

---

# XÂY DỰNG GIAO DIỆN CHO ỨNG DỤNG JAVA DESKTOP DÙNG AWT VÀ SWING

# Nội dung

---

- Giới thiệu AWT & Swing
- Component, Container, Layout Manager
- Nguyên tắc thiết kế GUI với AWT và Swing
- Một số kiểu Layout phổ biến
- Một số Demo về AWT & Swing components

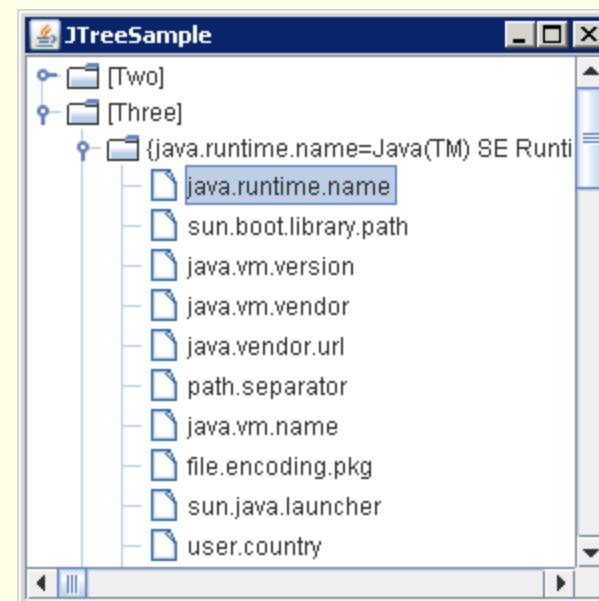
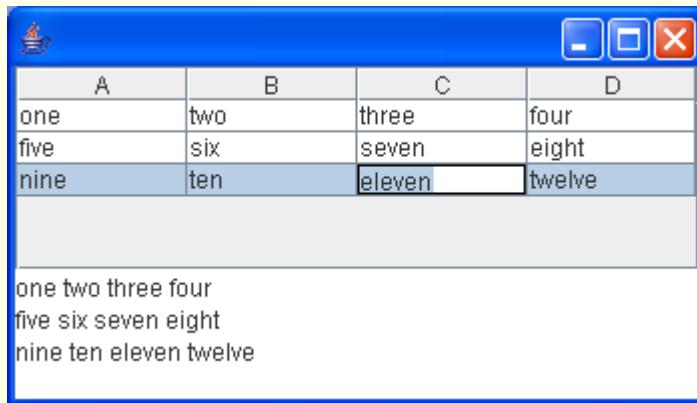
# Giới thiệu AWT & Swing

---

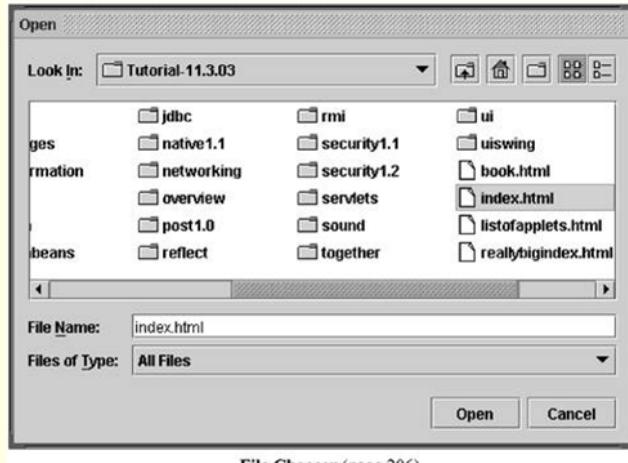
- Thư viện hỗ trợ: tập hợp các lớp java cung cấp hỗ trợ thiết kế, xây dựng GUI.
  - ✓ awt (java.awt.\* ) **Abstract Window Toolkit**
  - ✓ swing (javax.swing.\* )
  - ✓ Các components của các nhà cung cấp thứ 3

# Giới thiệu AWT & Swing (tt)

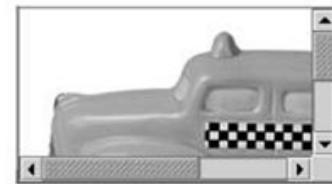
- Swing: phát triển từ AWT và viết hoàn toàn bằng Java.
- Swing cung cấp nhiều components hơn. Ví dụ: tables, lists, scrollpanes, colorchooser, tabbedPane, v.v...



# Giới thiệu AWT & Swing (tt)



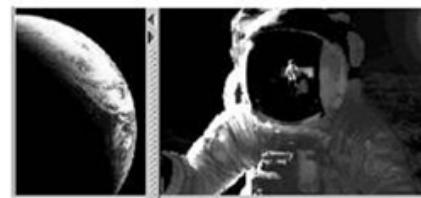
File Chooser (page 206)



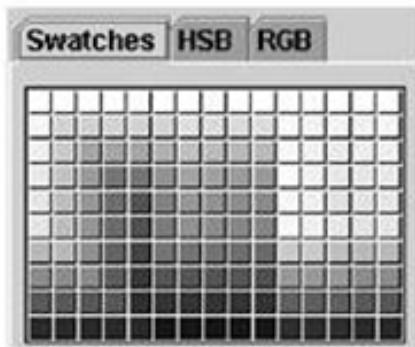
Panel (page 292)



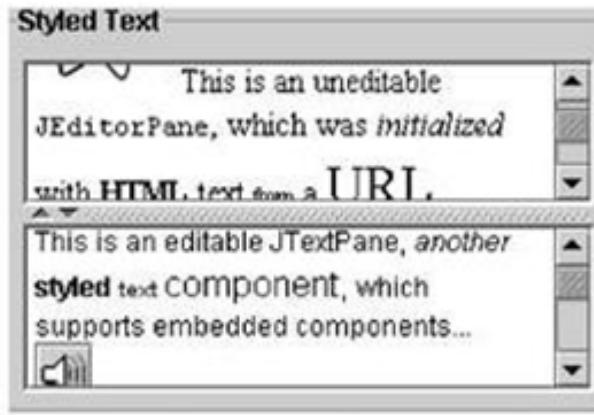
ScrollPane (page 325)



Split Pane (page 369)

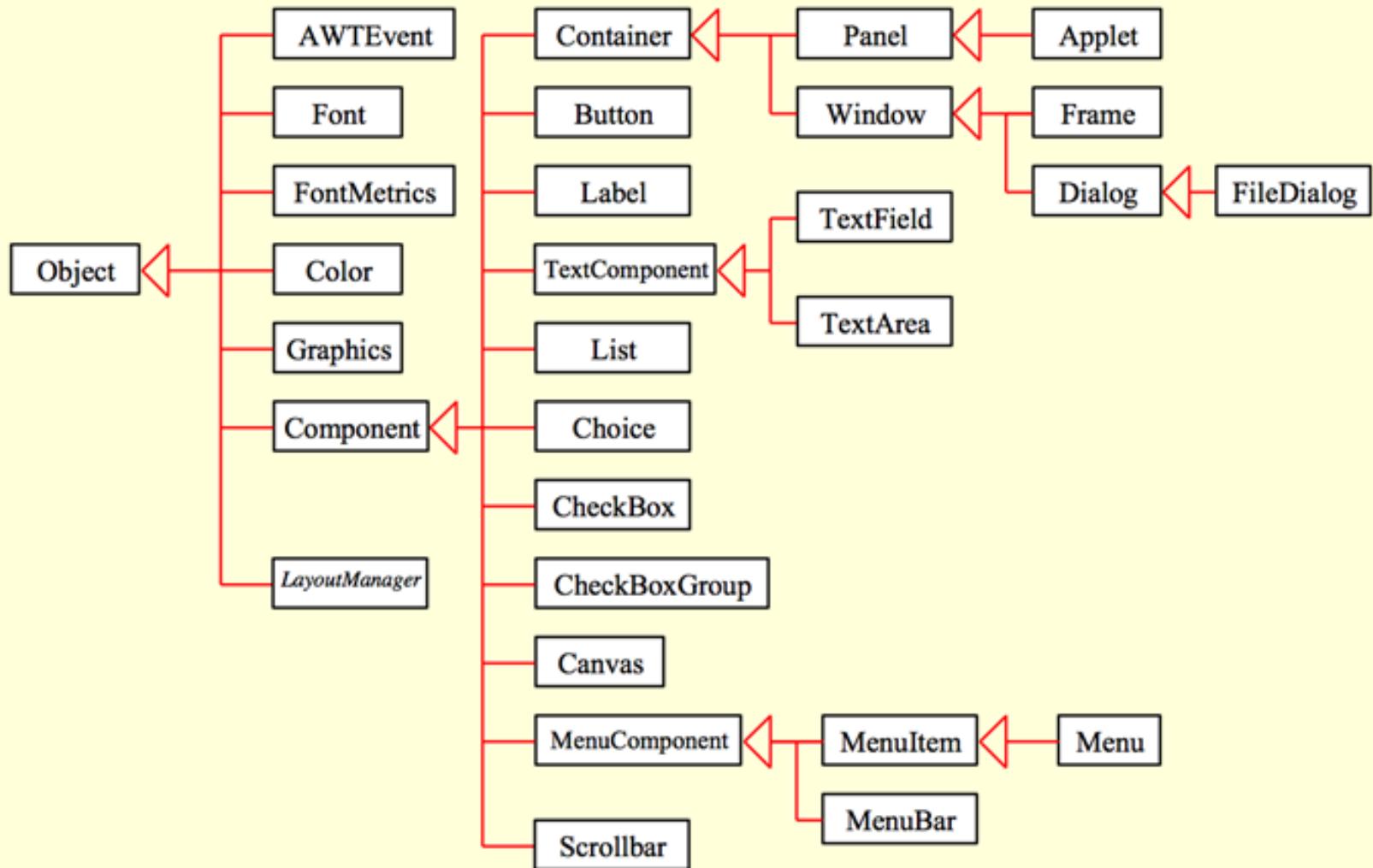


Color Chooser (page 167)

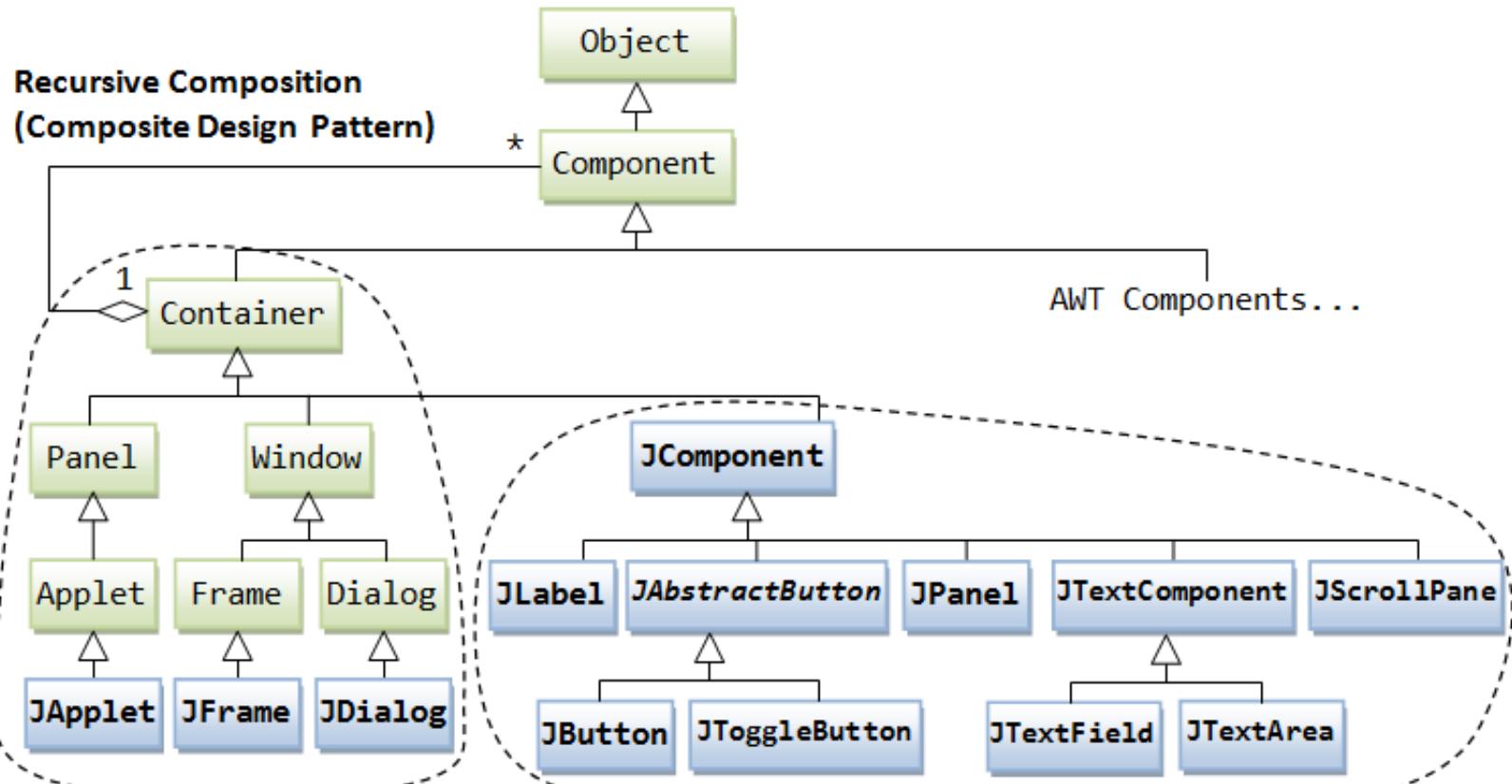


Editor Pane or Text Pane (page 200)

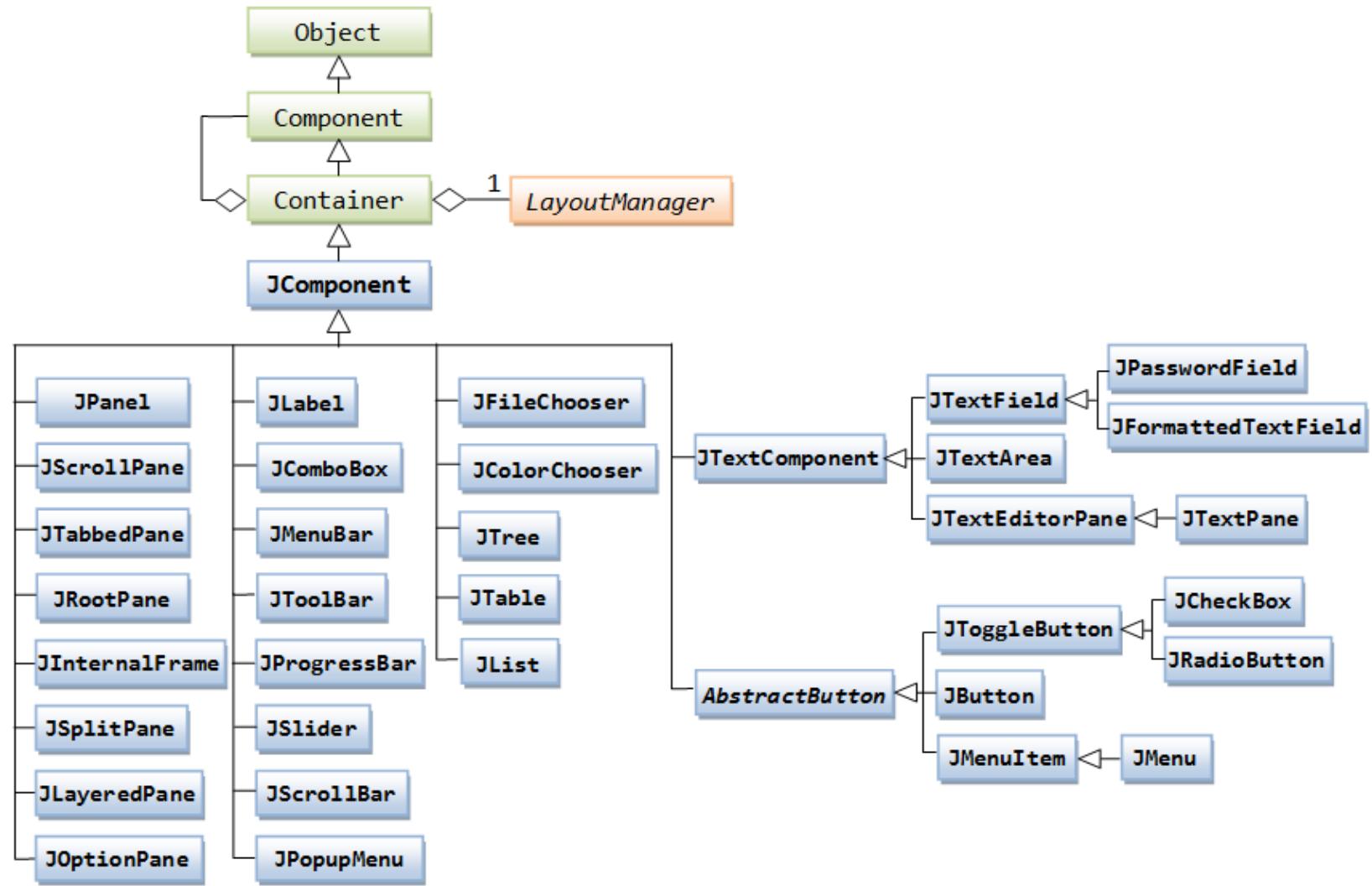
# Giới thiệu AWT & Swing (tt)



# Giới thiệu AWT & Swing (tt)



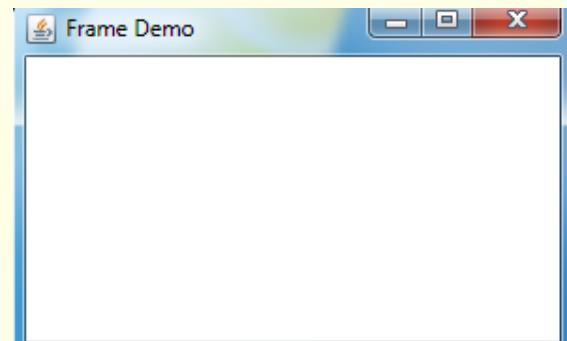
# Giới thiệu AWT & Swing (tt)



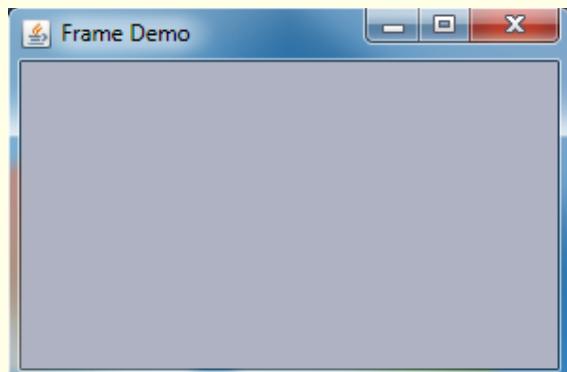
# Giới thiệu AWT & Swing (tt)

- Swing components: hỗ trợ LookAndFeel (AWT không hỗ trợ)

```
UIManager.setLookAndFeel(  
    "com.sun.java.swing.plaf.motif.MotifLookAndFeel");  
Frame fr = new Frame("Frame Demo");  
fr.setSize(300, 200);  
SwingUtilities.updateComponentTreeUI(fr);  
fr.setVisible(true);
```



```
UIManager.setLookAndFeel(  
    "com.sun.java.swing.plaf.motif.MotifLookAndFeel");  
JFrame fr = new JFrame("Frame Demo");  
fr.setSize(300, 200);  
SwingUtilities.updateComponentTreeUI(fr);  
fr.setVisible(true);
```



# Component?

---

- Các thành phần cấu thành GUI của ứng dụng
  - Frame, Window, Dialog, ScrollPane, ...
  - Menu, ToolBar,
  - Buttons, TextFields,
  - ...

# Container?

- ❖ Là thành phần mà có thể chứa các thành phần khác, có thể vẽ và tô màu.
  - ✓ Frame/JFrame, Panel/JPanel, Dialog/JDialog, ScrollPane/JScrollPane, ...
- ❖ Gắn component vào khung chứa
  - ✓ `containerObj.add(compObj);`
- ❖ Lấy thông tin của component
  - ✓ `objectName.get...( );`
- ❖ Gán thông tin cho component
  - ✓ `objectName.set...( );`

# Layout Manager?

---

- Sắp xếp các đối tượng trong các Container.
- Thiết lập Layout cho Container: *setLayout(...)*
- Các loại Layout phổ biến
  - Flow Layout
  - Border Layout
  - GridLayout
  - GridBagConstraints Layout
  - Null Layout
  - ...

# Nguyên tắc xây dựng GUI

---

- Lựa chọn 1 Container
- Thiết lập cách Layout cho khung chứa
- Tạo các điều khiển
- Đưa và sắp xếp điều khiển vào khung chứa
- Thêm xử lý sự kiện

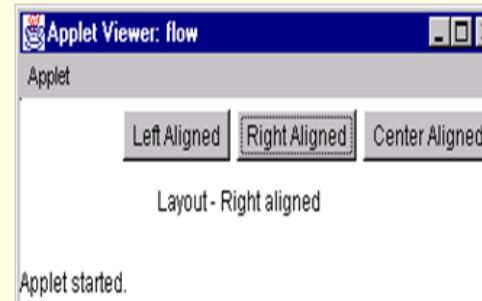
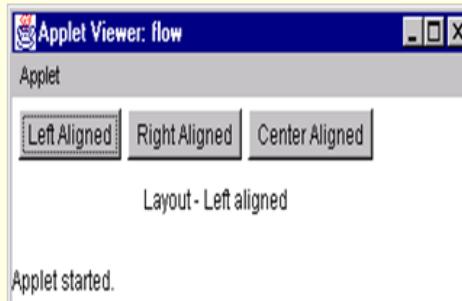
# FlowLayout

- Là Layout mặc định cho Applet và Panel
- Các thành phần sắp từ trái qua phải, trên xuống dưới

*FlowLayout layout = new FlowLayout();*

*FlowLayout layout = new FlowLayout(FlowLayout.RIGHT);*

*(canh lề phải)*

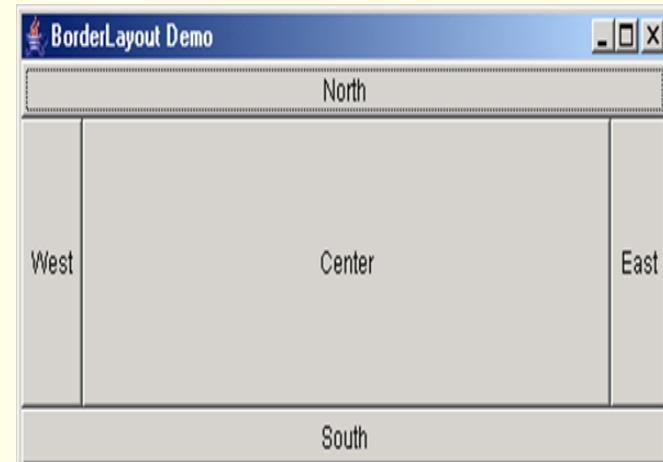


Flow Layout – Left and Right Aligned

# BorderLayout

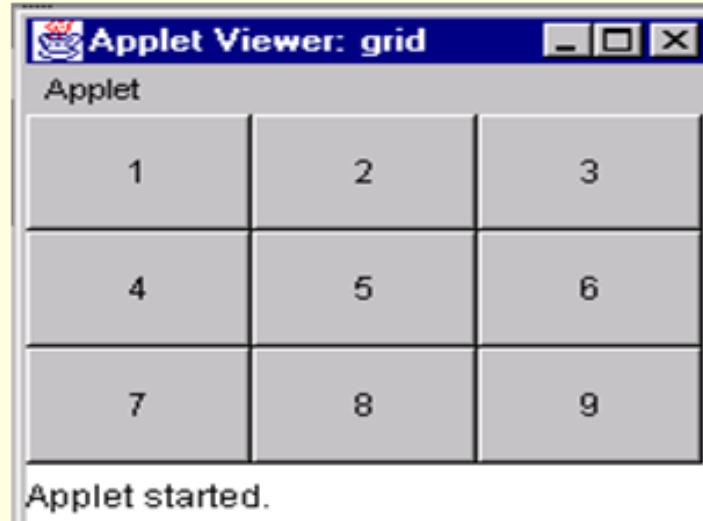
- Là kiểu Layout mặc định cho Frame, Window, Dialog.
- Container được chia thành 5 vùng: NORTH, SOUTH, EAST, WEST, CENTER

```
JFrame fr = new JFrame("BoderLayout Demo");  
fr.setLayout(new BorderLayout());  
  
JButton northButton = new JButton("North");  
fr.add(northButton, BorderLayout.NORTH)  
  
...
```



# Grid Layout

- ❖ 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);
```

# GridBag Layout

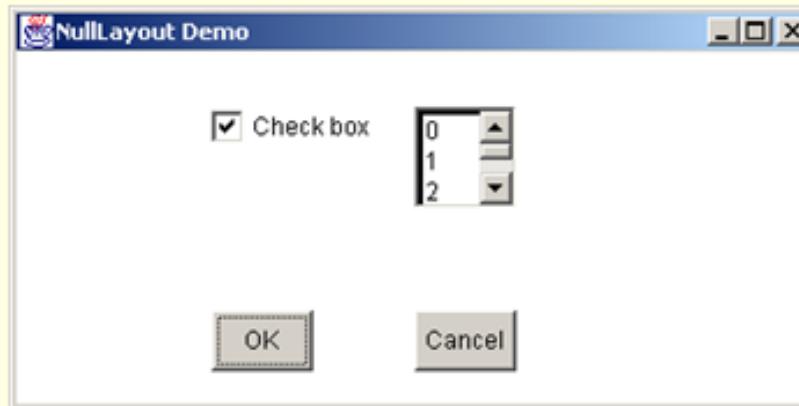
- Sắp xếp trong lưới dòng và cột
- Các thành phần không cần cùng kích thước.
- Không cần ứa trái qua phải & trên xuống dưới.



# Null Layout

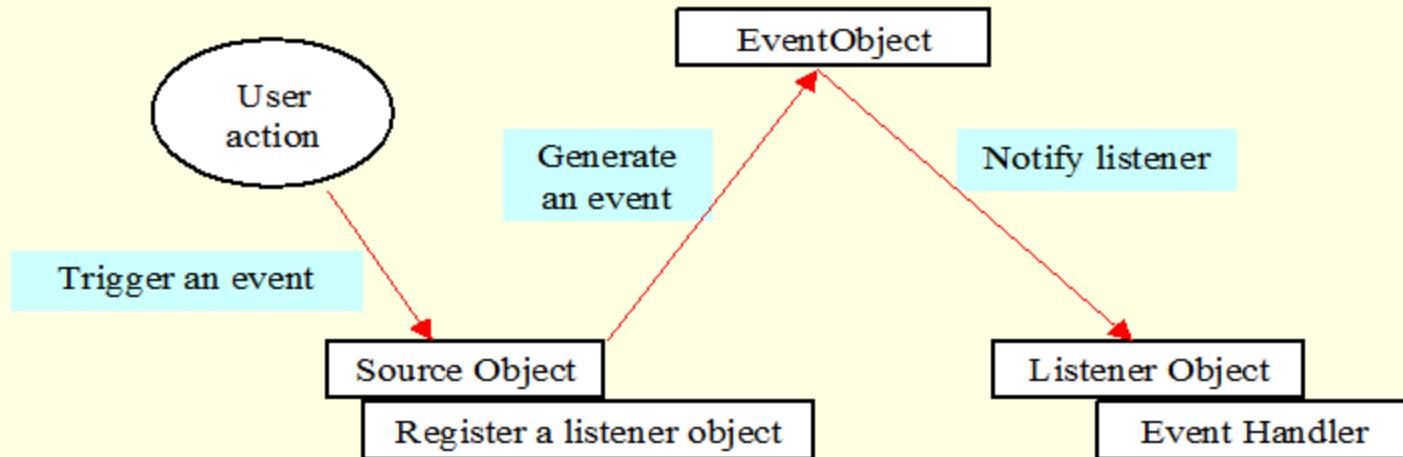
- Tự do sắp xếp các thành phần.

```
Frame fr = new Frame("NullLayout Demo");  
fr.setLayout(null);
```

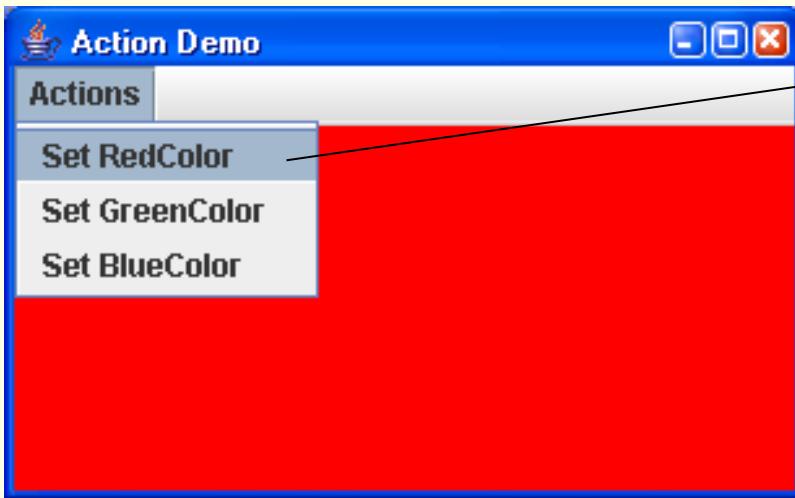


# Mô hình xử lý sự kiện

- 3 yếu tố quan trọng:
  - Nguồn phát sinh sự kiện (Event source)
  - Sự kiện (Event)
  - Bộ lắng nghe sự kiện



# Mô hình xử lý sự kiện



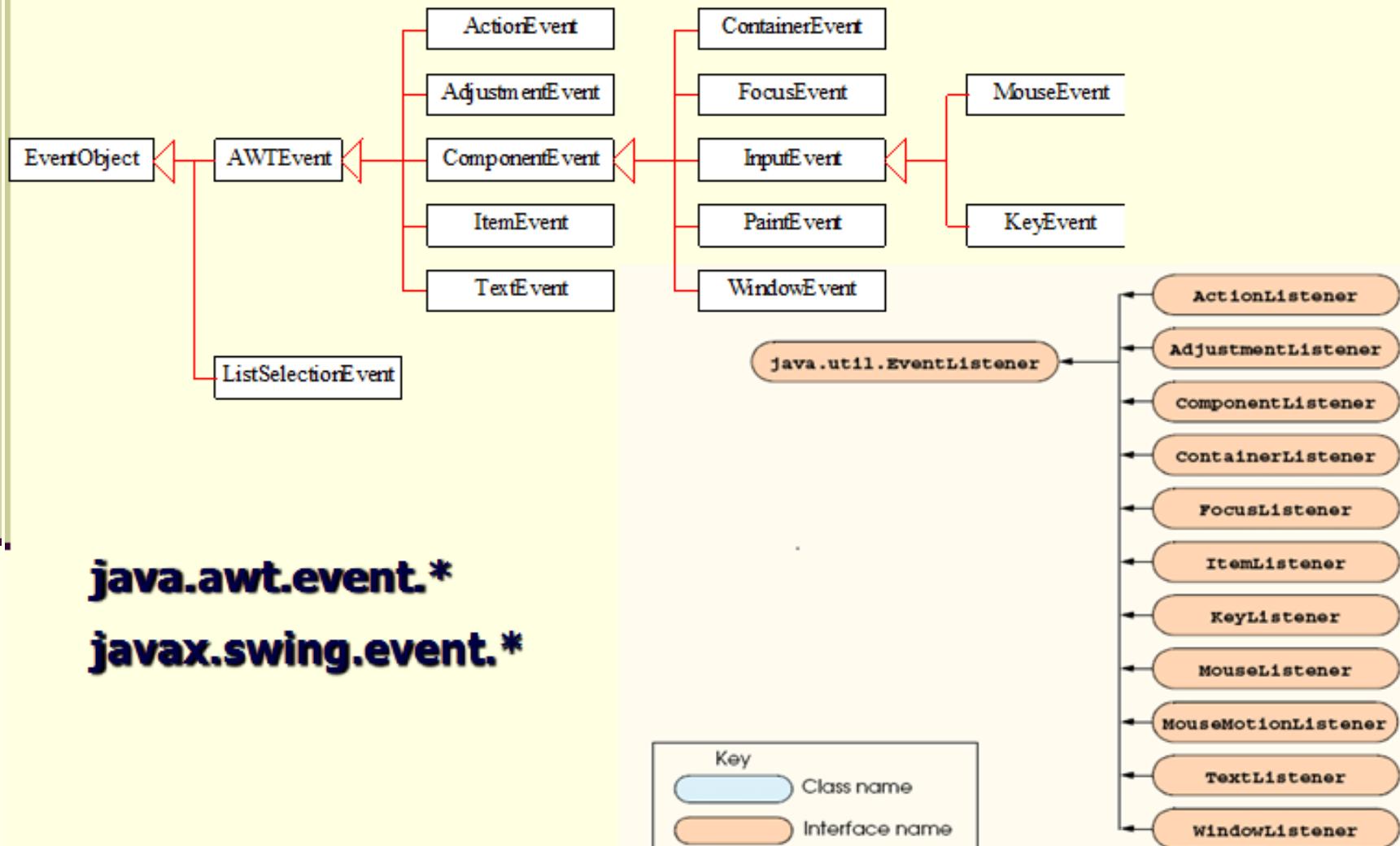
Nguồn phát sinh sự kiện cần được đăng ký bộ lắng nghe sự kiện (Listener) để xử lý khi có tác động từ người dùng.

```
Class MyMenuItemListener implements ActionListener {  
    ...  
}
```

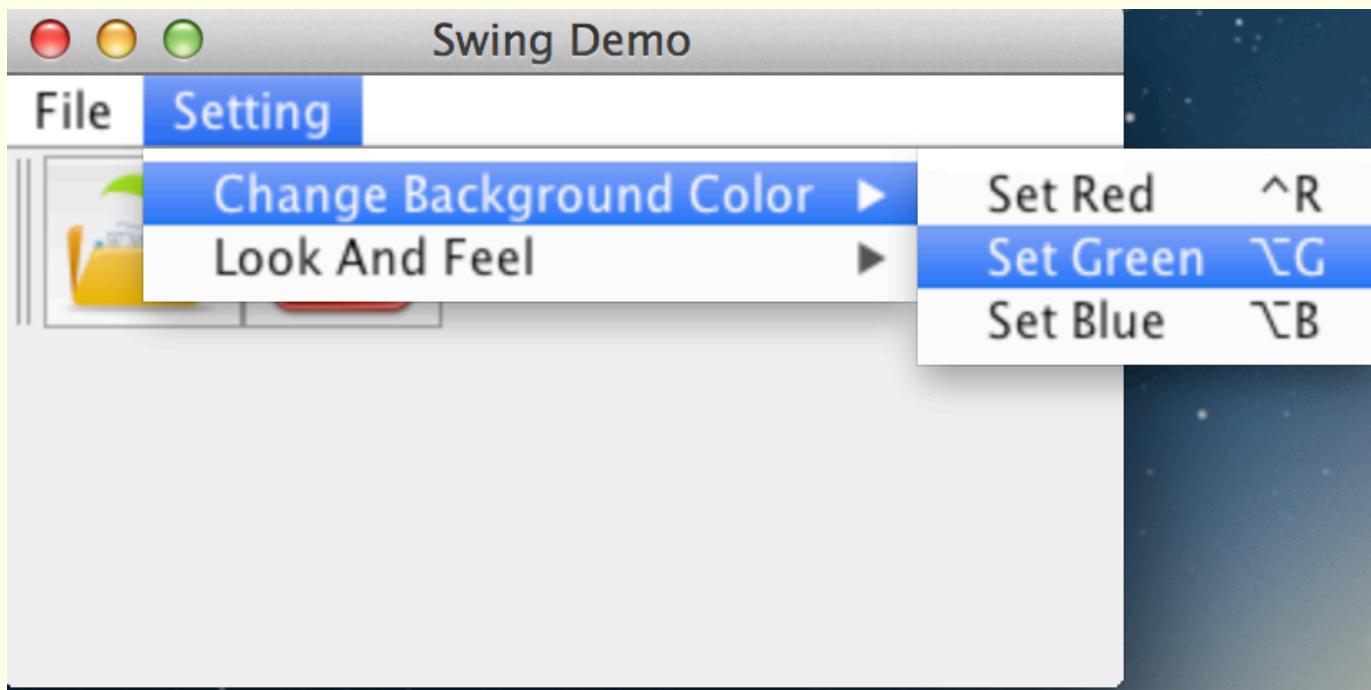
# Sự kiện và Lắng nghe

Đối tượng	Sự kiện	Bộ lắng nghe
Window, Frame, ...	WindowEvent	WindowListener
Button, MenuItem, ...	ActionEvent	ActionListener
TextComponent, ...	TextEvent	TextListener
List, ...	ActionEvent	ActionListener
...	ItemEvent	ItemListener
	ComponentEvent	ComponentListener
	MouseEvent	MouseListener
		MouseMotionListener
	KeyEvent	KeyListener

# Sự kiện và Lắng nghe (tt)



# Ví dụ xử lý sự kiện



```
public class MainFrame extends JFrame implements ItemListener {  
    MainFrame's components  
  
    public MainFrame(String title) {...}  
  
    private void initMenu() {...}  
  
    private void initToolBar() {...}  
  
    class SetRedMenuItemListener implements ActionListener {...}  
  
    class SetGreenMenuItemListener implements ActionListener {...}  
  
    class SetBlueMenuItemListener implements ActionListener {...}  
  
    @Override  
    public void itemStateChanged(ItemEvent e) {...}  
  
    class ExitMenuItemListener implements ActionListener {...}  
  
    Action openFileDialog = new AbstractAction("Open File") {...};  
  
    Action performExit = new AbstractAction("Exit") {...};  
}
```

# Ví dụ KeyListener

## (Nhận biết các character key)

```
public class MyKeyListener implements KeyListener {  
  
    public void keyTyped(KeyEvent e) {  
        System.out.println("Key typed: " + e.getKeyChar());  
    }  
  
    public void keyPressed(KeyEvent e) {  
        System.out.println("Key pressed: " + e.getKeyChar());  
    }  
  
    public void keyReleased(KeyEvent e) {  
        System.out.println("Key released: " + e.getKeyChar());  
    }  
}
```

# Ví dụ KeyListener

## (Nhận biết các non-character key)

```
public class MyKeyListener implements KeyListener {
    Detecting Character Key
    @Override
    public void keyTyped(KeyEvent e) {...}
    @Override
    public void keyPressed(KeyEvent e) {...}
    @Override
    public void keyReleased(KeyEvent e) {

        if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
            System.out.println("Right released.");
        } else if (e.getKeyCode() == KeyEvent.VK_LEFT) {
            System.out.println("Left released.");
        } else if (e.getKeyCode() == KeyEvent.VK_UP) {
            System.out.println("Up released.");
        } else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
            System.out.println("Down released.");
        } else {
            System.out.println("Key released: " + e.getKeyChar());
        }
    }
}
```

# Ví dụ KeyListener

```
public static void main(String[] args) {  
    JFrame frame = new JFrame("Custom Painting");  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    JPanel panel = new JPanel();  
    panel.addKeyListener(new MyKeyListener());  
  
    frame.add(panel);  
    frame.setSize(500, 500);  
    frame.setVisible(true);  
  
    //make sure the JPanel has the focus  
    panel.requestFocusInWindow();  
}
```

# Ví dụ MouseListener & MouseMotionListener

```
public class MyMouseListener implements MouseListener {  
    public void mouseClicked(MouseEvent e) {  
        System.out.println("Mouse clicked at " + e.getX() + ", " + e.getY());  
    }  
    public void mousePressed(MouseEvent e) {  
        System.out.println("Mouse pressed at " + e.getX() + ", " + e.getY());  
    }  
    public void mouseReleased(MouseEvent e) {  
        System.out.println("Mouse released at " + e.getX() + ", " + e.getY());  
    }  
    public void mouseEntered(MouseEvent e) {  
        System.out.println("Mouse entered at " + e.getX() + ", " + e.getY());  
    }  
    public void mouseExited(MouseEvent e) {  
        System.out.println("Mouse exited at " + e.getX() + ", " + e.getY());  
    }  
}
```

# Ví dụ MouseListener & MouseMotionListener

```
public class MyMouseMotionListener implements MouseMotionListener {  
    @Override  
    public void mouseDragged(MouseEvent e) {  
        System.out.println("Mouse dragged: " + e.getX() + ", " + e.getY());  
    }  
  
    @Override  
    public void mouseMoved(MouseEvent e) {  
        System.out.println("Mouse moved: " + e.getX() + ", " + e.getY());  
    }  
}
```

# Ví dụ MouseListener & MouseMotionListener

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Custom Painting");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    JPanel panel = new JPanel();
    panel.addKeyListener(new MyKeyListener());
    panel.addMouseListener(new MyMouseListener());
    panel.addMouseMotionListener(new MyMouseMotionListener());

    frame.add(panel);
    frame.setSize(500, 500);
    frame.setVisible(true);

    //make sure the JPanel has the focus
    panel.requestFocusInWindow();
}
```

# Demo một số Swing component phổ biến

(Sinh viên có thể tham khảo địa chỉ

<https://docs.oracle.com/javase/tutorial/uiswing/components/>)

# JFrame

```
// Thiet lap che do look and feel  
JFrame.setDefaultLookAndFeelDecorated(true);  
  
// Tao 1 doi tuong JFrame  
JFrame frame = new JFrame("JFrame Demo");  
  
// Xu ly khi dong frame  
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
// Tao 1 component va them vao JFrame  
JLabel label = new JLabel("This is a JLabel");  
frame.getContentPane().add(label, BorderLayout.NORTH);  
  
// Dinh cac thuoc tinh cho frame  
frame.setBounds(50, 50, 200, 150);  
frame.getContentPane().setBackground(Color.WHITE);  
  
// Hien thi frame  
frame.setVisible(true);
```



# JMenuBar, JMenu, JMenuItem, JRadioButtonMenuItem

```
JMenuBar mb = new JMenuBar();
JMenu menuActions = new JMenu("Actions");
JMenuItem miSetRed = new JMenuItem("Set Red");
JMenuItem miSetGreen = new JMenuItem("Set Green");
JMenuItem miSetBlue = new JMenuItem("Set Blue");

MainMenuListener menuListener = new MainMenuListener(frame);
miSetRed.addActionListener(menuListener);
miSetGreen.addActionListener(menuListener);
miSetBlue.addActionListener(menuListener);

menuActions.add(miSetRed);
menuActions.add(miSetGreen);
menuActions.add(miSetBlue);
mb.add(menuActions);

frame.setJMenuBar(mb);
```

# JMenuBar, JMenu, JMenuItem, JRadioButtonMenuItem

```
import java.awt.Color;

public class MainMenuListener implements ActionListener {
    private JFrame frame = null;
    public MainMenuListener(JFrame fr) {
        frame = fr;
    }
    public void actionPerformed(ActionEvent event) {
        String actionStr = event.getActionCommand();
        if (actionStr.equalsIgnoreCase("Set Red")) {
            frame.getContentPane().setBackground(Color.red);
        }
        if (actionStr.equalsIgnoreCase("Set Green")) {
            frame.getContentPane().setBackground(Color.green);
        }
        if (actionStr.equalsIgnoreCase("Set Blue")) {
            frame.getContentPane().setBackground(Color.blue);
        }
    }
}
```

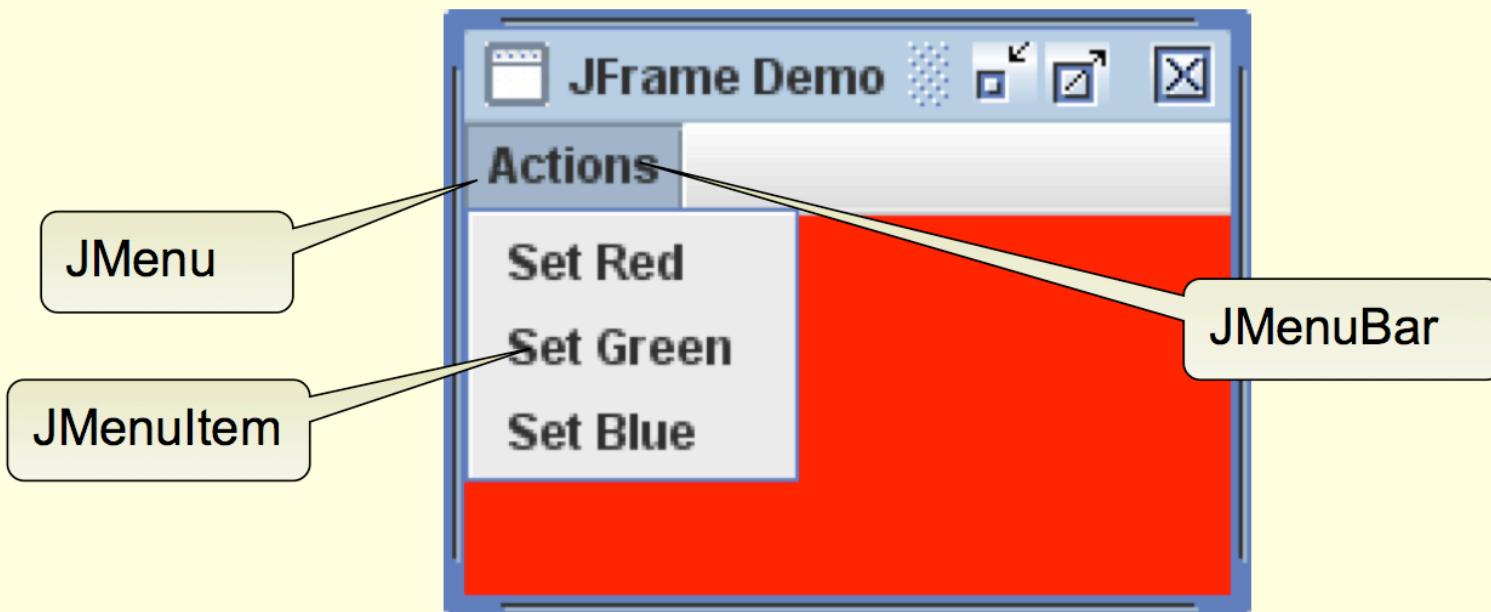
---

# Tạo Hệ thống Thực đơn

## dùng JMenuBar, JMenu, JMenuItem

# JMenuBar, JMenu, JMenuItem, JRadioButtonMenuItem

---

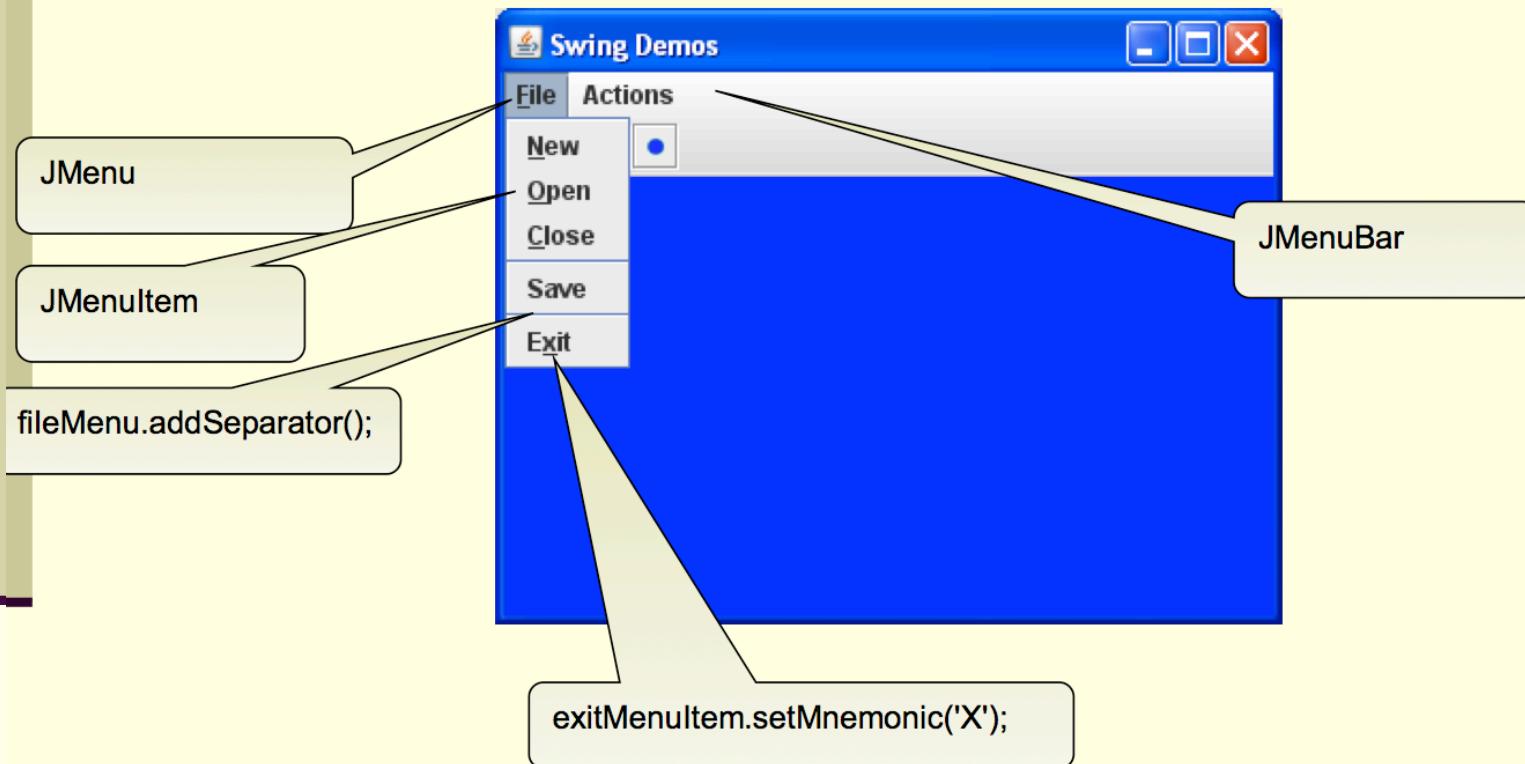


# Các bước thực hiện

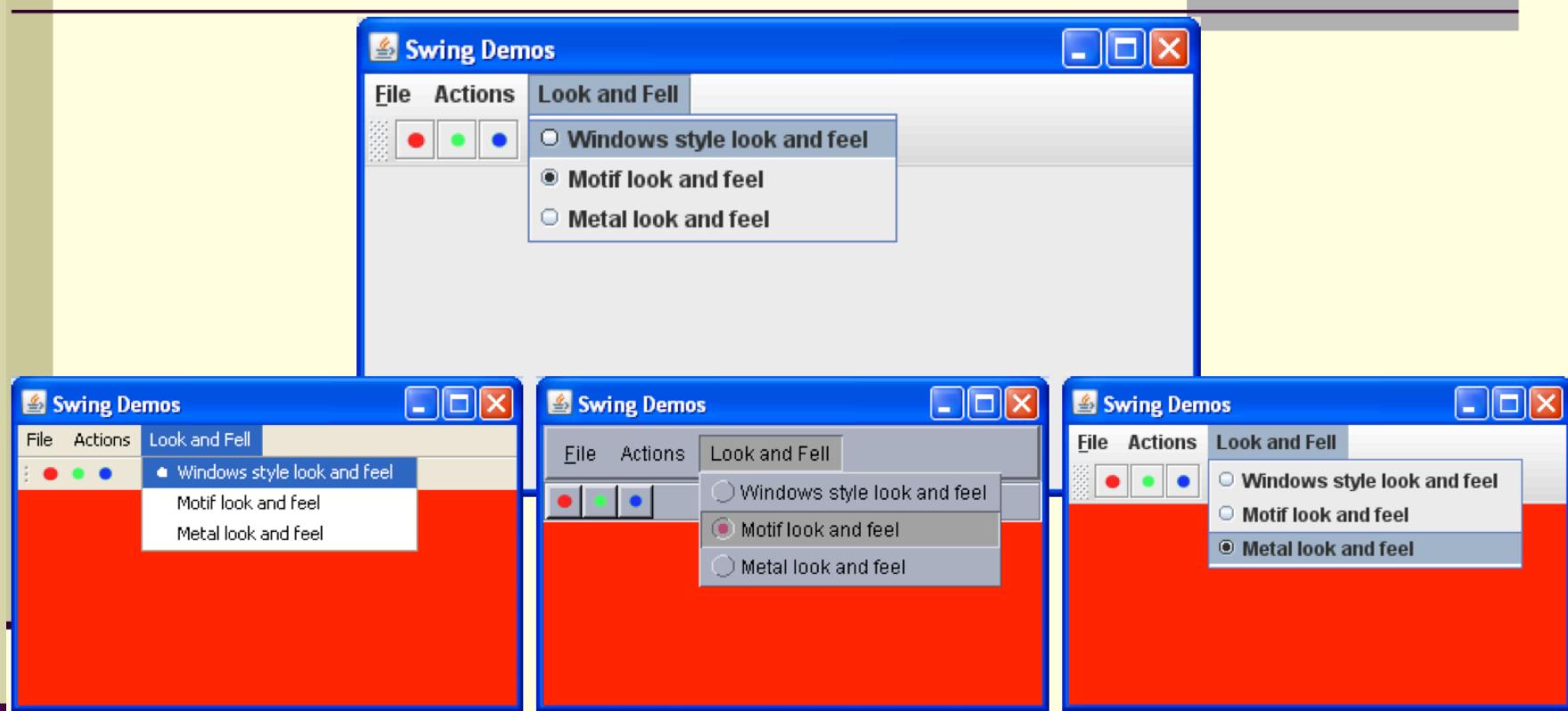
---

- Tạo cửa sổ chính dùng JFrame, đặt tên MainFrame
- Tạo các thành phần cho hệ thống thực đơn
  - menubar dùng JMenuBar
  - menu “Actions” dùng JMenu
  - Các menuitem như “Set Red”, “Set Green”, “Set Blue” dùng JMenuItem
- Gắn các menuitem vào menu, sau đó là menu vào menubar và cuối cùng là gắn menubar vào cửa sổ chính.
- Gắn xử lý sự kiện cho các menuitem dùng bộ xử lý sự kiện ActionListener

# JMenuBar, JMenu, JMenuItem, JRadioButtonMenuItem



# JRadioButtonMenuItem, ButtonGroup



- Tham khảo thêm về Look and Feel trong tài liệu JFC Swing Tutorial phần “**How to Set the Look and Feel**” trong chương 9 “**Other Swing Features Reference**”

# JRadioButtonMenuItem, ButtonGroup

```
public class MainClass extends JFrame implements ItemListener {

    MainClass() {
        JMenuBar menuBar = new JMenuBar();
        JMenu options = (JMenu) menuBar.add(new JMenu("L&F"));
        ButtonGroup group = new ButtonGroup();

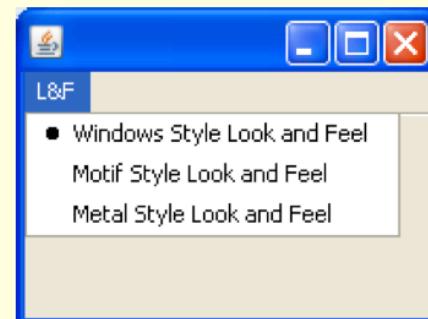
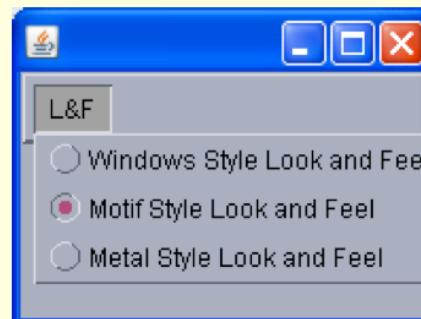
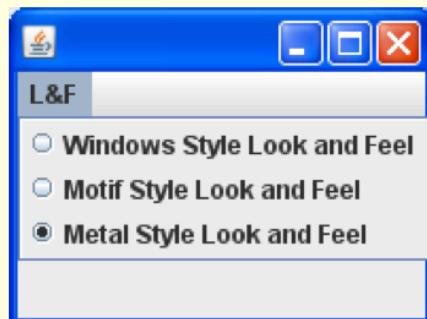
        JMenuItem miWin= options.add(new JRadioButtonMenuItem("Windows Style Look and Feel"));
        miWin.setActionCommand("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
        JMenuItem miMotif= options.add(new JRadioButtonMenuItem("Motif Style Look and Feel"));
        miMotif.setActionCommand("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
        JMenuItem miMetal= options.add(new JRadioButtonMenuItem("Metal Style Look and Feel"));
        miMetal.setActionCommand("javax.swing.plaf.metal.MetalLookAndFeel");

        group.add(miWin);
        group.add(miMotif);
        group.add(miMetal);
        miWin.addItemListener(this);
        miMotif.addItemListener(this);
        miMetal.addItemListener(this);
        this.setJMenuBar(menuBar);
        this.setBounds(50, 50, 200, 150);
        this.setVisible(true);
    }
}
```

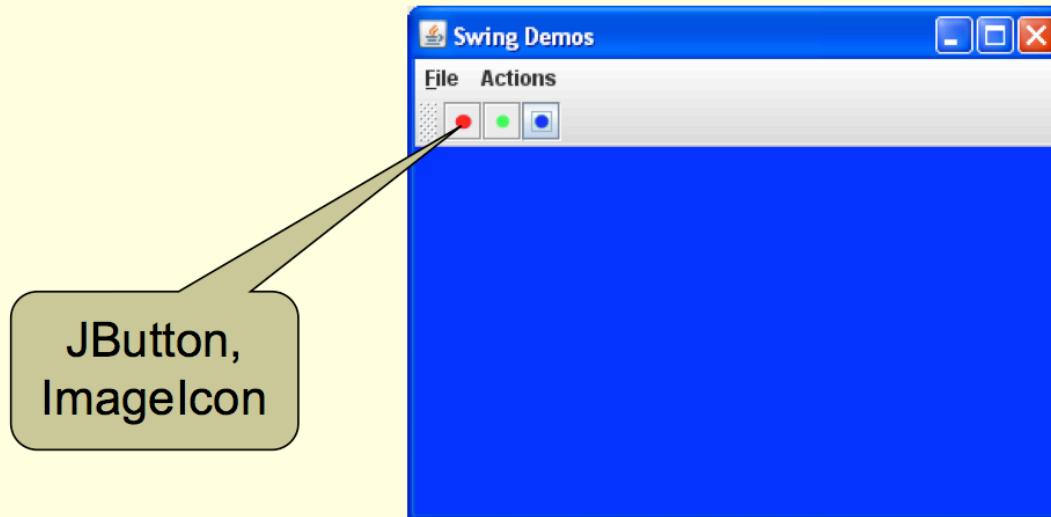
# JRadioButtonMenuItem, ButtonGroup

```
public void itemStateChanged(ItemEvent e) {
    JRadioButtonMenuItem button = (JRadioButtonMenuItem)e.getSource();
    try {
        if(button.isSelected()) {
            UIManager.setLookAndFeel(button.getActionCommand());
            button.setEnabled(true);
            SwingUtilities.updateComponentTreeUI(this);
        }
    }
    catch (Exception ex){
        System.out.println(e.toString());
    }
}

public static void main(String[] args) {
    new MainClass();
}
```



# JToolBar, Icon, ImageIcon



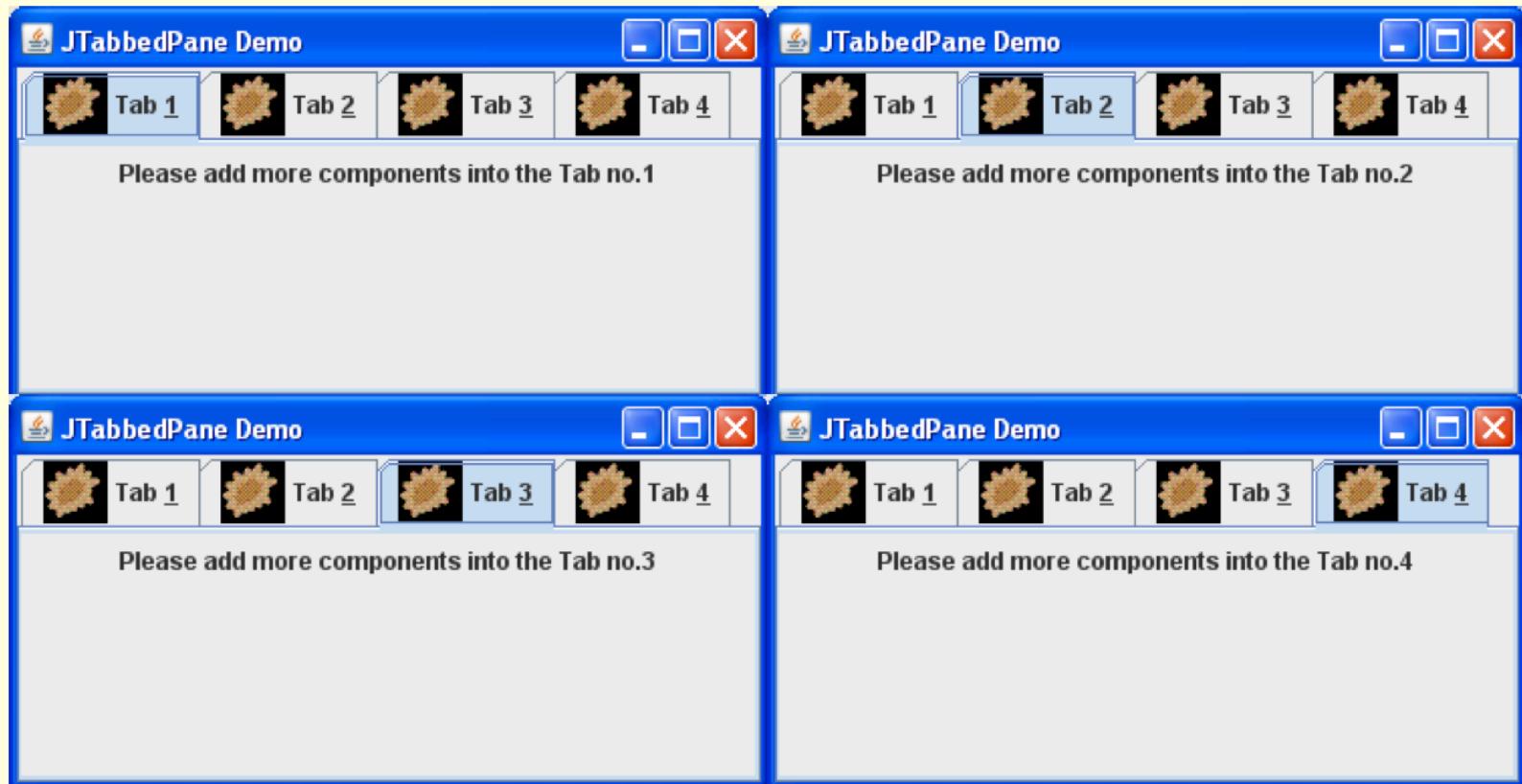
```
// Create toolbar
JToolBar toolbar = new JToolBar();
MainToolBarListener actionListener = new MainToolBarListener(this);
Icon red = new ImageIcon("images/red.png");
redIcon = new JButton(red);
redIcon.addActionListener(actionListener);
toolbar.add(redIcon);
```

# JToolBar, Icon, ImageIcon

- Đặt tooltip cho Icon trên thanh toolbar

```
// Create toolbar
JToolBar toolbar = new JToolBar();
MainToolBarListener actionListener = new MainToolBarListener(this);
Icon red = new ImageIcon("images/red.png");
redIcon = new JButton(red);
redIcon.setToolTipText("Set red color for the background");
redIcon.addActionListener(actionListener);
toolbar.add(redIcon);
```

# JTabbedPane



# JTablePane

---

`javax.swing`

## Class JTabbedPane

`java.lang.Object`

  └ `java.awt.Component`

    └ `java.awt.Container`

      └ `javax.swing.JComponent`

        └ `javax.swing.JTabbedPane`

### All Implemented Interfaces:

[ImageObserver](#), [MenuContainer](#), [Serializable](#), [Accessible](#), [SwingConstants](#)

---

# JTabbedPane

```
JTabbedPane tabbedPane = new JTabbedPane();
ImageIcon icon = new ImageIcon("middle.gif");
JPanel panel1 = new JPanel(new FlowLayout());
panel1.add(new JLabel("Please add more components into the Tab no.1"));
tabbedPane.addTab("Tab 1", icon, panel1, "This is the tab no.1");
tabbedPane.setMnemonicAt(0, KeyEvent.VK_1);

JPanel panel2 = new JPanel(new FlowLayout());
panel2.add(new JLabel("Please add more components into the Tab no.2"));
tabbedPane.addTab("Tab 2", icon, panel2, "This is the tab no.2");
tabbedPane.setMnemonicAt(1, KeyEvent.VK_2);
```

- **Tạo mới đối tượng JTabbedPane**

```
JTabbedPane tabbedPane = new JTabbedPane();
```

- **Gắn thêm 1 Tab mới vào đối tượng JTabbedPane**

```
tabbedPane.addTab("Tab name", icon, component, "Tooltip");
```

# JTabbedPane

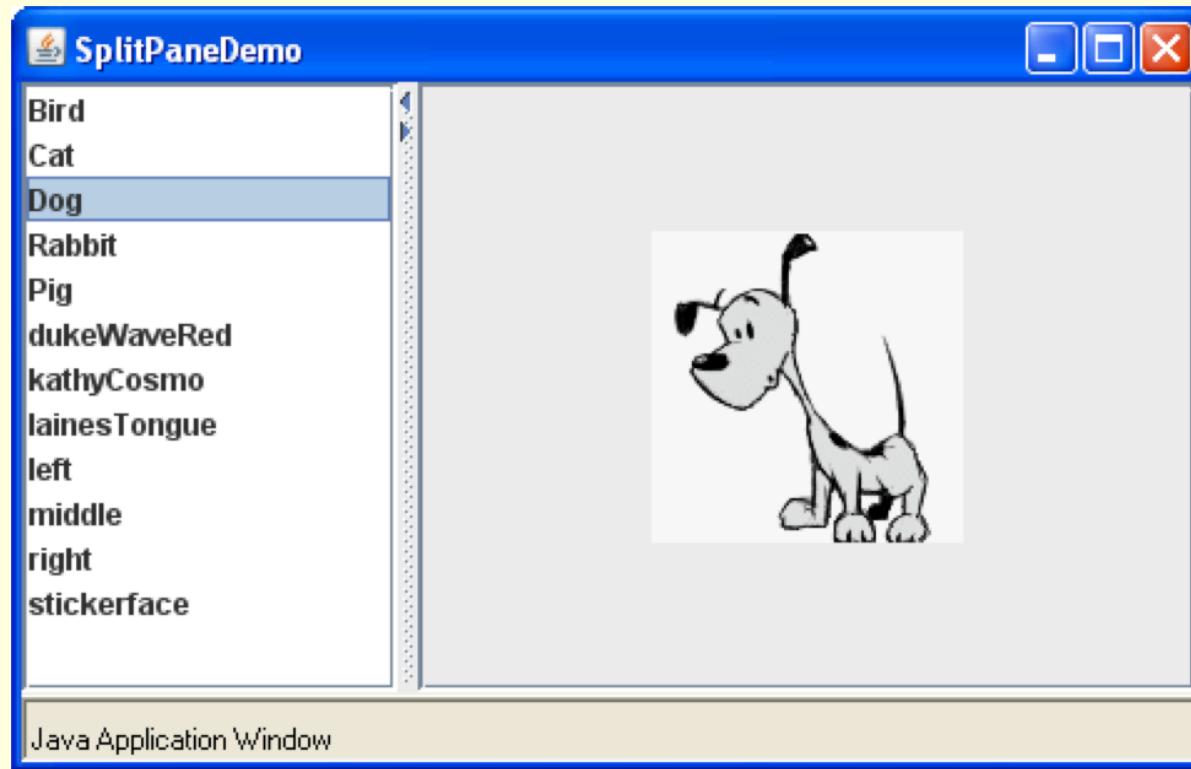
```
JPanel panel3 = new JPanel(new FlowLayout());
panel3.add(new JLabel("Please add more components into the Tab no.3"));
tabbedPane.addTab("Tab 3", icon, panel3, "This is the tab no.3");
tabbedPane.setMnemonicAt(2, KeyEvent.VK_3);

JPanel panel4 = new JPanel(new FlowLayout());
panel4.add(new JLabel("Please add more components into the Tab no.4"));
panel4.setPreferredSize(new Dimension(410, 50));
tabbedPane.addTab("Tab 4", icon, panel4, "This is the tab no.4");
tabbedPane.setMnemonicAt(3, KeyEvent.VK_4);

JFrame fr = new JFrame("JTabbedPane Demo");
fr.setBounds(50, 50, 400, 300);
fr.add(tabbedPane, BorderLayout.CENTER);

fr.setVisible(true);
```

# JSplitPane



<http://java.sun.com/docs/books/tutorial/uiswing/examples/components/>

# JSplitPane

```
JFrame fr = new JFrame("JSplitPane Demo");
fr.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JSplitPane splitPane;
JLabel picture = new JLabel();
String[] imageNames = { "Bird", "Cat", "Dog", "Rabbit", "Pig", "dukeWaveRed",
                      "kathyCosmo", "lainesTongue", "left", "middle", "right", "stickerface"};

//Create the list of images and put it in a scroll pane.
JList list = new JList(imageNames);
list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
list.setSelectedIndex(0);
MyListSelectionListener listener = new MyListSelectionListener(picture, imageNames);
list.addListSelectionListener(listener);
JScrollPane listScrollPane = new JScrollPane(list);
```

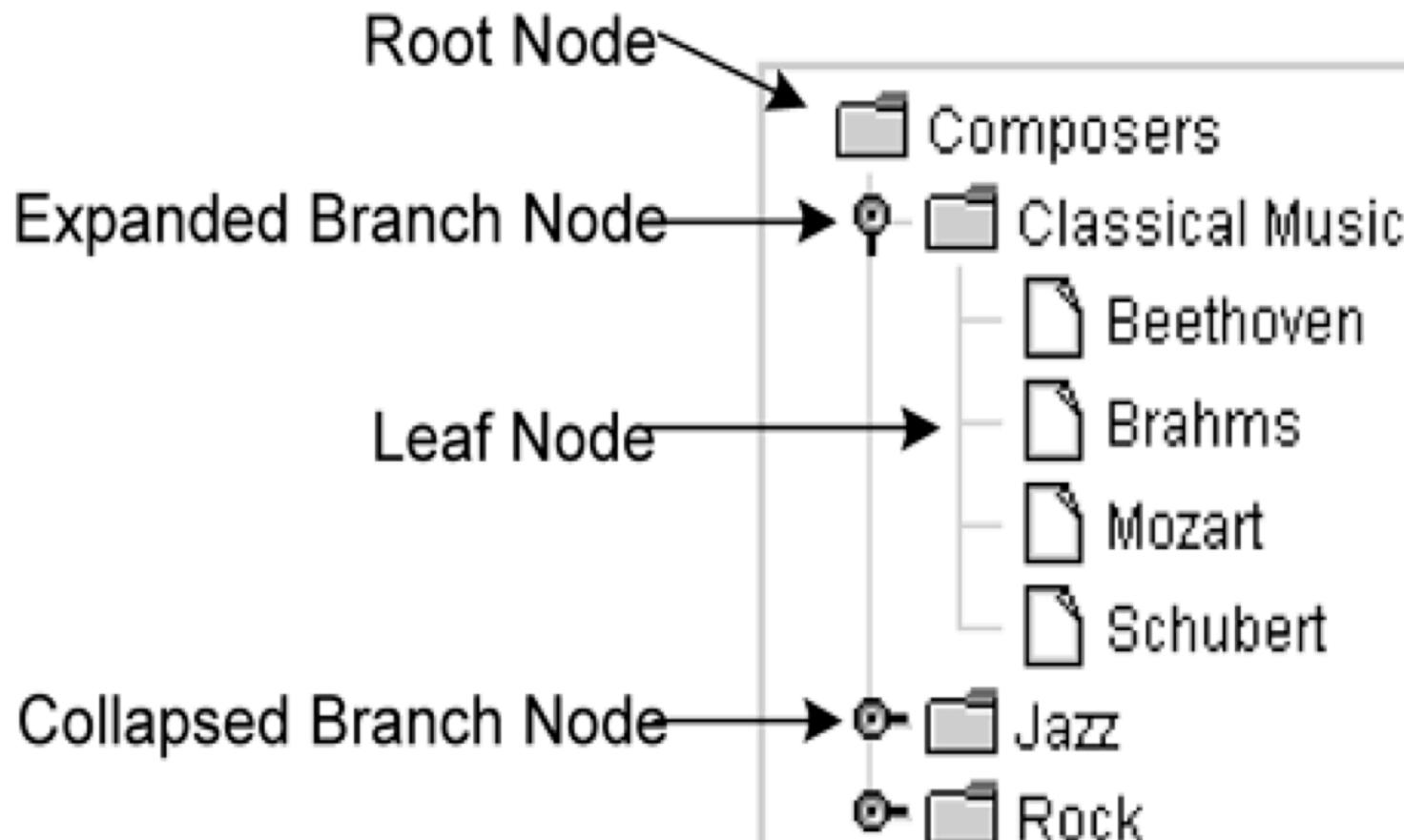
# JSplitPane, JScrollPane, JList

```
// Create a split pane with the two scroll panes in it.  
splitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, listScrollPane, pictureScrollPane);  
splitPane.setOneTouchExpandable(true);  
splitPane.setDividerLocation(150);  
  
// Provide minimum sizes for the two components in the split pane.  
Dimension minimumSize = new Dimension(100, 50);  
listScrollPane.setMinimumSize(minimumSize);  
pictureScrollPane.setMinimumSize(minimumSize);  
  
//Provide a preferred size for the split pane.  
splitPane.setPreferredSize(new Dimension(400, 200));  
  
fr.getContentPane().add(splitPane);  
fr.pack();  
fr.setVisible(true);
```

# JSplitPane, JScrollPane, JList

```
public class MyListSelectionListener implements ListSelectionListener {  
    String[] imageNames;  
    JLabel picture;  
  
    MyListSelectionListener(JLabel pic, String[] images) {  
        picture = pic;  
        imageNames = images;  
    }  
  
    public void valueChanged(ListSelectionEvent e) {  
        JList list = (JList)e.getSource();  
        ImageIcon icon = new ImageIcon(imageNames[list.getSelectedIndex()] + ".gif");  
        picture.setIcon(icon);  
    }  
}
```

# JTree



# JTree

`javax.swing`

## Class JTree

`java.lang.Object`

└ `java.awt.Component`

  └ `java.awt.Container`

    └ `javax.swing.JComponent`

      └ `javax.swing.JTree`

### All Implemented Interfaces:

`ImageObserver`, `MenuContainer`, `Serializable`, `Accessible`, `Scrollable`

# JTree

- ❖ Tạo một node trên cây làm root

```
DefaultMutableTreeNode top = new DefaultMutableTreeNode("The Java Series");
```

- ❖ Tạo các node bên dưới

```
DefaultMutableTreeNode category = null;
```

```
DefaultMutableTreeNode book = null;
```

```
category = new DefaultMutableTreeNode("Books for Java Programmers");
```

```
top.add(category);
```

```
book = new DefaultMutableTreeNode( "Core Java");
```

```
category.add(book);
```

```
...
```

```
JTree tree = new JTree(top);
```

```
...
```

```
JScrollPane treeView = new JScrollPane(tree);
```

# Tài liệu đọc thêm

---

- [1] Kathy Walrath, Mary Campione, Alison Huml, Sharon Zakhour. *The JFC Swing Tutorial, Second Edition.* Copyright © 2004 Sun Microsystems, Inc.

