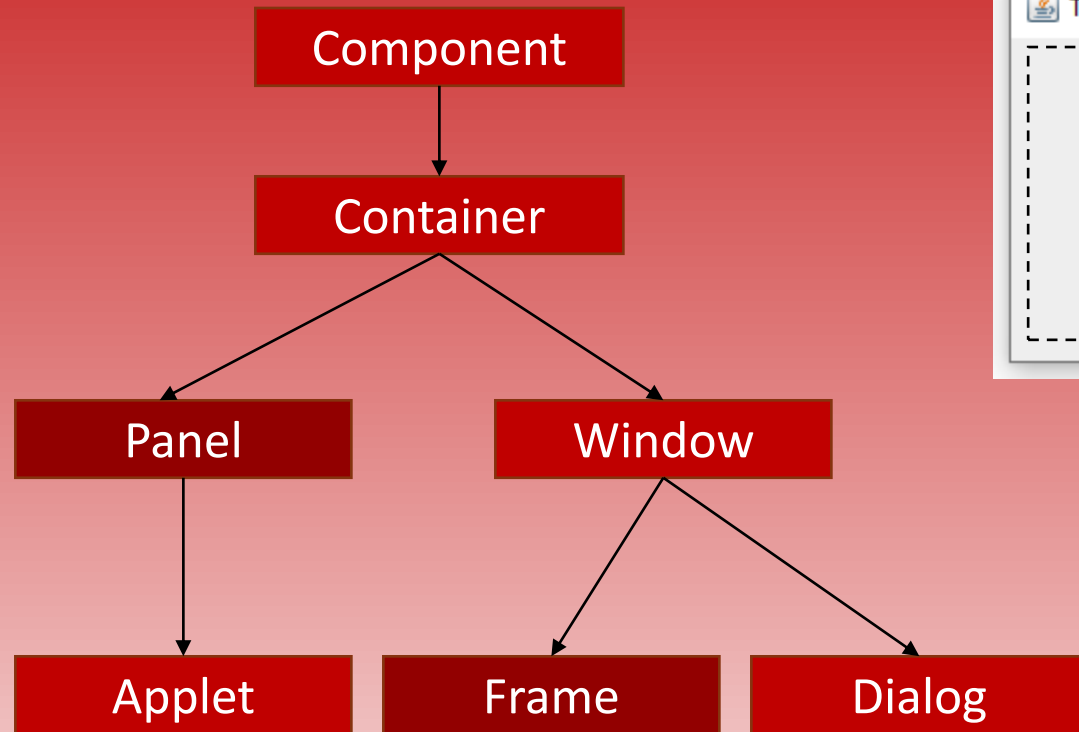
2 label
2 text-field
1 button



container

Container

Phân cấp thừa kế



frame

Frame là khung chữ nhật có đường viền và có nút điều khiển cửa sổ

Panel

Panel không có đường viền

Khái niệm

- Container: Đối tượng chứa các element, cho phép vẽ, tô màu lên container.
- **Frame** và **Panel** là các class thường dùng.
- **Panel** thường dùng để chứa các element trong 1 GUI phức tạp, 1 **Frame** có thể chứa nhiều Panel.
- Applet thường dùng để tạo 1 ứng dụng nhúng vào Browser.

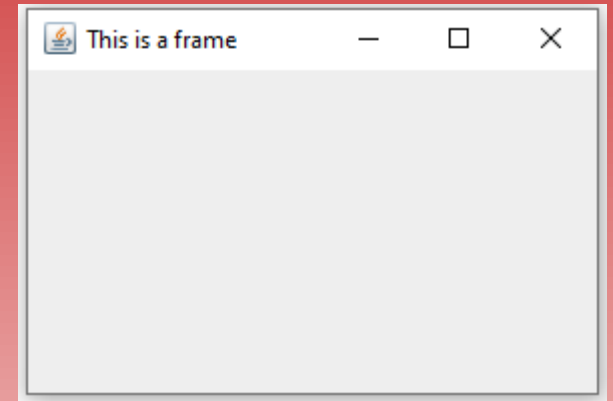
Ví dụ

- Tạo frame:

```
package container;

import javax.swing.JFrame;

public class ContainerDemo {
    public static void main(String[] args) {
        JFrame jframe1 = new JFrame();
        jframe1.setSize(300, 200);
        jframe1.setTitle("This is a frame");
        jframe1.setVisible(true);
    }
}
```



Ví dụ

- Tạo frame và thêm nút:

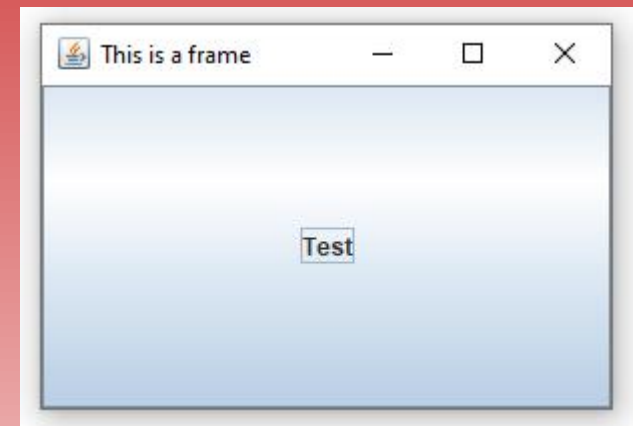
```
package container;

import javax.swing.JButton;
import javax.swing.JFrame;

public class ContainerDemo {
    public static void main(String[] args) {
        JFrame jframe1 = new JFrame();

        JButton jbutton1 = new JButton("Test");
        jframe1.add(jbutton1); // thêm button vào JFrame

        jframe1.setSize(300, 200);
        jframe1.setTitle("This is a frame");
        jframe1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ;
        jframe1.setVisible(true);
    }
}
```



Layout

Khái niệm

- Layout : Cách bố trí các components lên container.
- Không dễ dàng gì để tự quản lý vị trí của các components trên GUI.
- **LayoutManager** là interface mô tả về các layout.
- Java cung cấp sẵn một số layout, các lớp layout này đều implement **LayoutManager** interface.

BorderLayout

- Bố trí các component theo dạng ra biên của khung tạo ra 5 vị trí: EAST, WEST, SOUTH, NORTH, CENTER.
- Đây là layout **MẶC ĐỊNH** của **Frame**.
- Nếu container chỉ có 1 component và đặt nó ở vị trí CENTER thì component này phủ kín container.
- Cú pháp thêm 1 component vào container tại 1 vị trí:

```
Container.add("East", component);  
Container.add(BorderLayout.EAST, component);
```

- Tương tự cho “West”, “South”, “North”, “Center”

BorderLayout

```
package container;

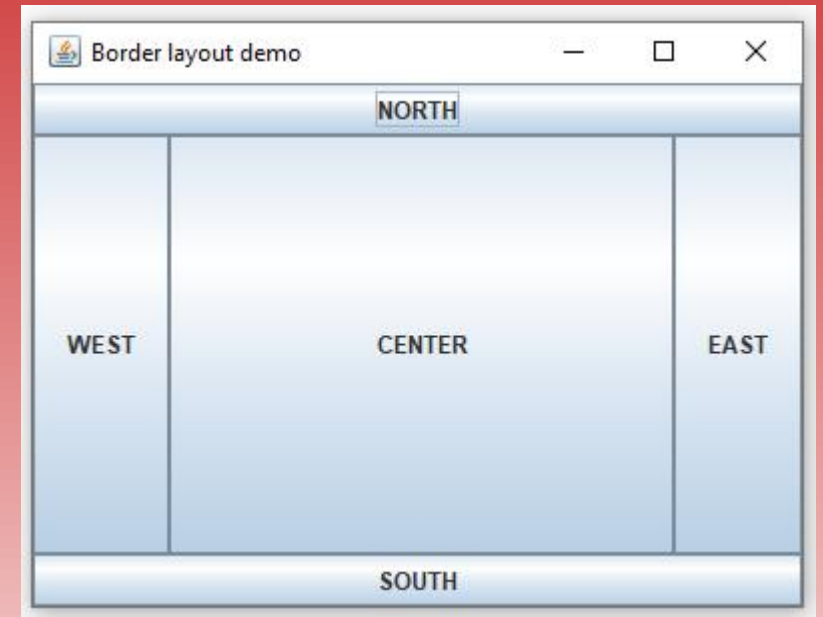
import java.awt.BorderLayout;

import javax.swing.JButton;
import javax.swing.JFrame;

public class BorderLayoutDemoFrame extends JFrame{
    public BorderLayoutDemoFrame() {
        this.getContentPane().add(BorderLayout.NORTH, new JButton("NORTH"));
        this.getContentPane().add(BorderLayout.SOUTH, new JButton("SOUTH"));
        this.getContentPane().add(BorderLayout.CENTER, new JButton("CENTER"));
        this.getContentPane().add(BorderLayout.WEST, new JButton("WEST"));
        this.getContentPane().add(BorderLayout.EAST, new JButton("EAST"));
    }

    public static void main(String[] args) {
        BorderLayoutDemoFrame frame = new BorderLayoutDemoFrame();

        frame.setSize(400, 300);
        frame.setTitle("Border layout demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



FlowLayout

- Bố trí các component theo dạng chảy xuôi theo thứ tự mà phần tử này được add vào container.
- **Đây là layout mặc định của Panel.**
- Nếu có nhiều component trên container -> Các component có thể được đặt trên nhiều dòng -> Vấn đề giống hàng (Align)
- Giữa các component, chúng hở nhau theo chiều dọc (vgap) bao nhiêu, theo chiều ngang (hgap) bao nhiêu?

FlowLayout

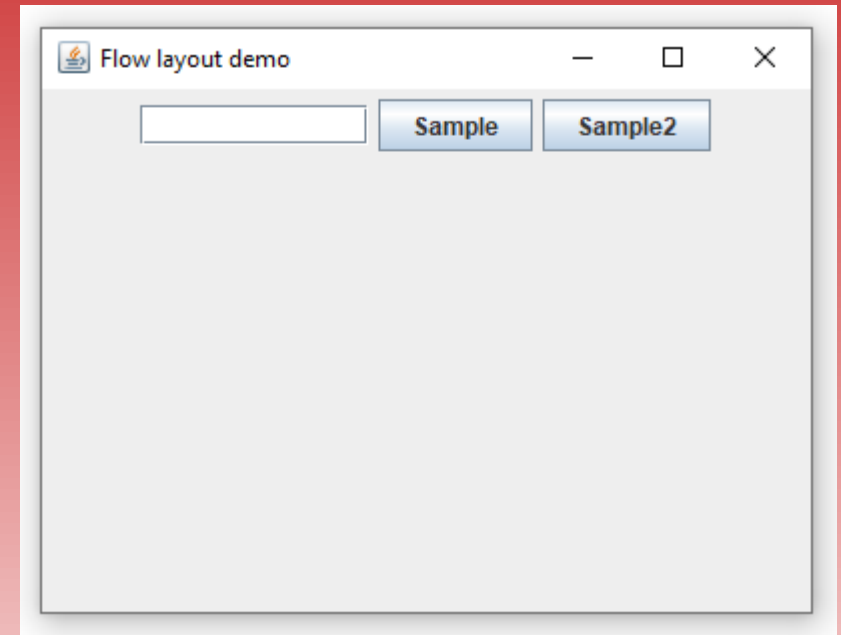
```
package container;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class FlowLayoutDemo extends JFrame{
    public FlowLayoutDemo() {
        JPanel panel1 = new JPanel();
        panel1.add(new JTextField(10));
        panel1.add(new JButton("Sample"));
        panel1.add(new JButton("Sample2"));
        getContentPane().add(panel1);
    }

    public static void main(String[] args) {
        FlowLayoutDemo frame = new FlowLayoutDemo();

        frame.setSize(400, 300);
        frame.setTitle("Flow layout demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



GridLayout

- Bố trí các component thành 1 lưới rows dòng, cols cột đều nhau.

```
package container;

import java.awt.GridLayout;

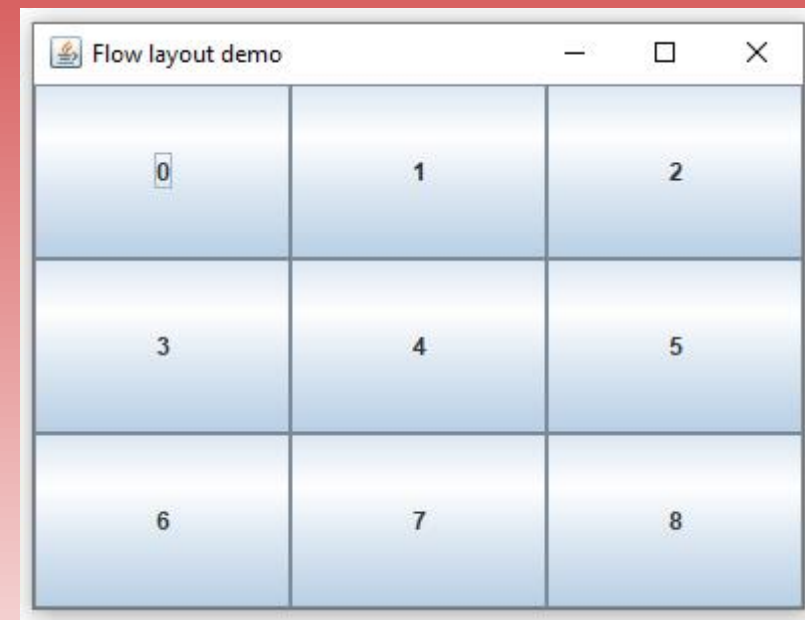
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class GridLayoutDemo extends JFrame {
    public GridLayoutDemo() {
        JPanel panel1 = new JPanel();
        panel1.setLayout(new GridLayout(3, 3));

        for (int i = 0; i < 9; i++) {
            panel1.add(new JButton(String.valueOf(i)));
        }
        getContentPane().add(panel1);
    }

    public static void main(String[] args) {
        GridLayoutDemo frame = new GridLayoutDemo();

        frame.setSize(400, 300);
        frame.setTitle("Flow layout demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



GridBagLayout

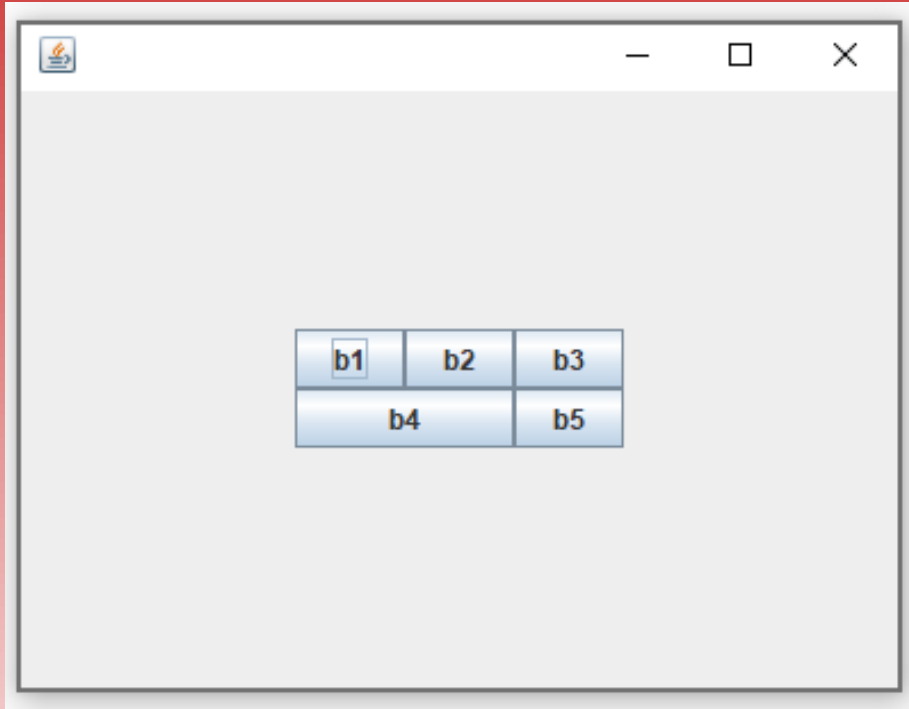
- Layout dạng lưới cho phép 1 component chiếm 1 số ô kề nhau theo cả 2 chiều.
- Cách để đưa component vào GridBagLayout:
 - 1 component vào 1 vị trí nhưng trải dài trên nhiều ô kề nhau là 1 sự “ràng buộc” 1 component vào các ô này.
 - Một đối tượng thuộc lớp **GridBagConstraints** sẽ đảm nhiệm việc này.

GridBagLayout - GridBagConstraints

Properties – Đa số là static data

- **gridx, gridy** : ô sẽ đặt component vào
- **gridwidth, gridheight** : số ô sẽ phủ theo 2 chiều khi thêm 1 component vào ô <gridx,gridy>
- **weightx, weighty**: Khoảng hở của lưới, mặc định là 0.
- **anchor**: Vị trí đặt component, mặc định là CENTER, các hằng khai báo sẵn:
GridBagConstraints.NORTH, EAST, WEST, SOUTH, NORTHEAST, SOUTHEAST, NORTHWEST, SOUTHWEST.
- **fill**: Xác định kiểu đặt khi component có kích thước lớn hơn ô sẽ được đặt vào. Các hằng khai báo sẵn: GridBagConstraints.NONE, HORIZONTAL, VERTICAL, BOTH.
- **insets**: Đặc tả khoảng hở <top, bottom, left, right> giữa các phần tử được đưa vào, mặc định là 0.
- **ipadx, ipady**: Khoảng đệm (số pixel trống) bên trong của phần tử theo 2 chiều. Mặc định là 0. Khi vẽ phần tử, sẽ thêm 2*ipadx và 2*ipady vào chiều rộng tối thiểu và chiều cao tối thiểu của phần tử.

GridBagLayout



```
package container;

import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;

import javax.swing.JButton;
import javax.swing.JFrame;

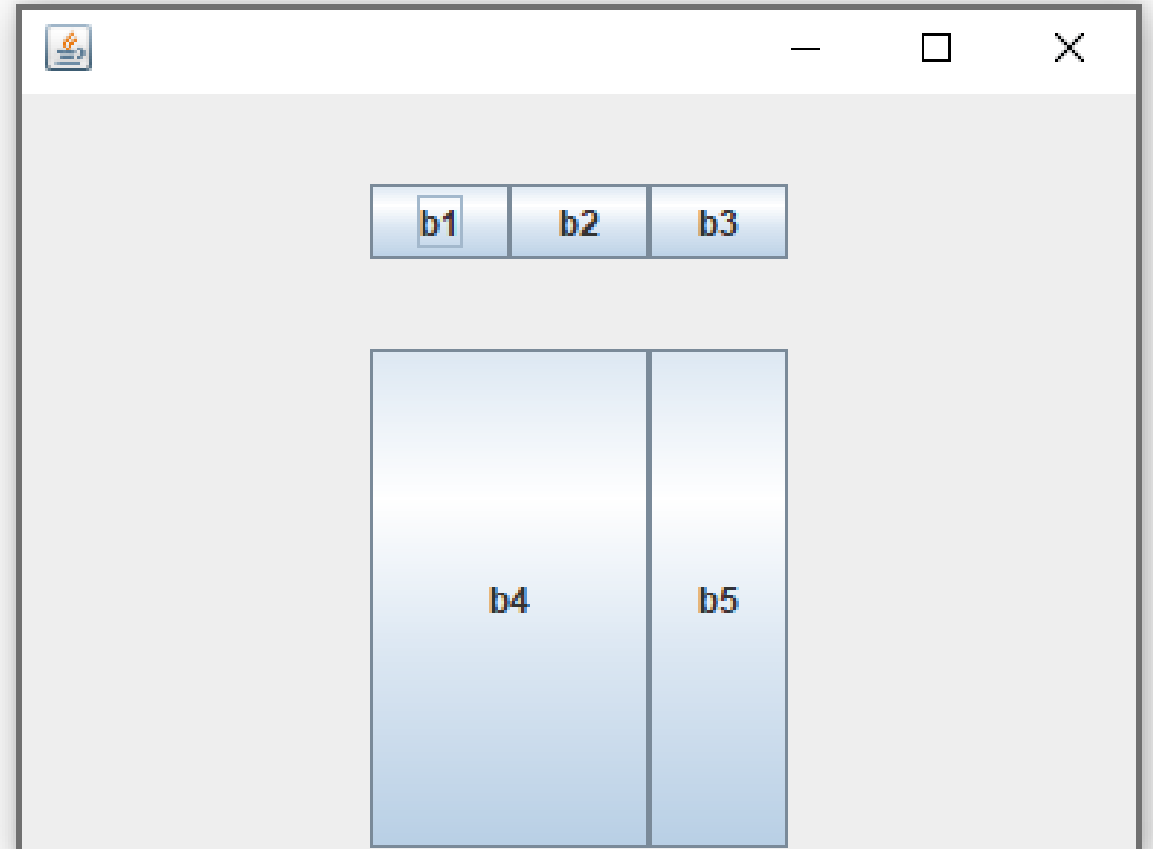
public class GridBagLayoutDemo extends JFrame{
    public GridBagLayoutDemo() {
        setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints();
        c.gridx = 0;
        c.gridy = 0;
        add(new JButton("b1"), c);
        c.gridx = 1;
        add(new JButton("b2"), c);
        c.gridx = 2;
        add(new JButton("b3"), c);
        c.gridx = 0;
        c.gridy = 1;
        c.gridwidth = 2;
        c.fill = GridBagConstraints.BOTH;
        add(new JButton("b4"), c);
        c.gridx = 2;
        c.gridwidth = 1;
        add(new JButton("b5"), c);
        setSize(400, 300);
        //hien thi giua man hinh
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    public static void main(String[] args) {
        GridBagLayoutDemo frame1 = new GridBagLayoutDemo();
        frame1.setVisible(true);
    }
}
```

GridBagLayout

- Để lưới chiếm hết cả không gian của container cần set **weightx** và **weighty** > 0
- **weightx** và **weighty** có tỉ lệ tương đối giữa các hàng/cột

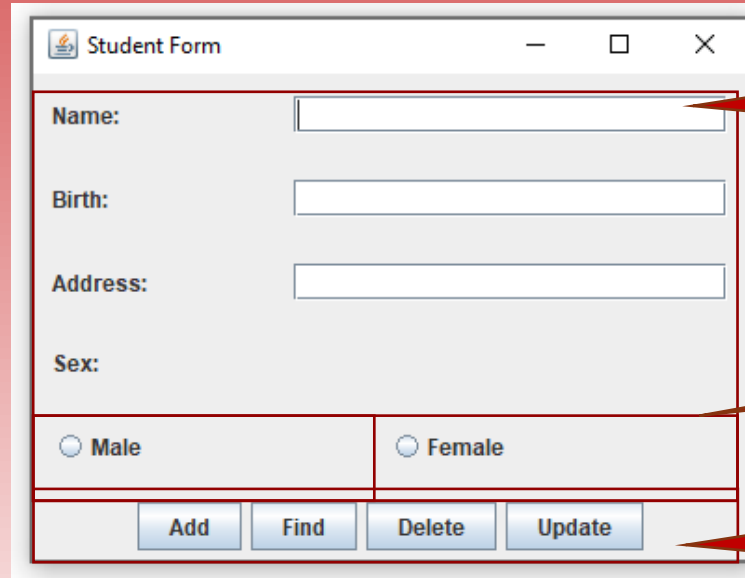
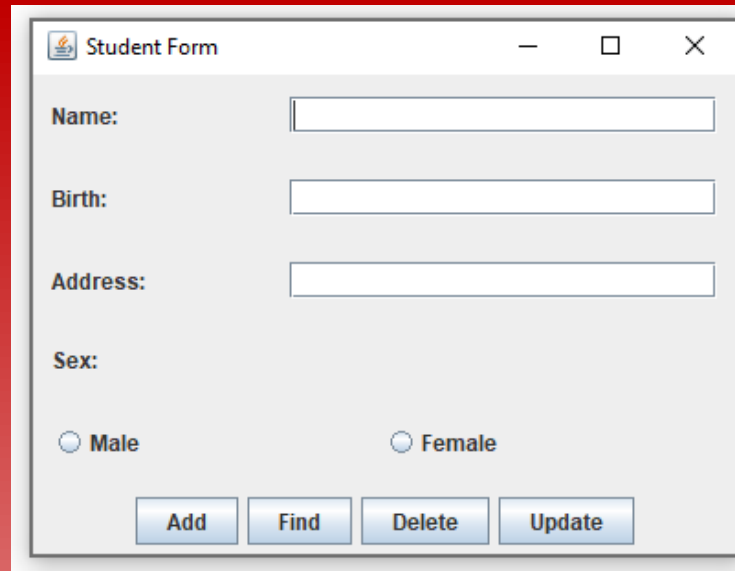
```
...
GridBagConstraints c = new GridBagConstraints();
c.gridx = 0;
c.gridy = 0;
c.weighty = 0.3;
add(new JButton("b1"), c);
c.gridx = 1;
add(new JButton("b2"), c);
c.gridx = 2;
add(new JButton("b3"), c);
c.gridx = 0;
c.gridy = 1;
c.gridwidth = 2;
c.weighty = 0.7;
c.fill = GridBagConstraints.BOTH;
add(new JButton("b4"), c);
c.gridx = 2;
c.gridwidth = 1;
add(new JButton("b5"), c);
...
```



Bài tập

- Thực hiện layout sau
- Gợi ý:
 - 2 Panel (Center + South của Frame)
 - Panel trên sử dụng GridBagLayout
 - Chú ý cần set **weightx** và **weighty** để chiếm hết cả frame
 - Hàng chứa radio button thêm 1 panel và set thành GridLayout

- Bài chữa ->



GridBagLayout

GridLayout

BorderLayout.South

Lập trình hướng sự kiện

Event

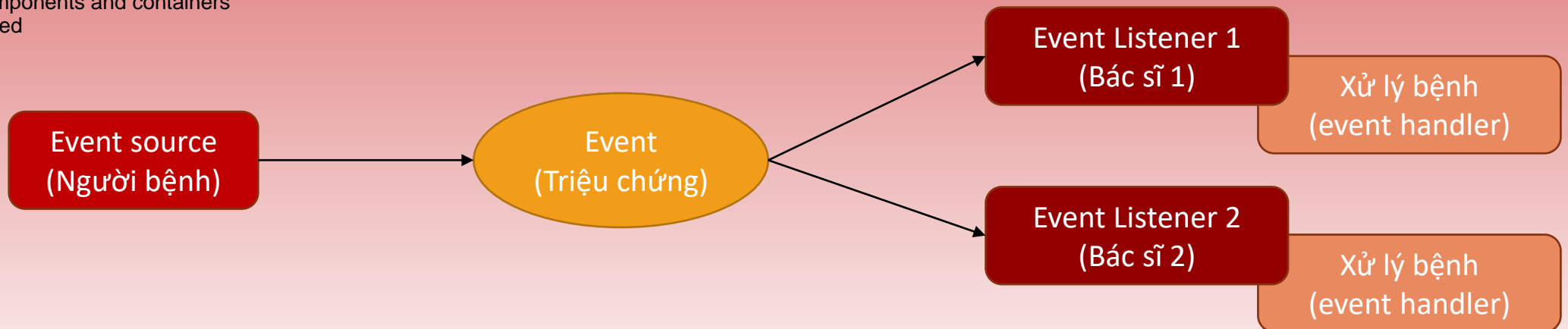
Whenever actives, all events start working. DO not running 2 events at one time, causing disruption.

- **Event** : một tín hiệu mà ứng dụng nhận biết có sự thay đổi trạng thái của 1 đối tượng.
- 3 nguồn phát xuất event:
 - (1) User(gõ phím, kích chuột vào 1 phần tử,...)
 - (2) Do hệ thống (do định thời 1 tác vụ)
 - (3) Do 1 event khác (các event kích hoạt nhau)
- Hiện nay, đa số các ngôn ngữ đều cung cấp mô hình này, VC++ cung cấp MFC (Microsoft Foundation Classes), Java cung cấp JFC (Java Foundation Classes).

Ủy thác xử lý sự kiện

- Khi một sự kiện xảy ra, việc xử lý tùy thuộc vào người được ủy thác sự kiện
- Ví dụ:
 - Khi ta (event source) bị bệnh sẽ phát sinh triệu chứng (event)
 - Ta có thể đến khám nhiều bác sĩ (listener)
 - Mỗi bác sĩ sẽ xử lý bệnh (event handler) theo cách khác nhau dựa trên triệu chứng (event data)

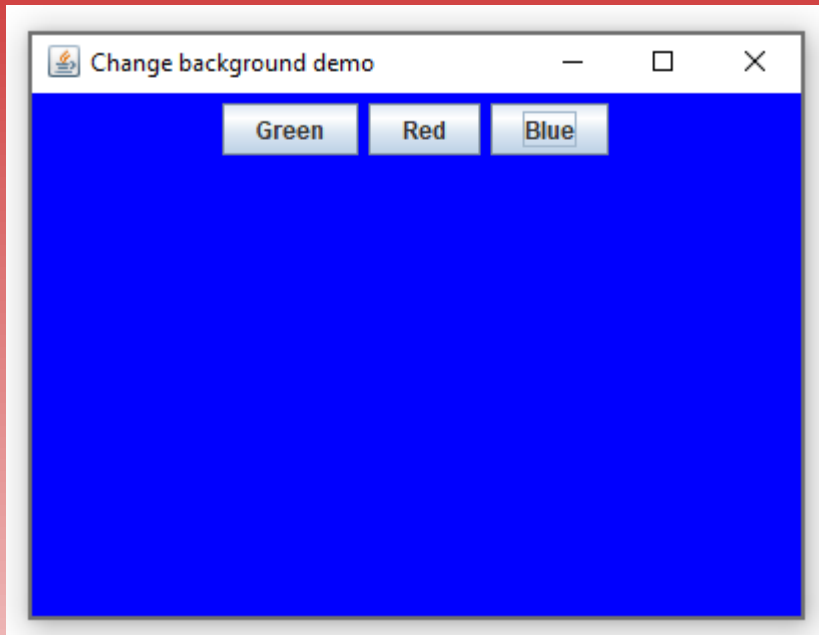
Whenever event happens, first step would be categorized events.
Event Source: components and containers
EVENT: categorized



Một số khái niệm

- **Event** : Là sự kiện phát sinh khi có 1 đối tượng đã thay đổi trạng thái.
- **Event handler**: Là đoạn code để đạt phản ứng của ứng dụng khi gặp 1 **event**.
- **Event source**: Đối tượng kích hoạt event (thí dụ: nút lệnh bị user kích chuột).
- **Listener** : Đối tượng nhận sự ủy nhiệm xử lý sự kiện cho đối tượng khác
- Java định nghĩa sẵn các **Listener** Interface cho các tình huống khác nhau (mỗi **Event** object có **Listener** interface xử lý tương ứng).
- Một lớp có khả năng **listener** sẽ phải cụ thể hóa - viết code- một số hành vi xử lý một **event** phù hợp.

Ví dụ



EventDemoFrame.java

```
...
public class EventDemoFrame extends JFrame implements ActionListener {
    JButton buttonGreen = new JButton("Green");
    JButton buttonBlue = new JButton("Blue");
    JButton buttonRed = new JButton("Red");

    public EventDemoFrame() {
        setLayout(new FlowLayout());
        buttonGreen.addActionListener(this);
        add(buttonGreen);
        buttonRed.addActionListener(this);
        add(buttonRed);
        buttonBlue.addActionListener(this);
        add(buttonBlue);

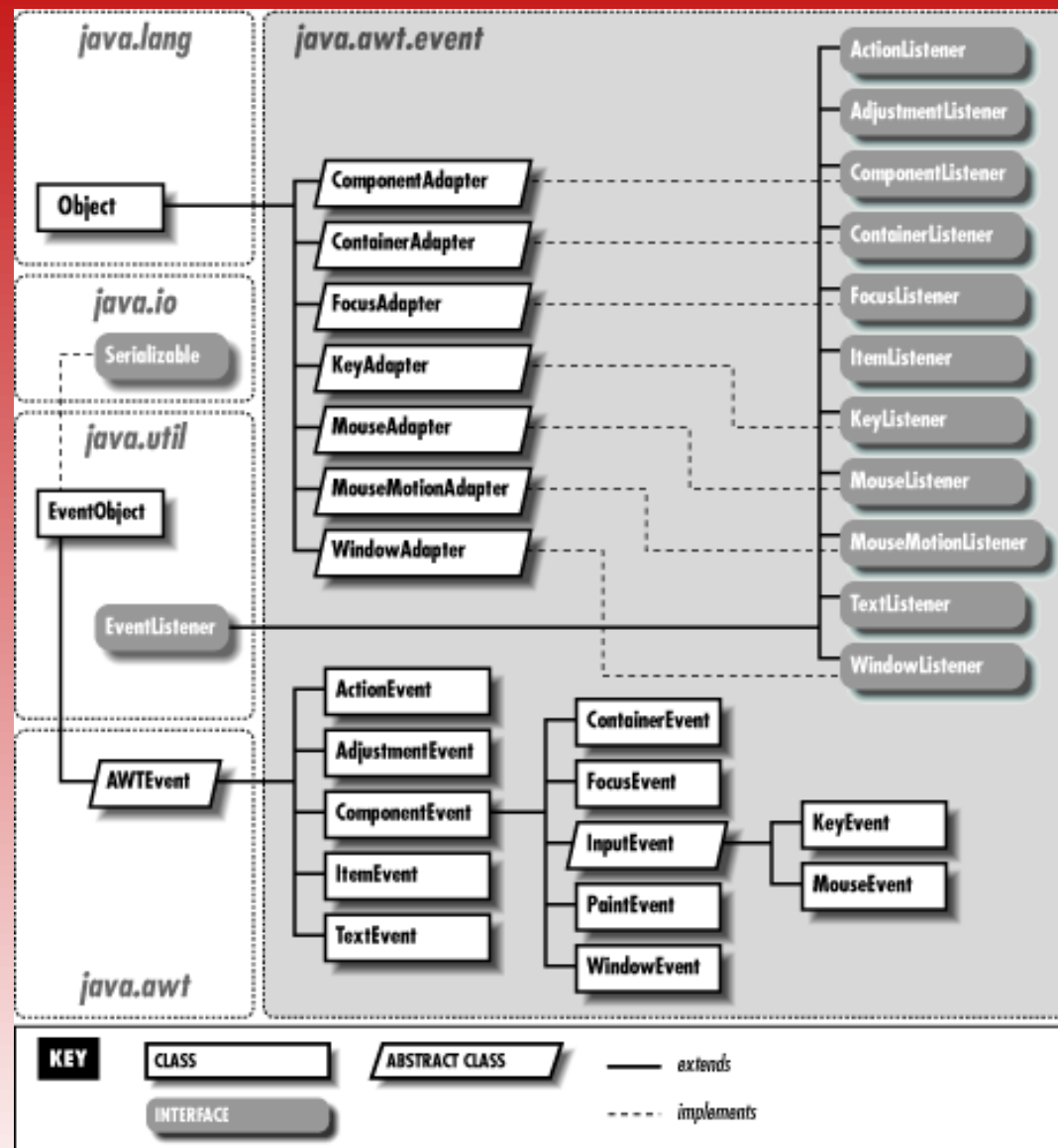
        setSize(400, 300);
        setTitle("Change background demo");
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if(e.getSource().equals(buttonGreen)) {
            getContentPane().setBackground(Color.GREEN);
        }
        else if(e.getSource().equals(buttonBlue)) {
            getContentPane().setBackground(Color.BLUE);
        }
        else if(e.getSource().equals(buttonRed)) {
            getContentPane().setBackground(Color.RED);
        }
    }

    public static void main(String[] args) {
        EventDemoFrame frame = new EventDemoFrame();
        frame.setVisible(true);
    }
}
```

Các event

- Các component sẽ phát sinh ra các **event** khác nhau
 - Mỗi **event** sẽ do 1 listener tương ứng xử lý
 - **Listener** là interface
 - **Adapter** cài đặt mặc định cho các phương thức của Listener.
 - Nếu extends Adapter sẽ không bắt buộc phải cung cấp cài đặt cho tất cả các phương thức của các interface Listener -> giúp tiết kiệm code
- Java thng dùng Listener k dùng Adapter vì Java k có a k tha



Action Event

- Một ActionEvent object được sinh ra khi: 1 nút lệnh bị kích, một mục chọn trong danh sách bị kích đôi, 1 mục menu bị kích.
- Các hằng kiểm tra có 1 phím bị nhấn khi kích chuột hay không: ALT_MASK (phím Alt), CTRL_MASK (phím Ctrl), META_MASK (phím meta, ký tự mô tả về 1 ký tự khác - ký tự escape), SHIFT_MASK (phím Shift).

Adjustment Event

- Được sinh ra khi 1 thanh cuộn bị thao tác.
- Các hằng BLOCK_DECREMENT, BLOCK_INCREMENT: Độ giảm/tăng theo khối khi user kích chuột vào vùng giữa con trượt và 1 biên của thanh cuộn, UNIT_DECREMENT, UNIT_INCREMENT: Đơn vị giảm/tăng khi user kích chuột vào mũi tên ở 2 đầu thanh cuộn. TRACK: Giá trị mô tả thanh cuộn khi bị user kéo

Container Event

- Được sinh ra khi 1 component được thêm/xóa khỏi 1 container.
- Các hằng mô tả sự kiện: `COMPONENT_ADDED`, `COMPONENT_REMOVED`.

Component Event

- Được sinh ra khi 1 componet bị ẩn đi, được hiển thị, bị di chuyển, bị thay đổi kích thước.
- Các hằng mô tả trạng thái gồm: COMPONENT_HIDDEN, COMPONENT_MOVED, COMPONENT_RESIZED, COMPONENT_SHOWN

Input Event

- Là lớp cha của 2 lớp con: KeyEvent và MouseEvent.
- Các hằng khai báo trong lớp này mô tả các bit mặt nạ truy xuất phím đi kèm sự kiện hoặc nút chuột nào bị nhấn: ALT_MASK, CTRL_MASK, META_MASK, SHIFT_MASK, BUTTON1_MASK, BUTTON2_MASK, BUTTON3_MASK.

Key Event

- Được sinh ra khi user thao tác với bàn phím .
- Các hằng kiểu `intKEY_PRESSED`, `KEY_RELEASED`, `KEY_TYPED`. Nếu phím chữ, phím số được gõ, cả 3 loại sự kiện được sinh ra (`pressed`, `released`, `typed`). Nếu phím đặc biệt được thao tác (phím Home, End, PageUp, PageDown- modifier key), chỉ có 2 sự kiện được sinh ra: `pressed`, `released`.

Mouse Event

- Được sinh ra khi user thao tác chuột với 1 component.
- Các hằng int: `MOUSE_CLICKED`, `MOUSE_DRAGGED`, `MOUSE_ENTERED`, `MOUSE_EXITED`, `MOUSE_MOVED`, `MOUSE_PRESSED`, `MOUSE_RELEASED`.

Text Event

- Được sinh ra khi các ký tự trong 1 TextField hay 1 textArea bị đổi.
- Hằng int: TEXT_VALUE_CHANGED

Tài liệu tham khảo

- <https://viettuts.vn/java-swing>
- <https://viettuts.vn/java-awt/cac-lop-adapter-trong-java-awt>
- http://www.netfoo.net/oreilly/java/javanut/figs/jn2_2001.gif