Lập trình đồ họa với AWT

CAO Duc Thong – Thanglong University
thongcd@thanglong.edu.vn

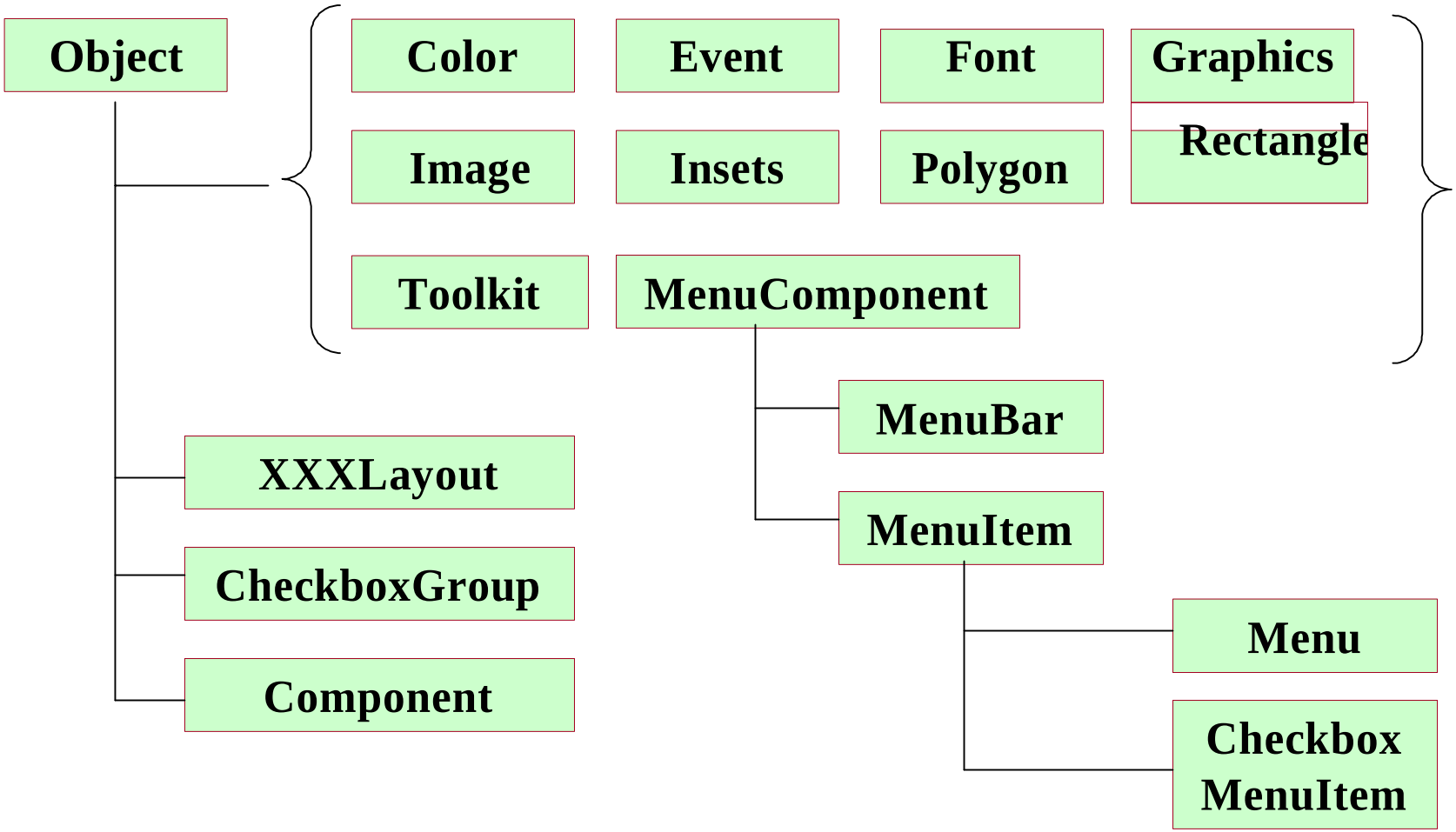
Nội dung

- Tổng quan về AWT
- Các thành phần AWT
- Quản lý trình bày
- Xử lý Sự kiện

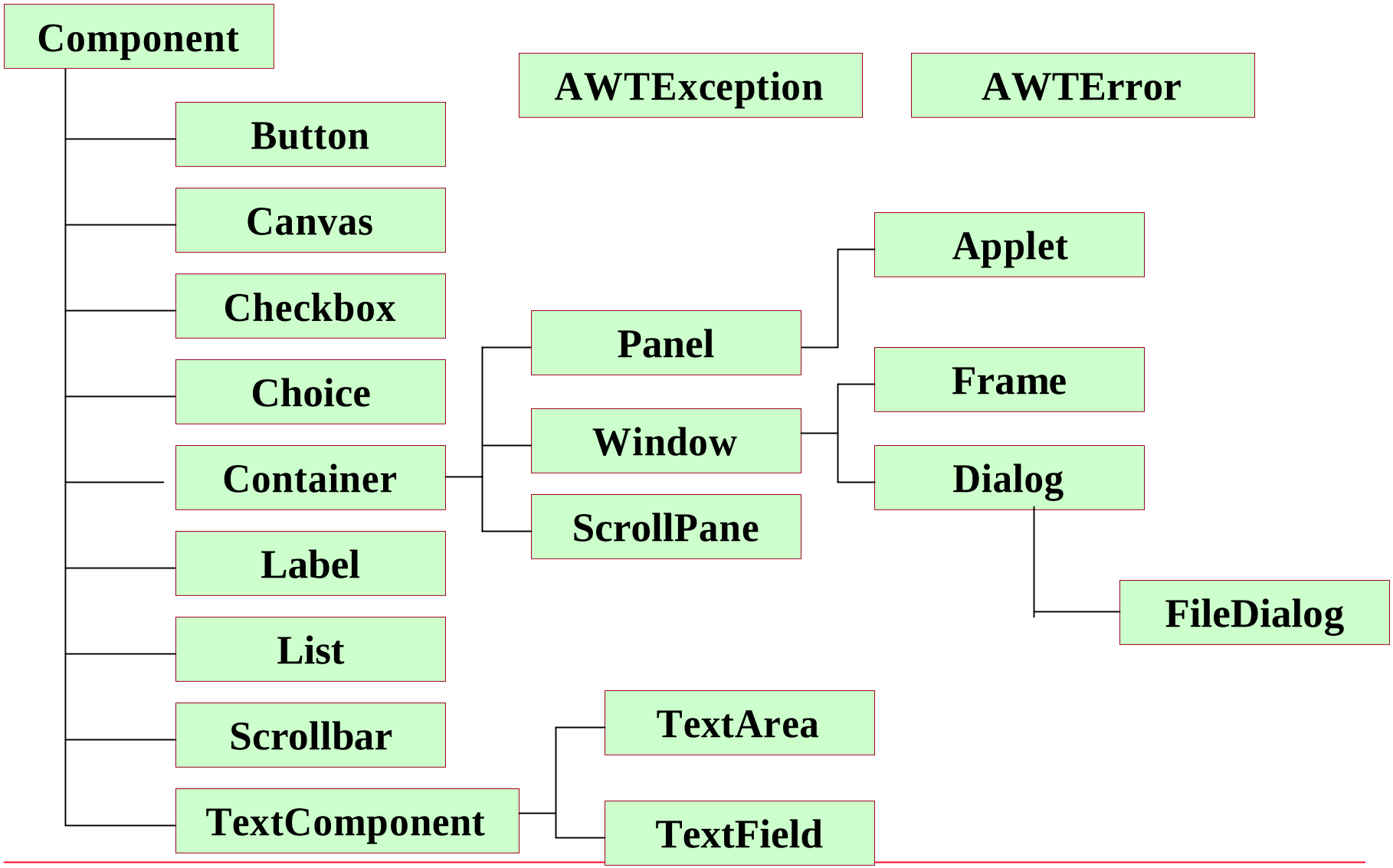
Thư viện AWT

- AWT là viết tắt của Abstract Windowing Toolkit
- AWT cho phép tạo các thành phần đồ họa
- AWT cho phép nhận dữ liệu từ chuột, bàn phím
- Các thành phần đồ họa cơ bản của AWT
  - Vật chứa (Container)
  - Thành phần (Component)
  - Trình quản lý trình bày (Layout Manager)
  - Đồ họa (Graphics), phông chữ (Font), sự kiện (Event)

Thư viện AWT


Object
Color
Event
Font
Graphics
Image
Insets
Polygon
Rectangle
Toolkit
MenuComponent
MenuBar
XXXLayout
MenuItem
CheckboxGroup
Menu
Component
Checkbox
MenuItem

Thư viện AWT


Component
AWTException
AWTError
Button
Canvas
Checkbox
Choice
Container
Label
List
Scrollbar
TextComponent
Panel
Window
ScrollPane
Applet
Frame
Dialog
FileDialog
TextArea
TextField

Cácthànhphầ̀nAWT

- Component (thành phần)
  - Là một đối tượng đồ họa có thể hiển thị được trên màn hình và có thể tương tác với người dùng
  - Là một abstract superclass cho hầu hết các component của AWT
  - Ví dụ về Component: button, checkbox, scrollbar...
- Một số phương thức cơ bản của Component
  - getBackGround(): trả về mầu nền của Component
  - getBounds(): trả về phạm vi của Component (Rectangle)
  - getFont(): trả về font hiện tại của Component
  - getForeGround(): trả về màu vẽ của Component
  - getHeight(): trả về chiều cao của Component (pixel, kiểu int)

Cácthànhphầ nAWT

- Một số phương thức cơ bản của Component
  - getSize(): trả về kích thước của Component (Dimenstion)
  - getWidth(): trả về chiều rộng của Component (int)
  - getX(), getY(): trả về tọa độ hiện tại
  - isEnable(): boolean
  - paint(Graphics): chịu trách nhiệm hiển thị component
  - repaint(): được gọi để vẽ lại giao diện cho component
  - setVisible(boolean): hiển thị component

Cácthànhphầ nAWT

- Container (vật chứa)
  - Chứa trong gói java.awt
  - Là vùng có thể đặt các thành phần giao diện
  - Một số loại Container: Panel, Frame, Dialog
  - Có một Component có khả năng chứa các Component khác
  - Để thêm một Component vào Container ta sử dụng phương thức add(Component)
  - Container sử dụng một “layout manager” để sắp xếp các Component

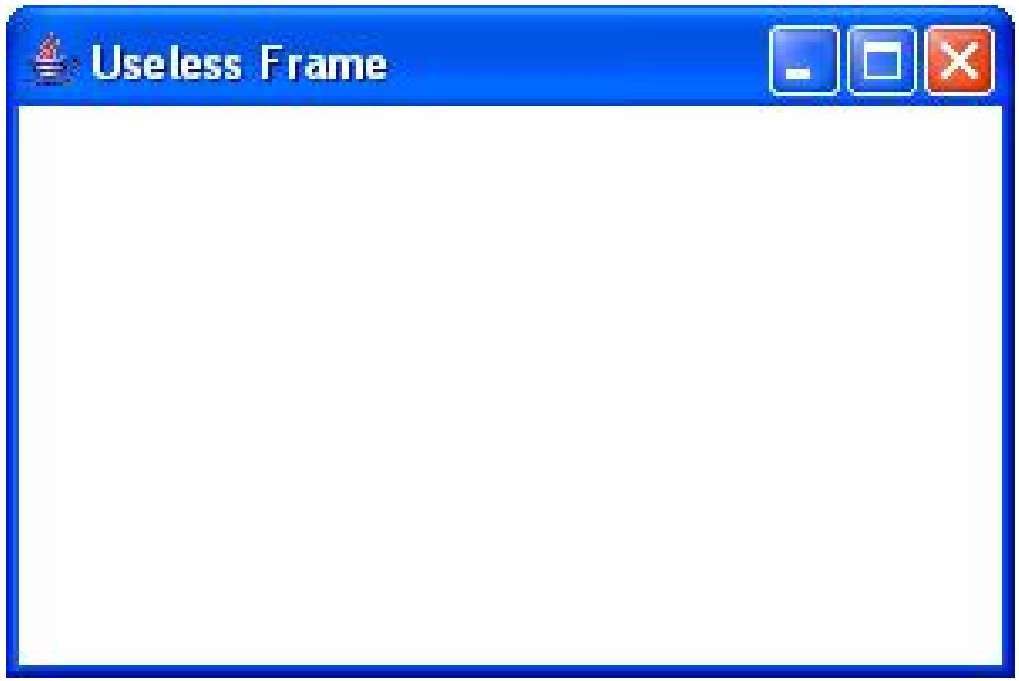
Cácthànhphầ nAWT

- Frame
  - Thừa kế từ Window nên cũng là một Container
  - Frame vừa là Component vừa là Container
  - Tạo Frame
    - Frame()
    - Frame(String title)
  - Ví dụ

③ Frame

```
import java.awt.*;
public class UseLessFrame extends Frame {
        public UseLessFrame(){
                super("Useless Frame");
                setSize(300,200);
                setVisible(true);
        }
        public static void main(String[] args) {
                UseLessFrame frame = new UseLessFrame();
        }
}
```

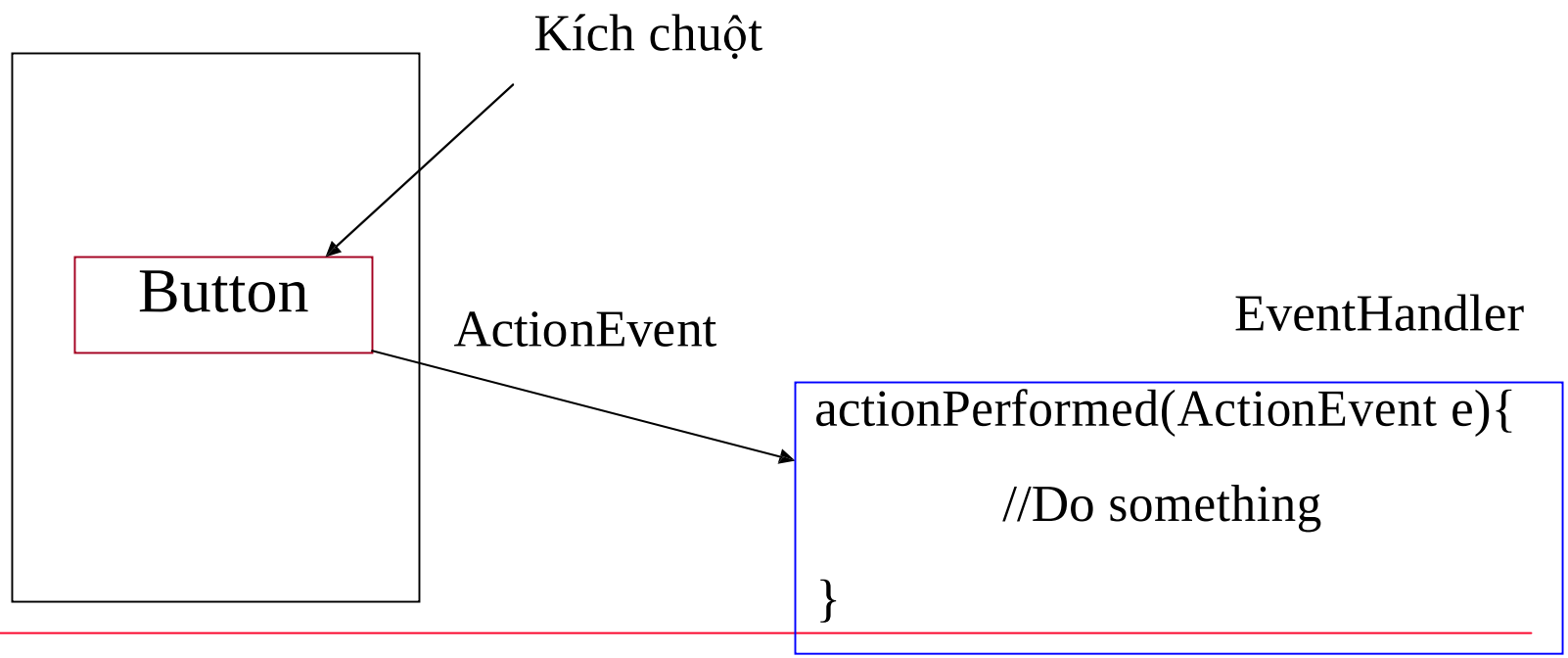
- Frame



Xử lý cácsựkiện

Sự kiện là gì

- Khi người dùng thực hiện một hành động trên GUI, một sự kiện sẽ được sinh ra.
- Các sự kiện là các đối tượng mô tả những gì đã xảy ra
- Mỗi tác động của người dùng sẽ tạo ra những loại sự kiện khác nhau



Xử lý các sự kiện

- Event Sources
  - Là nguồn sinh ra sự kiện.
  - Sự kiện được sinh ra dưới dạng một object, ví dụ ActionEvent.
- Event Handlers
  - Event handler là một phương thức, nó nhận về một đối tượng event, giải mã và xử lý các tương tác với người dùng
  - Lớp chứa các phương thức (event handler) được gọi là lớp nghe sự kiện (listener)
  - Để một lớp nghe có thể xử lý một sự kiện nào đó, nguồn sinh ra sự kiện cần phải đăng ký lớp nghe
  - Một nguồn sinh ra sự kiện có thể có nhiều lớp nghe
  - Một lớp muốn là listener phải implement một giao tiếp thích hợp

- Ví dụ về xử lý sự kiện

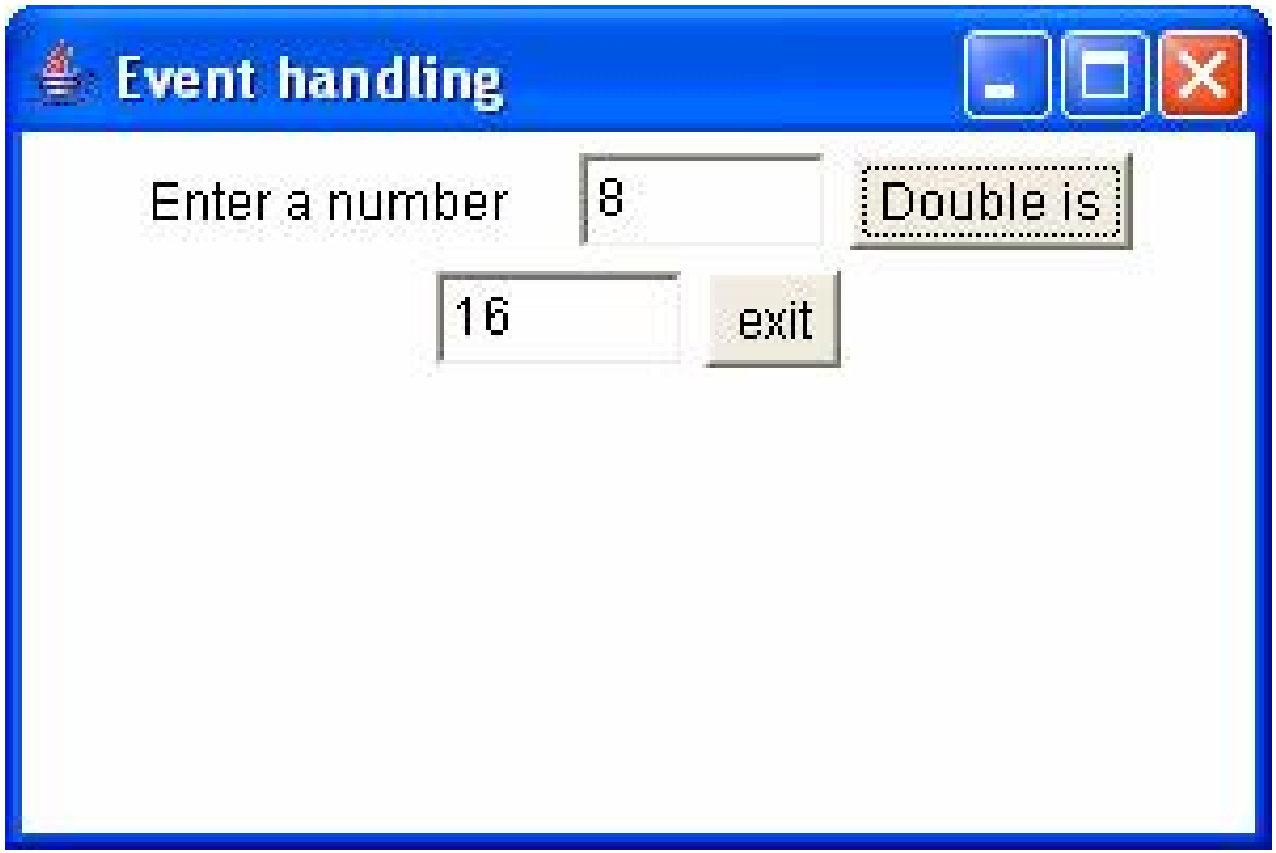
```
import java.awt.*;
import java.awt.event.*;
class EventTest extends Frame implements ActionListener {
        Label lab = new Label("Enter a number");
        TextField tf1 = new TextField(5);
        TextField tf2 = new TextField(5);
        Button btnResult = new Button("Double is");
        Button ext = new Button("exit");
        public EventTest(String title) {
                super(title);
                setLayout(new FlowLayout());
                btnResult.addActionListener(this);
                ext.addActionListener(this);
                add(lab); add(tf1); add(btnResult); add(tf2);  add(ext);
        }
```

- Ví dụ về xử lý sự kiện

```
        public void actionPerformed(ActionEvent ae) {
                if (ae.getSource() == btnResult) {
                        int num = Integer.parseInt(tf1.getText()) * 2;
                        tf2.setText(String.valueOf(num));
                }
                if (ae.getSource() == ext) {
                        System.exit(0);
                }
        }
        public static void main(String args[]) {
                EventTest t = new EventTest("Event handling");
                t.setSize(300, 200);
                t.setVisible(true);
        }
}
```

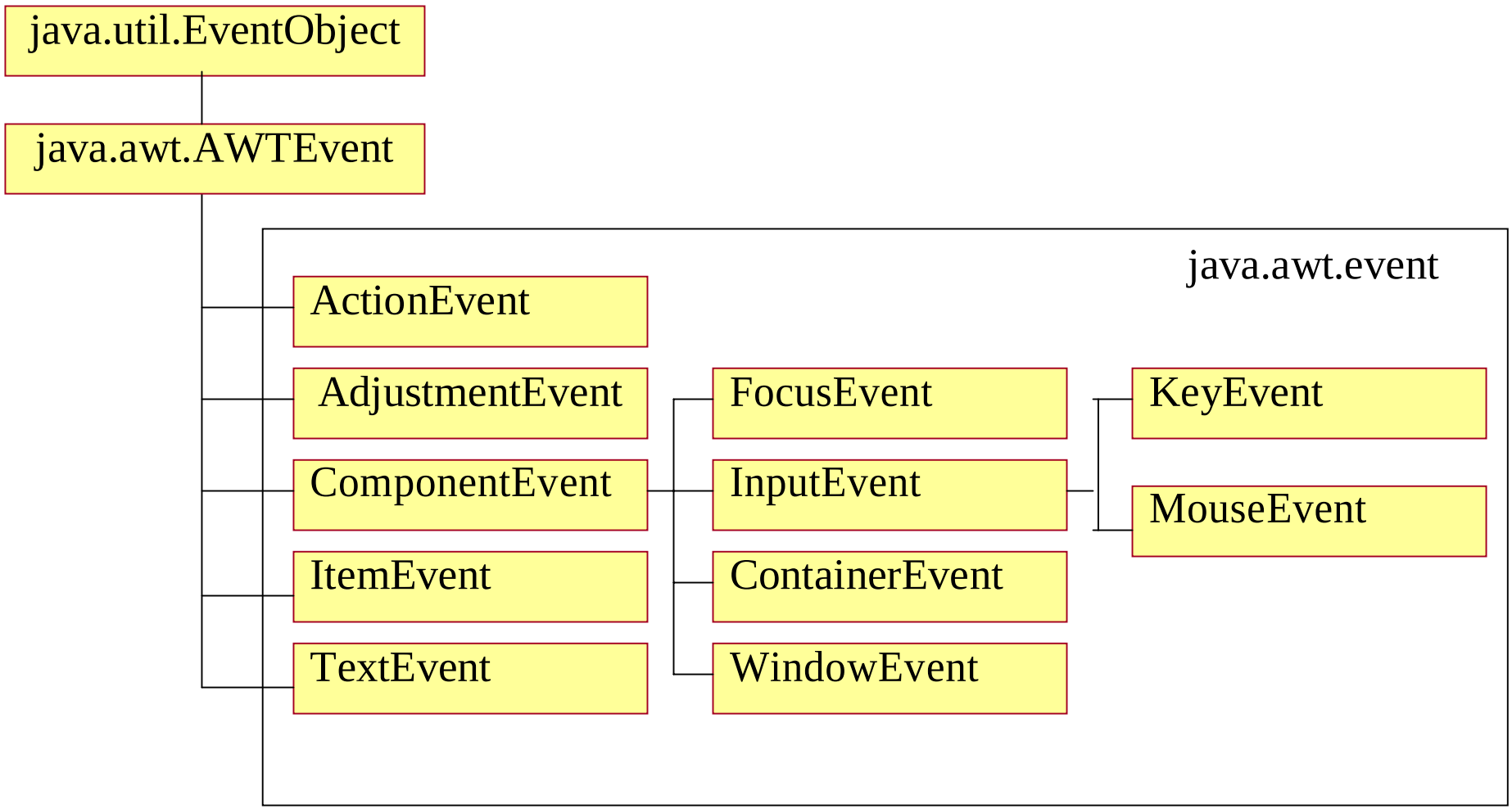
Xử lý cácsựkiện

- Ví dụ về xử lý sự kiện



Xử lý cácsựkiện

- Các loại sự kiện (Even



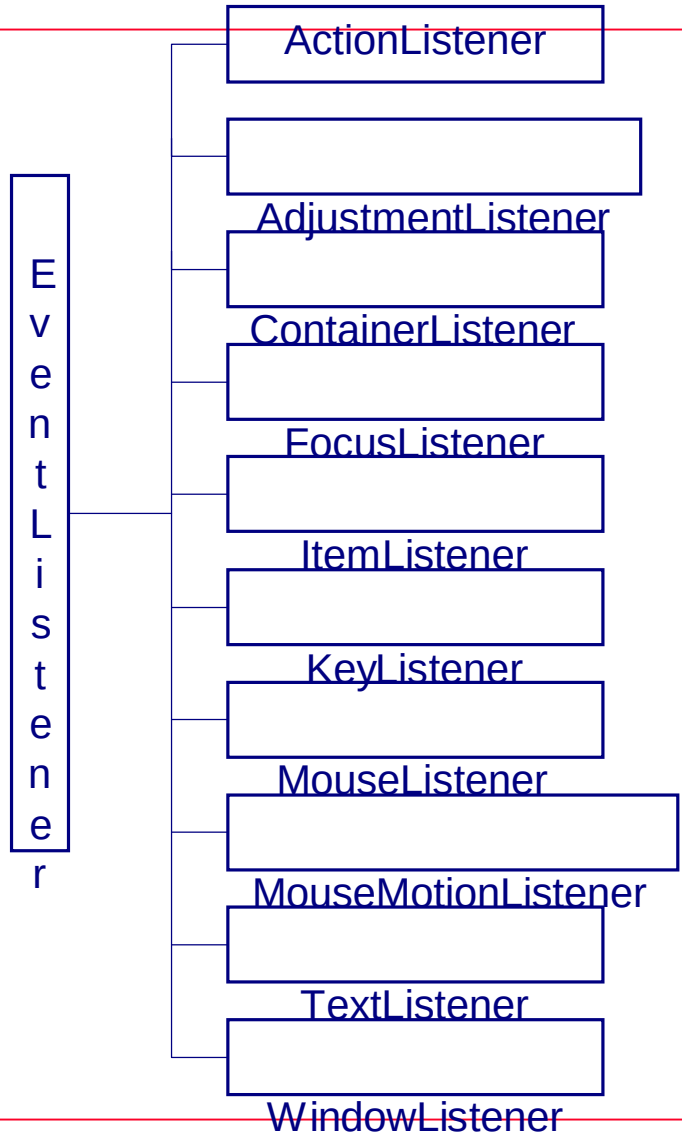
Xử lý cácsựkiện

③ Mô tả sự kiện

Lớp sự kiện	Mô tả
ActionEvent	Phát sinh khi một button được nhấn, một item trong danh sách chọn lựa được nhấn đúp (double-click) hay một menu được chọn.
AdjustmentEvent	Phát sinh khi một thanh scrollbar được sử dụng.
ComponentEvent	Phát sinh khi một thành phần được thay đổi kích thước, được di chuyển, bị ẩn hay làm cho hoạt động được.
FocusEvent	Phát sinh khi một thành phần mất hay nhận focus từ bàn phím.
ItemEvent	Phát sinh khi một mục menu được chọn hay bỏ chọn; hay khi một checkbox hay một item trong danh sách được click.
WindowEvent	Phát sinh khi một cửa sổ được kích hoạt, được đóng, được mở hay thoát.
TextEvent	Phát sinh khi giá trị trong thành phần textfield hay textarea bị thay đổi.
MouseEvent	Phát sinh khi chuột di chuyển, được click, được kéo hay thả ra.
KeyEvent	Phát sinh khi bàn phím ấn, nhả.

Xử lý cácsựkiện

- Các loại Listener



Xử lý cácsựkiện

- Ví dụ với danh sách

- Đăng ký đối tượng nghe
  - add + loại sự kiện + Listener(sự kiện)
  - Ví dụ với nút Button

    addActionListener(ActionListener)

List

addActionListener(A

ctionListener)

addItemListener(Ite
mListener)

Xử lý cácsựkiện

Cài đặt quản lý sự kiện

- Xác định đối tượng sẽ gây ra sự kiện (source)
- Xác định sự kiện có thể xẩy ra tương ứng với đối tượng mà ta cần quản lý (object)
- Xác định đối tượng “nghe” (listener) và cài đặt các phương thức tương ứng
- Đăng ký đối tượng nghe cho đối tượng gây ra sự kiện

- Ví dụ về quản lý sự kiện

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
public class TestButton {
        private Frame f;
        private Button b;
        public TestButton(){
                f = new Frame("Test");
                b = new Button("Press me");
                b.setActionCommand("ButtonPressed");
        }
        public void init(){
                b.addActionListener(new ButtonHandler());
                f.add(b, BorderLayout.CENTER);
                f.pack();
                f.setVisible(true);
```

}

Xử lý cácsự kiện

- Ví dụ về quản lý sự kiện

```
        public static void main(String[] args){
                TestButton test = new TestButton();
                test.init();
        }
}

class ButtonHandler implements ActionListener{
        public void actionPerformed(ActionEvent e) {
                System.out.println("Button's command is: "+
e.getActionCommand());
        }
}
```

③ Ví dụ về quản lý sự kiện


```
Button's command is: ButtonPressed
Button's command is: ButtonPressed
Button's command is: ButtonPressed
```

- Frame dùng để test các thành phần khác

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

public class ComponentTestFrame extends Frame implements
WindowListener {

        public ComponentTestFrame(String title){
                super(title);
                setBackground(SystemColor.control);
                setSize(400,300);
                setLocation(200,150);
                setLayout(new FlowLayout());
                addWindowListener(this);
        }
```

- Frame dùng để test các thành phần khác

```
        public void windowClosing(WindowEvent e){
                dispose();
                System.exit(0);
        }
        public void windowActivated(WindowEvent e){}
        public void windowClosed(WindowEvent e){}
        public void windowIconified(WindowEvent e){}
        public void windowDeiconified(WindowEvent e){}
        public void windowDeactivated(WindowEvent e){}
        public void windowOpened(WindowEvent e){}
}
```

- Một số phương thức của Frame
  - Frame()
  - Frame(String)
  - Image getIconImage()
  - MenuBar getMenuBar()
  - String getTitle()
  - Boolean isResizeable()
  - setIconImage(Image)
  - setMenuBar(MenuBar)
  - setTitle(String)
  - setVisible(boolean)

③ GUIFrame

```
import java.awt.*;
import java.awt.event.*;
public class GUIFrame extends Frame {
        public GUIFrame(String title){
                 super(title);
                setBackground(SystemColor.control);
                addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e){
                                dispose();
                                System.exit(0);
                        }
                });
        }
```

③ GUIFrame

```
        public void setVisible(boolean visible){
                if(visible){
                        Dimension d =
                        Toolkit.getDefaultToolkit().getScreenSize();
                        setLocation((d.width - getWidth())/2, (d.height
                                -getHeight())/2);
                }
                super.setVisible(visible);
        }

        public static void main(String[] args){
                GUIFrame frame = new GUIFrame("GUI Frame");
                frame.setSize(400,300);
                frame.setVisible(true);
        }
}
```

- GUIFrame

Cácthànhphầ nAWT

- Label
  - Dùng để hiển thị một đoạn văn bản trong một Container
  - Các phương thức khởt tạo
    - Label()
    - Label(String text)
    - Label(String text, alignment): alignment có thể nhận các giá trị Label.LEFT, Label.RIGHT, Label.CENTER
  - Phương thức khác
    - setFont(Font f)
    - setText(String s)
    - getText()
    - getAlignment()

③ Label

```
import java.awt.*;
public class LabelTest {
    public LabelTest() {
            Label l1 = new Label("Label");
            Label l2 = new Label("I am a label");
            l2.setFont(new Font("Timesroman", Font.BOLD, 18));
            Label l3 = new Label();
            l3.setText("I am disable");
            l3.setEnabled(false);
            Label l4 = new Label("Colored, right aligned", Label.RIGHT);
            l4.setForeground(Color.green);
            l4.setBackground(Color.black);
            ComponentTestFrame frame = new
                                    ComponentTestFrame("Label Test");
```

- Label


```
            frame.add(l1);
            frame.add(l2);
            frame.add(l3);
            frame.add(l4);
            frame.setVisible(true);
    }

    public static void main(String[] args) {
       LabelTest lt = new LabelTest();
    }
}
```

TextComponent

- Là lớp cha của TextField và TextArea
- Một số phương thức của TextComponent
  - getCaretPosition()
  - getSelectedText()
  - getSelectionStart()
  - getSelectionEnd()
  - getText(), setText()
  - select(int, int)
  - setCaretPosition(int)
  - setEditable(boolean)
  - setSelectionStart(int)
  - setSelectionEnd(int)

Cácthànhphầ nAWT

- Một số phương thức

- TextField
  - Chỉ chứa một dòng văn bản
    - TextField()
    - TextField(int columns)
    - TextField(String s)
    - TextField(String s, int columns)
    - addActionListener(ActionListener)
    - echoCharIsSet()
    - setEchoChar(char)
    - setText()
    - setColumn(int)

- TextField
  - Một số phương thức
    - setEditable(boolean): đặt chế độTextField có soạn thảo được hay không
    - isEditable(): xác định xem có ở chế độ Editable không

- TextField

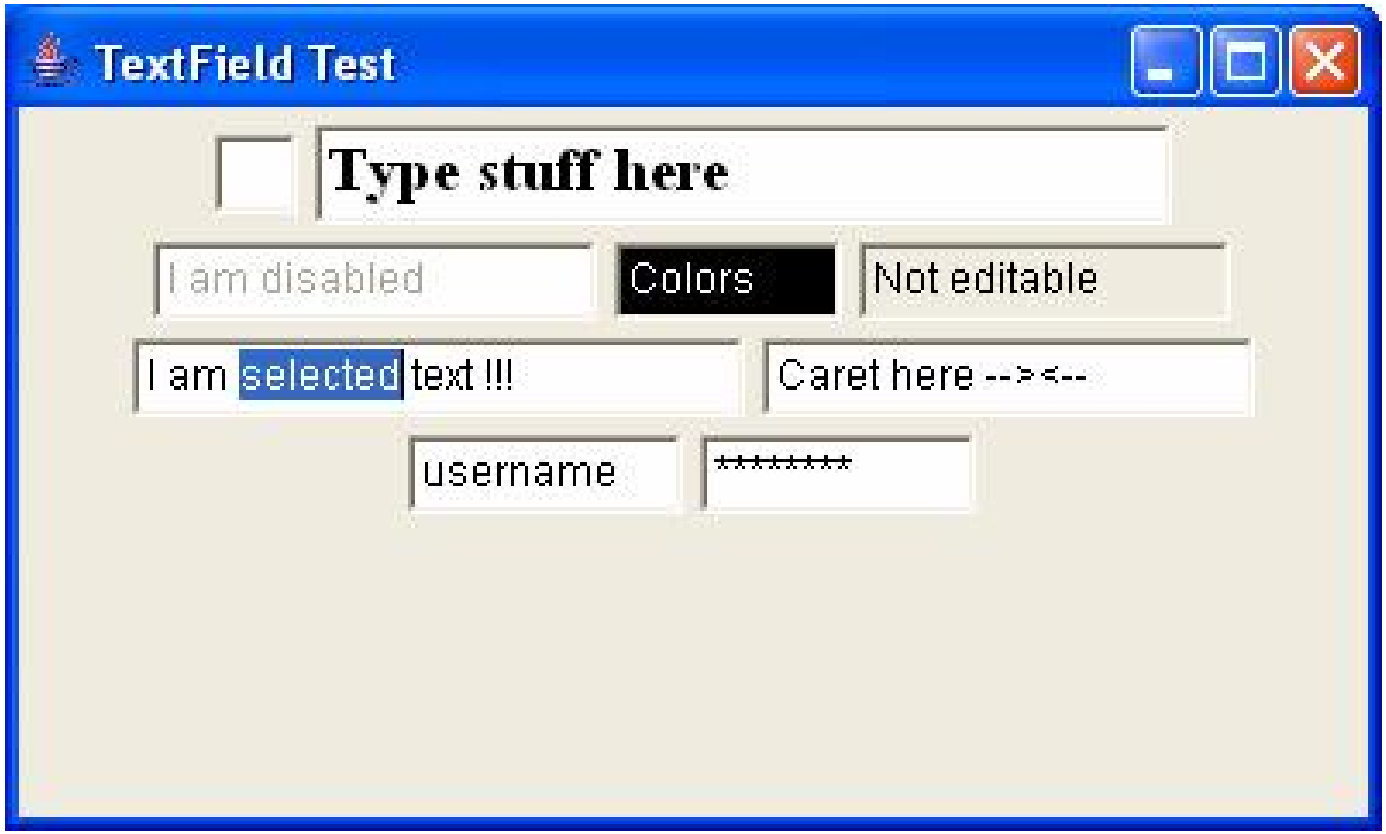
```
import java.awt.*;
public class TextFieldTest {
        public TextFieldTest() {
                super();
                TextField tf1 = new TextField();
                TextField tf2 = new TextField(25);
                tf2.setText("Type stuff here");
                tf2.setFont(new Font("Timesroman",Font.BOLD,18));
                TextField tf3 = new TextField("I am disabled",15);
                tf3.setEnabled(false);
                TextField tf4 = new TextField("Colors");
                tf4.setBackground(Color.BLACK);
                tf4.setForeground(Color.WHITE);
                TextField tf5 = new TextField("Not editable");
                tf5.setEditable(false);
                TextField tf6 = new TextField("I am selected text !!!");
```

```
tf6.select(5, 13);
```

- TextField

```
                TextField tf7 = new TextField("Caret here --><--");
                TextField tf8 = new TextField("username",8);
                TextField tf9 = new TextField("password",8);
                tf9.setEchoChar('*');
                ComponentTestFrame frame = new
                        ComponentTestFrame("TextField Test");
                frame.add(tf1);    frame.add(tf2);    frame.add(tf3);
                frame.add(tf4);    frame.add(tf5);    frame.add(tf6);
                frame.add(tf7);    frame.add(tf8);    frame.add(tf9);
                frame.setVisible(true);
                tf7.setCaretPosition(14);
        }
        public static void main(String[] args) {
                TextFieldTest test = new TextFieldTest();
        }
}
```

- TextField


TextField Test
Type stuff here
I am disabled
Colors
Not editable
I am selected text !!!
Caret here --><--
username
****

- TextArea
  - Hiển thị văn bản có nhiều hơn một dòng
  - Mỗi TextArea có một Scrollbar
  - Một số phương thức khởi tạo
    - TextArea()
    - TextArea(int rows, int columns)
    - TextArea(String text)
    - TextArea(String text, int rows, int columns)
    - TextArea(String text, int rows, int columns, int ScrollType)

- TextArea
  - Một số phương thức thường dùng
    - setText/getText
    - get/set row/column
    - setEditable/isEditable
    - append(String)
    - insert(String s, int i): chèn chuỗi vào một vị trí
    - replaceRange(String, int, int): thay thế văn bản nằm giữa vị trí int và int cho trước

③ TextArea

```
import java.awt.*;
public class TextAreaTest {
        public TextAreaTest() {
                super();
                TextArea ta1 = new TextArea(10,20);
                TextArea ta2 = new TextArea("Text Area\n with
                        color",10,10,TextArea.SCROLLBARS_NONE);
                ta2.setFont(new Font("Timesroman",Font.ITALIC,12));
                ta2.setBackground(Color.BLACK);
                ta2.setForeground(Color.GREEN);
                TextArea ta3 = new TextArea("This textarea is not
                        editable",
                10,15,TextArea.SCROLLBARS_HORIZONTAL_ONLY);
                ta3.setEditable(false);
                TextArea ta4 = new TextArea("This textarea is not enable",
                        4,25,TextArea.SCROLLBARS_NONE);
```

③ TextArea

```
                ta4.setEditable(false);
                ComponentTestFrame frame = new
                        ComponentTestFrame("TextArea Test");
                frame.add(ta1);    frame.add(ta2);
                frame.add(ta3);    frame.add(ta4);
                frame.setVisible(true);
        }
        public static void main(String[] args) {
                TextAreaTest test = new TextAreaTest();
        }
}
```

- TextArea


TextArea Test
This is a free text
Text Area
with color
This textarea is not ed
This textarea is not enable

Cácthànhphầ nAWT

- Button
  - Tương tác với người dùng
  - Thực hiện một hành động nào đó khi người dùng nhấn nút
  - Một số phương thức
    - Button()
    - Button(String text)
    - addActionListener(ActionListener)
    - String getLabel()
    - setLabel(String)
    - removeActionListener(ActionListener)

③ Button

```
import java.awt.Frame;

import java.awt.*;
public class ButtonTest {
    public ButtonTest() {
            Button b1 = new Button("Button");
            Button b2 = new Button();
            b2.setLabel("Press me!");
            b2.setFont(new Font("Timesroman", Font.BOLD, 16));
            Button b3 = new Button("Can't press me");
            b3.setEnabled(false);
            Button b4 = new Button("Colors");
            b4.setForeground(Color.green);
            b4.setBackground(Color.black);
            ComponentTestFrame frame = new
                        ComponentTestFrame("Button Test");
```

- Button


```
            frame.add(b1);
            frame.add(b2);
            frame.add(b3);
            frame.add(b4);
            frame.setVisible(true);
    }

    public static void main(String[] args) {
      ButtonTest lt = new ButtonTest();
    }
}
```

- Checkbox và RadioButton
  - Checkbox cho phép người dùng chọn một hay nhiều tùy chọn
  - Có thể chọn nhiều option của Checkbox
  - RadioButton cũng giống như Checkbox nhưng chỉ cho phép chọn một option tại một thời điểm
  - Thành phần Checkbox có thể dùng một lớp phụ (CheckboxGroup để tạo RadioButton)
  - Một số phương thức
    - Checkbox()
    - Checkbox(String)
    - Checkbox(String, boolean)
    - Checkbox(String, boolean, CheckboxGroup)

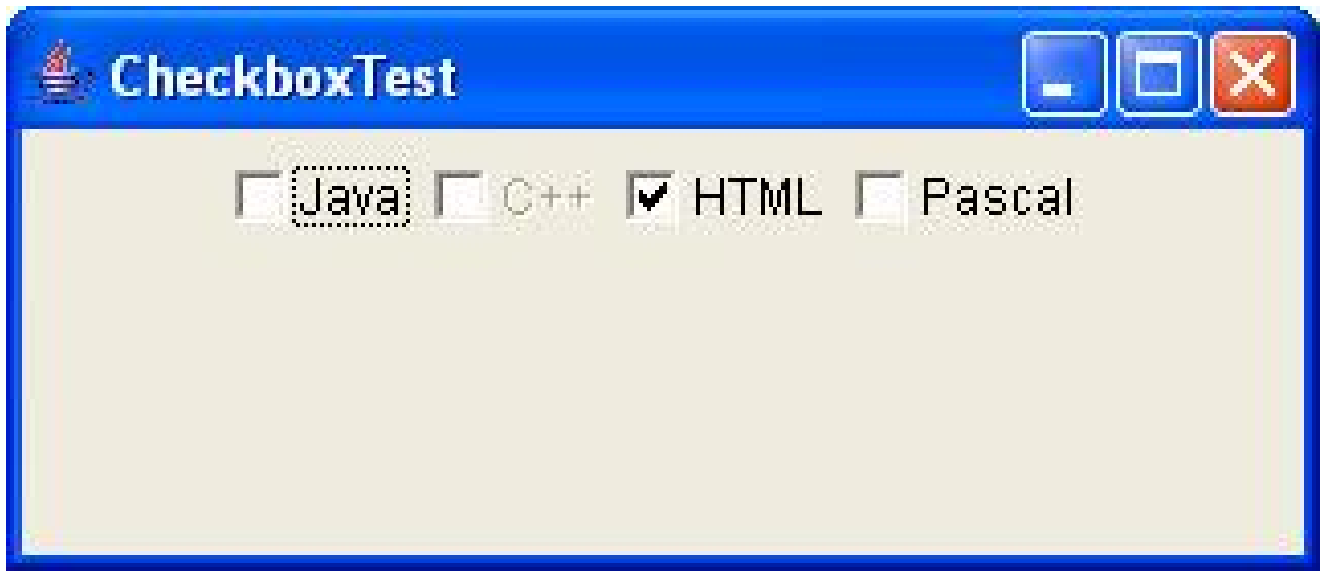
- Checkbox và RadioButton
  - Một số phương thức
    - addItemListener(ItemListener)
    - set/getCheckboxGroup()
    - set/getLabel()
    - set/getState()
    - removeItemListener(ItemListener)

③ Checkbox và RadioButton

```
import java.awt.Checkbox;
public class CheckboxTest {
        public CheckboxTest() {
                super();
                Checkbox cb1 = new Checkbox("Java",false);
                Checkbox cb2 = new Checkbox("C++",false);
                cb2.setEnabled(false);
                Checkbox cb3 = new Checkbox("HTML",true);
                Checkbox cb4 = new Checkbox();
                cb4.setLabel("Pascal");     cb4.setState(false);
                ComponentTestFrame frame = new
                        ComponentTestFrame("CheckboxTest");
                frame.add(cb1);   frame.add(cb2);
                frame.add(cb3);   frame.add(cb4);
                frame.setVisible(true);
        }
```

③ Checkbox và RadioButton

```
        public static void main(String[] args) {
                CheckboxTest test = new CheckboxTest();
        }
}
```



③ Checkbox và RadioButton

```
import java.awt.*;
public class CheckboxGroupTest {
        public CheckboxGroupTest() {
                super();
                CheckboxGroup group = new CheckboxGroup();
                Checkbox cb1 = new Checkbox("Java",false,group);
                Checkbox cb2 = new Checkbox("C++",false,group);
                cb2.setEnabled(false);
                Checkbox cb3 = new Checkbox("HTML",true,group);
                Checkbox cb4 = new Checkbox("",true,group);
                cb4.setLabel("Pascal");
                ComponentTestFrame frame = new
                    ComponentTestFrame("CheckboxGroupTest");
                frame.add(cb1);   frame.add(cb2);
                frame.add(cb3);   frame.add(cb4);
```

③ Checkbox và RadioButton

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

Cácthànhphầ̀nAWT

Choice

- Cho phép lựa chọn trên một danh sách các thành phần
- Một số phương thức
  - Choice()
  - add(String)
  - addItem(String)
  - addItemListener(ItemListener)
  - getItem(int)
  - getItemCount()
  - getSelectedIndex()
  - insert(String, int)

- Choice
  - Một số phương thức
    - removeItemListener(ItemListener)
    - select(int)
    - select(String)
    - remove(int)
    - remove(String)
    - removeAll()

- Choice

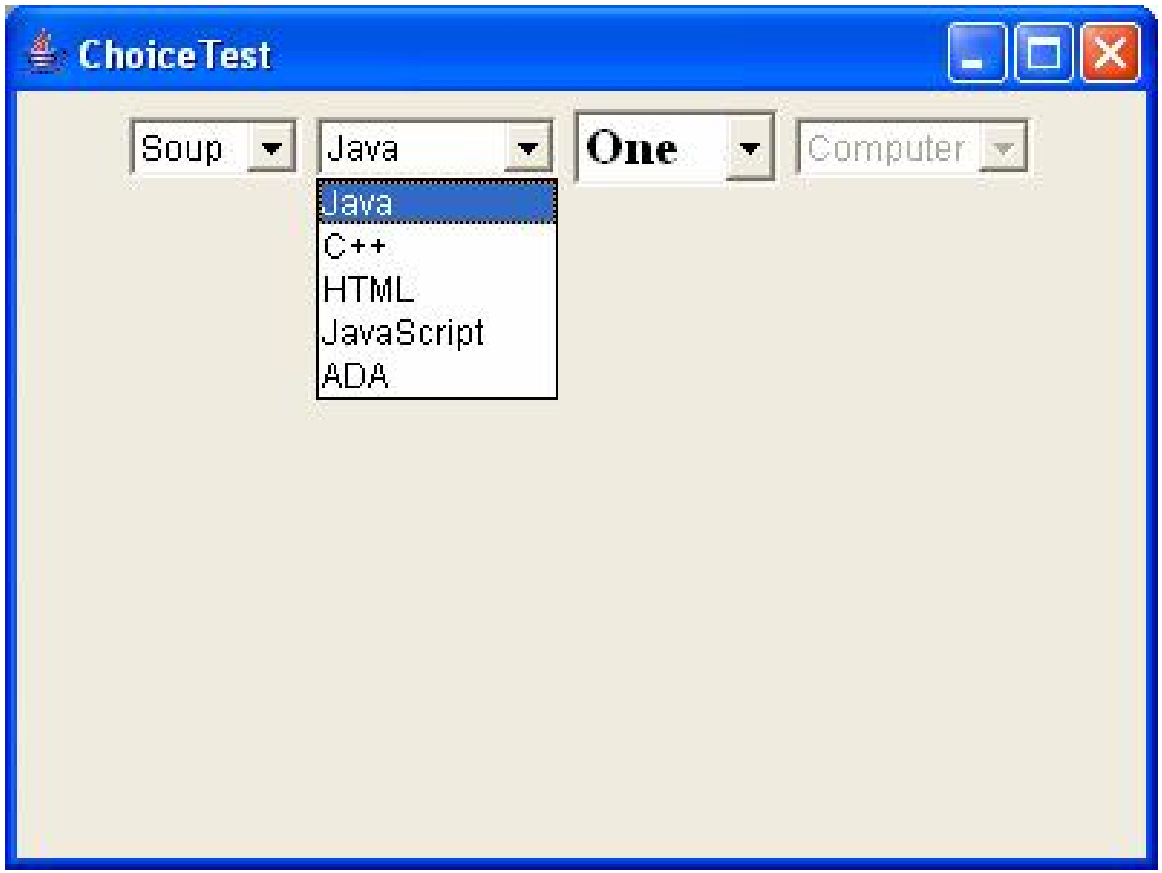
```
import java.awt.*;
public class ChoiceTest {
    public ChoiceTest() {
        super();

Choice c1 = new Choice();
c1.add("Soup");
c1.add("Salad");
Choice c2 = new Choice();
c2.add("Java");
c2.add("C++");
c2.add("HTML");
c2.add("JavaScript");
c2.add("ADA");
Choice c3 = new Choice();
c3.add("One");
c3.add("Two");
c3.add("Three");
```

- Choice

```
                c3.setFont(new Font("Timesroman",Font.BOLD,18));
                Choice c4 = new Choice();
                c4.add("Computer");
                c4.setEnabled(false);
                ComponentTestFrame frame = new
                        ComponentTestFrame("ChoiceTest");
                frame.add(c1);
                frame.add(c2);
                frame.add(c3);
                frame.add(c4);
                frame.setVisible(true);
        }
        public static void main(String[] args) {
                ChoiceTest test = new ChoiceTest();
        }
}
```

Cácthànhphầ nAWT

③ Choice



Cácthànhphầ nAWT

List

- Giống như Choice nhưng cho phép hiện thị nhiều lựa chọn cùng lúc
- Người dùng có thể lựa chọn nhiều item
- Tham khảo API Documentation

Canvas

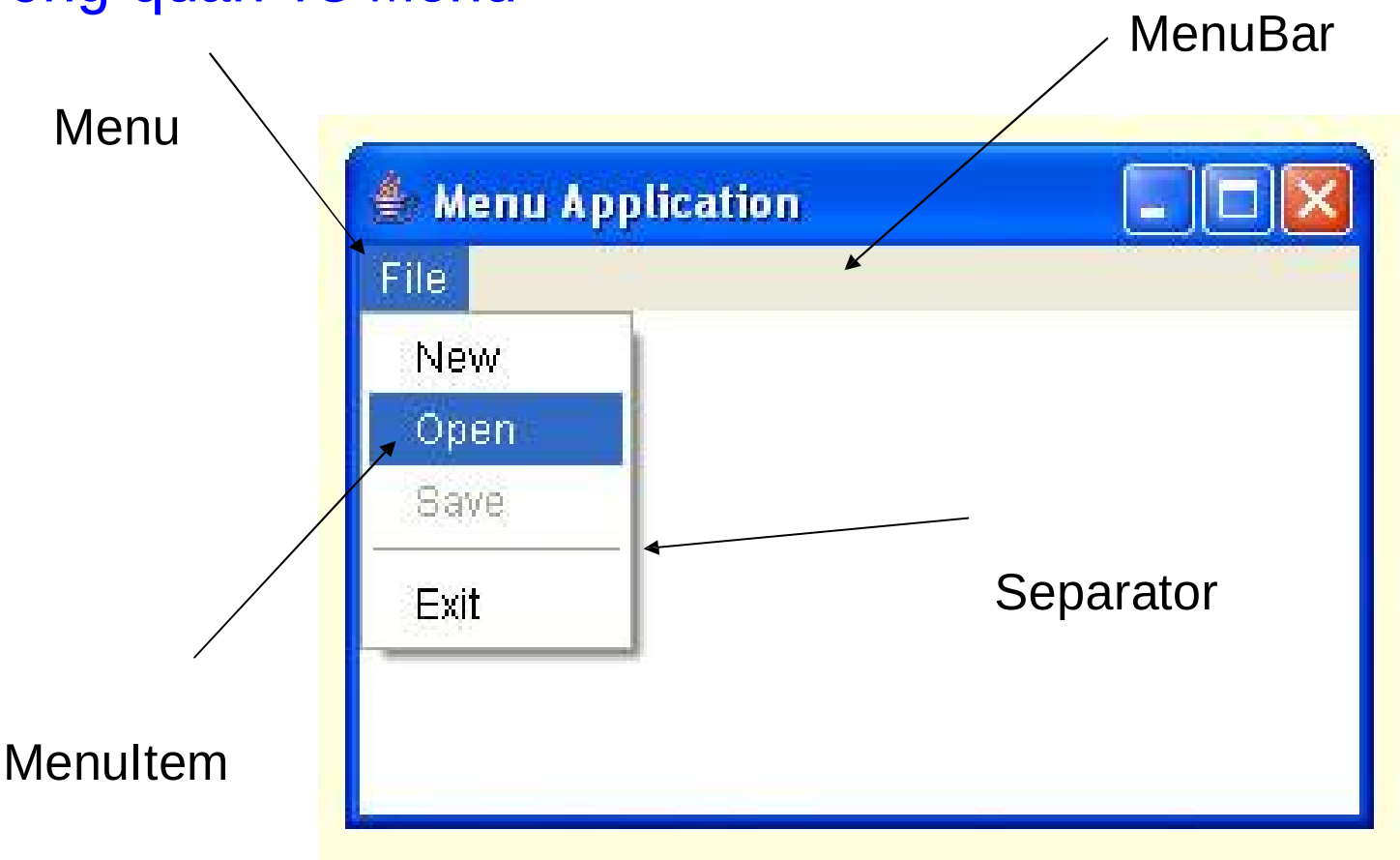
- Là khu vực cho phép hiển thị hình ảnh hoặc nhận về các sự kiện của người dùng
- Muốn vẽ bất cứ thứ gì trên Canvas ta phải cài đặt lại phương thức paint(Graphics)
- Ví dụ

```
import java.awt.*;
public class CanvasTest extends Canvas {

        public void paint(Graphics g){
                setFont(new Font("Arial", Font.BOLD + Font.ITALIC, 16));
                g.drawString("Canvas", 15, 25);
        }
```

③ Canvas

```
        public static void main(String[] args) {
                CanvasTest c1 = new CanvasTest();
                c1.setSize(100,100);
                CanvasTest c2 = new CanvasTest();
                c2.setSize(100,100);
                c2.setBackground(Color.orange);
                c2.setForeground(Color.blue);
                CanvasTest c3 = new CanvasTest();
                c3.setSize(200,50);
                c3.setBackground(Color.white);
                ComponentTestFrame frame = new ComponentTestFrame
                                                ("Canvas Test");
                frame.add(c1);frame.add(c2);frame.add(c3);
                frame.pack();
                frame.setVisible(true);
        }
}
```

③ Tổng quan về Menu



- Tổng quan về Menu

  MenuBar menuBar = new MenuBar();

  - Gắn MenuBar vào cửa sổ

    myFrame.setMenuBar(menuBar);

  - Tạo Menu

    Menu fileMenu = new Menu(“File”);

  - Gắn Menu vào MenuBar

    menuBar.add(fileMenu);

  - Tạo MenuItem

    MenuItem fileOpen = new MenuItem(“Open”);

- Tổng quan về Menu
  - Gắn MenuItem vào Menu

```
fileMenu.add(fileOpen);
```

- Tạo đường phân cách

```
fileMenu.addSeparator();
```

Cácthànhphầ nAWT

- MenuItem
  - MenuItem()
  - MenuItem(String)
  - MenuItem(String, MenuShortcut)
  - addActionListener(ActionListener)
  - set/getLabel();
  - get/setShortcut(MenuShortcut)
  - isEnabled()
  - setEnabled()

③ Ví dụ Menu

```
import java.awt.*;
import java.awt.event.KeyEvent;
public class MenuTest {
        public MenuTest() {
                super();
                MenuItem fileNew = new MenuItem("New");
                fileNew.setShortcut(new
MenuShortcut(KeyEvent.VK_N));
                MenuItem fileOpen = new MenuItem("Open");
                fileOpen.setShortcut(new
MenuShortcut(KeyEvent.VK_O));
                MenuItem fileSave = new MenuItem("Save");
                fileSave.setShortcut(new
MenuShortcut(KeyEvent.VK_S));
                fileSave.setEnabled(false);
                MenuItem fileSaveAs = new MenuItem("Save As");
```

③ Ví dụ Menu

```
                fileSaveAs.setShortcut(new
MenuShortcut(KeyEvent.VK_A));
                fileSaveAs.setEnabled(false);
                MenuItem fileExit = new MenuItem("Exit");
                fileExit.setShortcut(new
MenuShortcut(KeyEvent.VK_X));
                MenuItem editCut = new MenuItem("Cut");
                editCut.setShortcut(new
MenuShortcut(KeyEvent.VK_X));
                MenuItem editCopy = new MenuItem("Copy");
                editCopy.setShortcut(new
MenuShortcut(KeyEvent.VK_C));
                MenuItem editPaste = new MenuItem("Paste");
                editPaste.setShortcut(new
MenuShortcut(KeyEvent.VK_P));
                Menu file = new Menu("File");
```

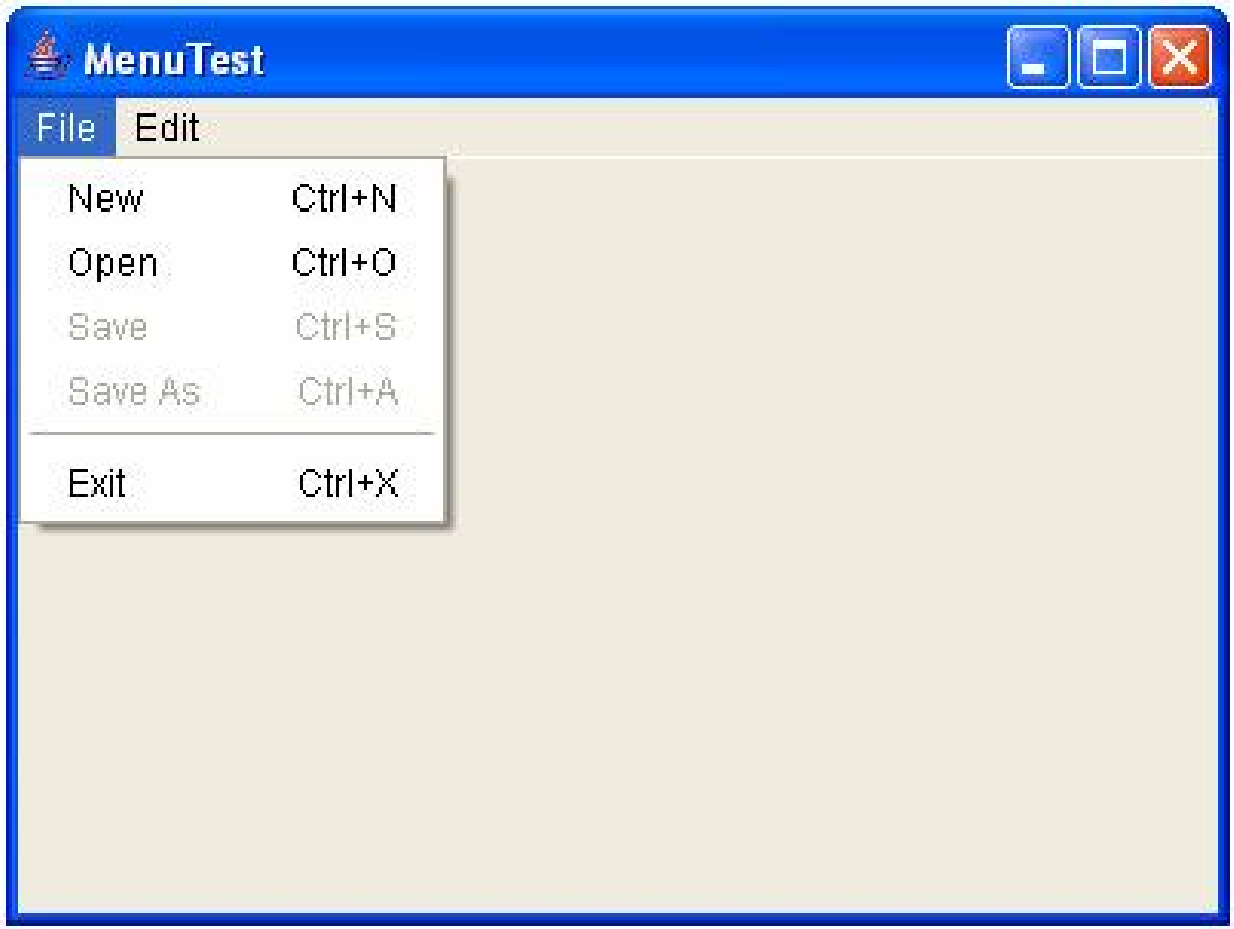
- Ví dụ Menu

```
file.add(fileNew);file.add(fileOpen);
file.add(fileSave);file.add(fileSaveAs);
file.addSeparator();file.add(fileExit);
Menu edit = new Menu("Edit");
edit.add(editCut); edit.add(editCopy);
edit.add(editPaste);
MenuBar menuBar = new MenuBar();
menuBar.add(file); menuBar.add(edit);
ComponentTestFrame frame = new
        ComponentTestFrame("MenuTest");
frame.setMenuBar(menuBar);
frame.setVisible(true);

}
public static void main(String[] args) {
        MenuTest test = new MenuTest();
}
```

```
}
```

Cácthànhphầ̀nAWT

- Ví dụ Menu



- Là lớp con của Menu

- PopupMenu
  - Đối tượng PopupMenu có thể hiển thị mà không cần MenuBar
  - Có thể hiển thị ở vị trí bất kỳ theo yêu cầu

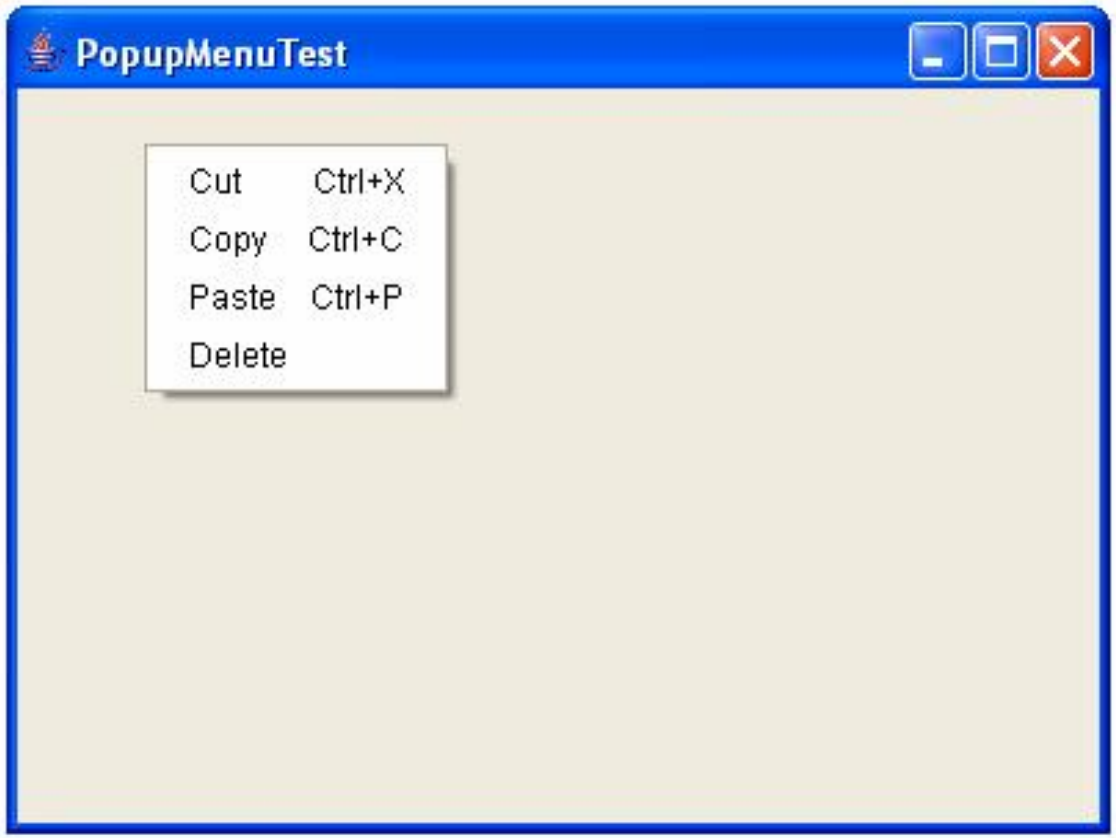
```
import java.awt.*;
import java.awt.event.KeyEvent;
public class PopupMenuTest {
        public PopupMenuTest() {
                super();
                MenuItem cut = new MenuItem("Cut");
                cut.setShortcut(new MenuShortcut(KeyEvent.VK_X));
                MenuItem copy = new MenuItem("Copy");
                copy.setShortcut(new
MenuShortcut(KeyEvent.VK_C));
                MenuItem paste = new MenuItem("Paste");
```

③ PopupMenu

```
                paste.setShortcut(new
MenuShortcut(KeyEvent.VK_P));
                MenuItem delete = new MenuItem("Delete");
                PopupMenu popupMenu = new
PopupMenu("Clipboard");
                popupMenu.add(cut);popupMenu.add(copy);
                popupMenu.add(paste);popupMenu.add(delete);
                ComponentTestFrame frame = new
                        ComponentTestFrame("PopupMenuTest");
                frame.add(popupMenu);frame.setVisible(true);
                popupMenu.show(frame,50,50);
        }
        public static void main(String[] args) {
                PopupMenuTest test = new PopupMenuTest();
        }
}
```

Cácthànhphầ nAWT

- ③ PopupMenu



- Panel
  - Là một Container nhưng không thể tồn tại độc lập
  - Phải gắn Panel vào một Frame
  - Dùng Panel để nhóm các thành phần GUI lại với nhau

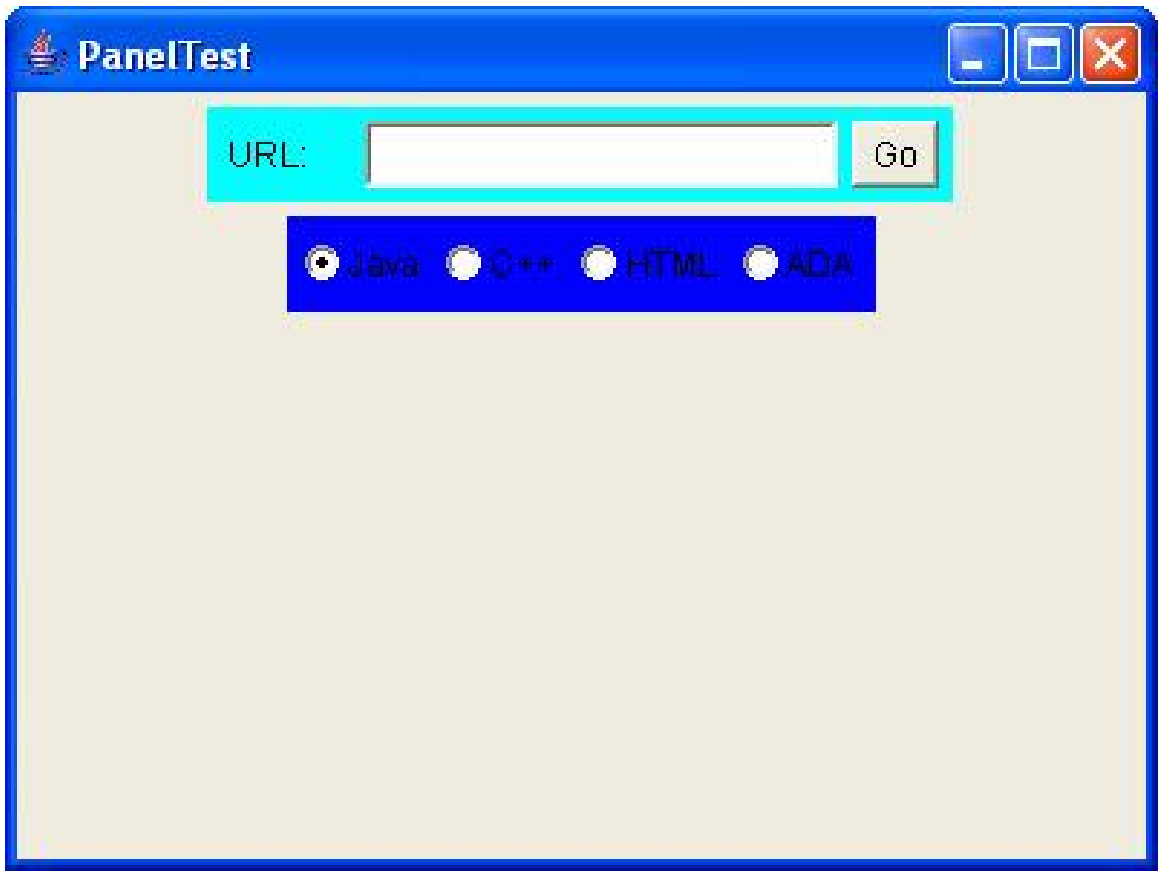
```
import java.awt.*;
public class PanelTest {
        public PanelTest() {
                super();
                Label lb1 = new Label("URL: ");
                TextField tf1 = new TextField("",20);
                Button b1 = new Button("Go");
                Panel p1 = new Panel();
                p1.setBackground(Color.CYAN);
                p1.add(lb1); p1.add(tf1); p1.add(b1);
                CheckboxGroup group = new CheckboxGroup();
```

- Panel

```
                Checkbox cb1 = new Checkbox("Java",group,true);
                Checkbox cb2 = new Checkbox("C++",group,false);
                Checkbox cb3 = new Checkbox("HTML",group,false);
                Checkbox cb4 = new Checkbox("ADA",group,false);
                Panel p2 = new Panel();
                p2.setBackground(Color.BLUE);
                p2.add(cb1); p2.add(cb2); p2.add(cb3); p2.add(cb4);
                ComponentTestFrame frame = new
                        ComponentTestFrame("PanelTest");
                frame.add(p1); frame.add(p2);
                frame.setVisible(true);
        }
        public static void main(String[] args) {
                PanelTest test = new PanelTest();
        }
}
```

Cácthànhphầ nAWT

- Ví dụ về Panel



- Dialog
  - Là cửa sổ có thanh tiêu đề, đường biên
  - Thường dùng để cho phép nhập dữ liệu
  - Một Dialog phải tồn tại trong một Frame hoặc Dialog khác
  - Một Dialog có thể là modal hoặc non-modal
    - Modal dialog
    - Non-modal dialog

- Dialog
  - Một số phương thức
    - Dialog(Dialog)
    - Dialog(Dialog, String)
    - Dialog(Dialog, String, boolean)
    - Dialog(Frame)
    - Dialog(Frame, String)
    - Dialog(Frame, String, boolean)

③ Ví dụ Dialog

```
import java.awt.*;
import java.awt.event.*;
public class DialogTest implements WindowListener {
        public DialogTest() {
                super();
                ComponentTestFrame frame = new
                        ComponentTestFrame("DialogTest");
                Dialog d1 = new Dialog(frame,"Modal Dialog Test",true);
                d1.add(new Label("Modal Dialog Test"));
                d1.addWindowListener(this);
                Dialog d2 = new Dialog(frame,"NonModal Dialog
Test",false);
                d2.addWindowListener(this);
                d2.add(new Label("NonModal Dialog Test"));
                frame.setLocation(100,100);
```

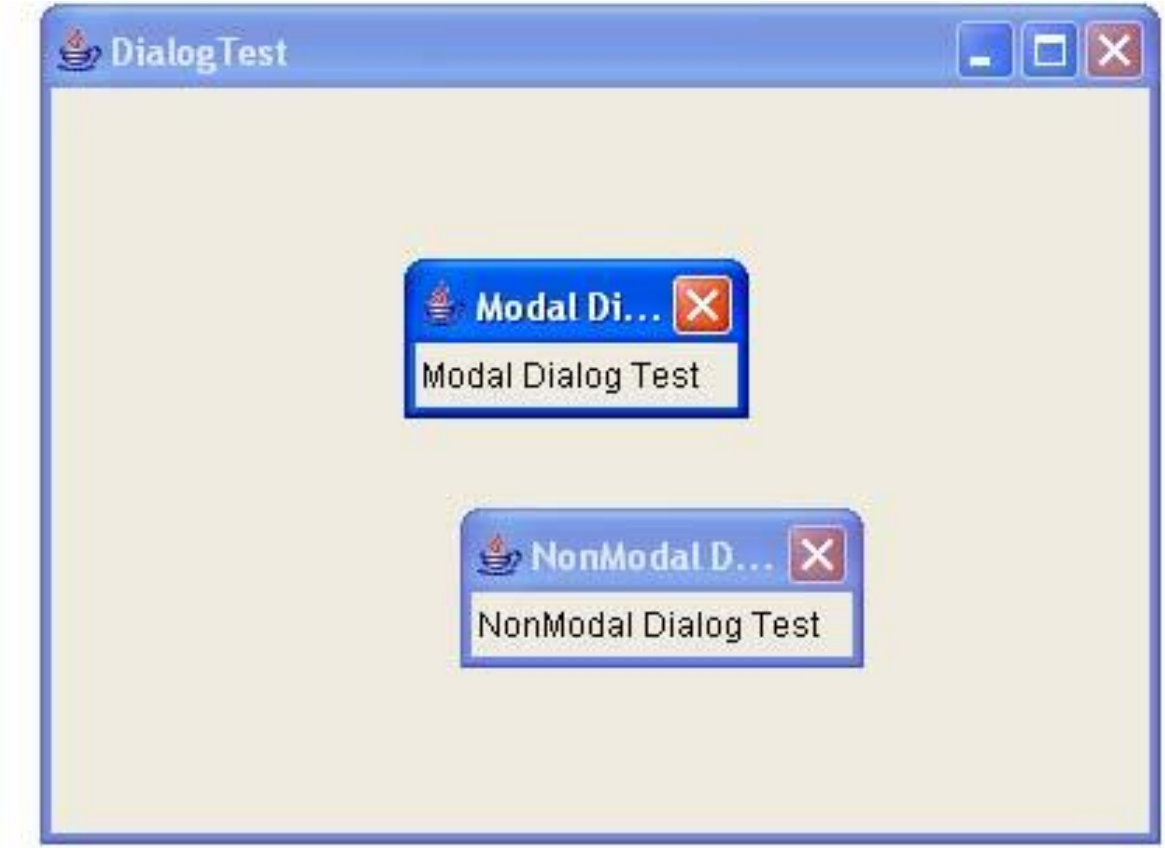
③ Ví dụ Dialog

```
            frame.setVisible(true);
            d2.pack();
            d2.setLocation(250,280);
            d2.setVisible(true);
            d1.pack();
            d1.setLocation(220,170);
            d1.setVisible(true);
        }
        public void windowOpened(WindowEvent arg0) {
        }
        public void windowClosing(WindowEvent arg0) {
            ((Dialog)arg0.getSource()).setVisible(false);
        }
        public void windowClosed(WindowEvent arg0) {
        }
```

③ Ví dụ Dialog

```
        public void windowIconified(WindowEvent arg0) {
        }
        public void windowDeiconified(WindowEvent arg0) {
        }
        public void windowActivated(WindowEvent arg0) {
        }
        public void windowDeactivated(WindowEvent arg0) {
        }
        public static void main(String[] args) {
                DialogTest test = new DialogTest();
        }
}
```

③ Ví dụ Dialog



Quản lý trình bày

- Layout manager: quản lý cách trình bày của các GUI components trong một Container
- Các layout manager
  - FlowLayout
  - BorderLayout
  - CardLayout
  - GridLayout
  - GridBagLayout
- Sử dụng phương thức setLayout() của một Container để thay đổi layout

Quản lý trình bày

- FlowLayout
  - Là layout mặc định cho Applet và Panel
  - Các thành phần được sắp xếp từ trái sang phải, trên xuống dưới
  - Các hàm khởi tạo
    - FlowLayout()
    - FlowLayout(int alignment)

      Alignment có thể là FlowLayout.CENTER, LEFT, RIGHT
    - FlowLayout(int, int, int)

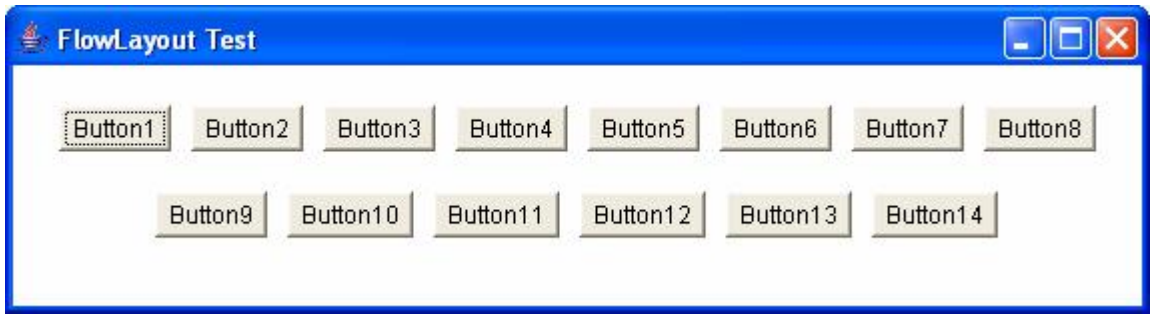
③ FlowLayout

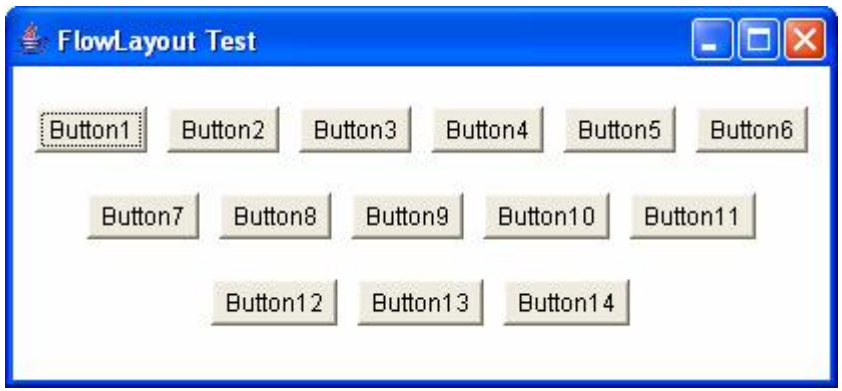
```
import java.awt.*;
import java.awt.event.*;
public class FlowLayoutTest extends Frame implements WindowListener {
        public FlowLayoutTest(String title){
                super(title);
                addWindowListener(this);
                setLayout(new FlowLayout(FlowLayout.CENTER, 20, 50));
                for(int i = 1; i<15; i++)
                        add(new Button("Button"+i));
                setSize(575, 300);
                setVisible(true);
        }
        public void windowClosing(WindowEvent e){
                dispose();
                System.exit(0);
        }
```

③ FlowLayout

```
        public void windowActivated(WindowEvent e){}
        public void windowClosed(WindowEvent e){}
        public void windowIconified(WindowEvent e){}
        public void windowDeiconified(WindowEvent e){}
        public void windowDeactivated(WindowEvent e){}
        public void windowOpened(WindowEvent e){}

        public static void main(String[] args) {
        FlowLayoutTest ft = new FlowLayoutTest("FlowLayout Test");
        }
}
```

③ FlowLayout





- BorderLayout
  - Là layout mặc định cho Frame, Window, Dialog
  - Các thành phần được sắp xếp trên 5 khu vực khác nhau
    - NORTH – Đặt ở đỉnh của container.
    - EAST – Đặt phía bên phải của container.x`
    - SOUTH – Đặt ở phía dưới của container.
    - WEST – Đặt phía bên trái của container.
    - CENTER – Đặt ở giữa của container.

  Để thêm một thành phần vào vùng 'North'

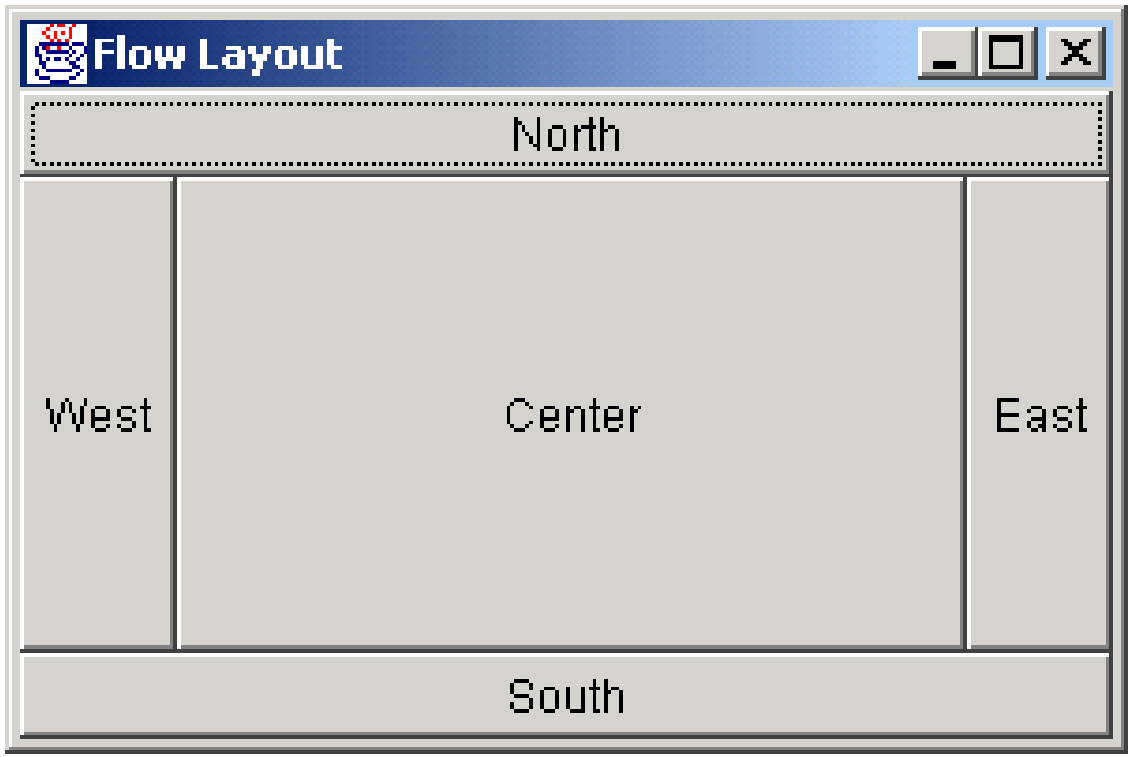
  Button b1=new Button("North Button"); // khai báo thành phần

  setLayout(new BorderLayout()); // thiết lập layout

  add(b1,BorderLayout.NORTH); // thêm thành phần vào layout

- Khởi tạo: BorderLayout(), BorderLayout(int, int)

- ③ BorderLayout



Quản lý trình bày

- CardLayout
  - Tự tìm hiểu trong API Documentation

- GridLayout
  - Chia Container thành các ô lưới bằng nhau
  - Các Component được đặt trong các ô
  - Mỗi ô lưới nên chứa ít nhất một thành phần
  - Ví dụ

    GridLayout l1 = new GridLayout(4, 3)
  - Nếu new GridLayout(0,3) nghĩa là ô có 3 cột, số hàng là bất kỳ
  - Không được sử dụng new GridLayout(0,0);
  - Các hàm khởi tạo:
    - GridLayout(int), GridLayout(int, int), GridLayout(int, int, int ,int)

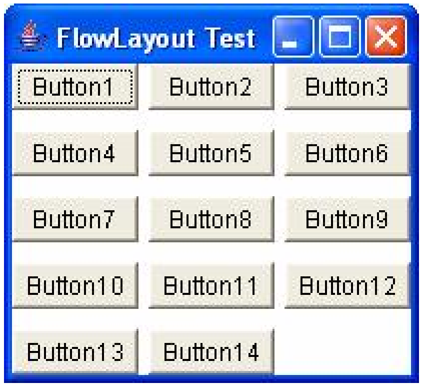
③ GridLayout

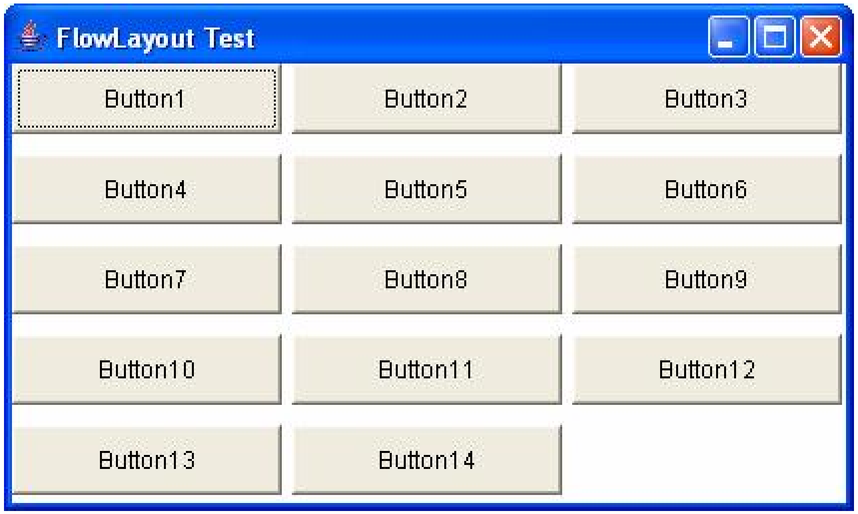
```
import java.awt.*;
import java.awt.event.*;
public class GridLayoutTest extends Frame implements WindowListener {

        public GridLayoutTest(String title){
                super(title);
                addWindowListener(this);
                setLayout(new GridLayout(0, 3, 5, 10));
                for(int i = 1; i<15; i++)
                        add(new Button("Button"+i));
                pack();
                setVisible(true);
        }
```

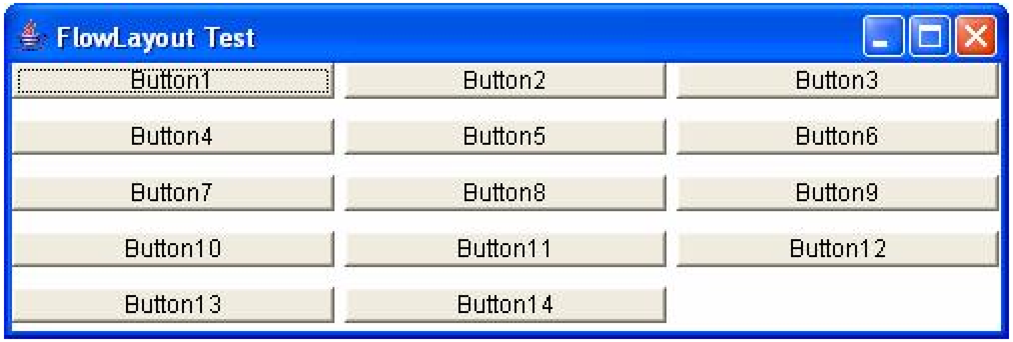
```
        public void windowClosing(WindowEvent e){
                dispose();
                System.exit(0);
        }

        public void windowActivated(WindowEvent e){}
        public void windowClosed(WindowEvent e){}
        public void windowIconified(WindowEvent e){}
        public void windowDeiconified(WindowEvent e){}
        public void windowDeactivated(WindowEvent e){}
        public void windowOpened(WindowEvent e){}

        public static void main(String[] args) {
                GridLayoutTest ft = new GridLayoutTest("GridLayout Test");
        }
}
```

Quảnlý trình bày

③ GridLayout







Quản lý trình bày

- GridBagLayout
  - Tự đọc

Xử lý cácsự kiện

- WindowEvent
  - Cài đặt giao tiếp WindowListener
  - Xem ví dụ về Frame

Xử lý cácsựkiện

- WindowEvent
  - Cài đặt giao tiếp WindowListener
  - Xem ví dụ về Frame
- Adapter class
  - WindowAdapter
  - FocusAdapter
  - KeyAdapter
  - MouseAdapter
  - MouseMotionAdapter

Xử lý cácsựkiện

ActionEvent

- Được phát sinh bởi Button, MenuItem, TextField, List
- Lớp nghe cài đặt giao tiếp ActionListener hay cài đặt phương thức actionPerformed(ActionEvent)
- Một số biến & phương thức của ActionEvent
  - int ALT_MASK: phím ALT có được nhấn ?
  - int CTRL_MASK: phím CTRL có được nhấn ?
  - int SHIFT_MASK: phím SHIFT có được nhấn ?
  - int getModifiers(): có thể trả về ALT_MASK, CTRL_MASK...
  - String getActionCommand(): trả về command gắn với mỗi ActionEvent

- ActionEvent

```
import java.awt.*;
import java.awt.event.*;
class ActionListenerTest extends GUIFrame implements ActionListener {
        Panel controlPanel, whoDoneItPanel, commandPanel;
        MenuBar menuBar;
        Menu menu;
        MenuItem menuItem;
        Button button;
        List list;
        Label whoDoneItLabel, commandLabel;
        TextField whoDoneItTextField, commandTextField, textField;

        public ActionListenerTest(){
                super("ActionListener Test");
                //create menu bar
                menuBar = new MenuBar();
                menu = new Menu("A Menu");
```

Xử lý cácsựkiện

③ ActionEvent

```
menuItem = new MenuItem("A Menu Item",new
            MenuShortcut(KeyEvent.VK_M));
menuItem.addActionListener(this);
menu.add(menuItem);
menuBar.add(menu);
setMenuBar(menuBar);
//create whoDoneItPanel
whoDoneItPanel = new Panel();
whoDoneItPanel.setBackground(Color.pink);
whoDoneItLabel = new Label("Who done it",

Label.RIGHT);   whoDoneItTextField = new TextField("A TextField");
                //whoDoneItTextField.addActionListener(this);
                whoDoneItTextField.setEditable(false);
                whoDoneItPanel.add(whoDoneItLabel);
                whoDoneItPanel.add(whoDoneItTextField);
                add(whoDoneItPanel,BorderLayout.NORTH);
```

Xử lý các sự kiện

③ ActionEvent

```
//create controlPanel
controlPanel = new Panel();
controlPanel.add(new Label("A TextField", Label.RIGHT));
textField = new TextField(15);
textField.addActionListener(this);
controlPanel.add(textField);
button = new Button("A Button");
button.addActionListener(this);
button.setActionCommand("My Action Commmand");
controlPanel.add(button);
controlPanel.add(new Label("A List",Label.RIGHT));
list = new List(5,false);
list.add("Breakfast");
list.add("Lunch");
list.add("Diner");
```

ck"); list.add("Dessert");

Xử lý cácsự kiện

③ ActionEvent

```
list.add("Brunch");
list.addActionListener(this);
controlPanel.add(list);
add(controlPanel, BorderLayout.CENTER);
//create commandPanel
commandPanel = new Panel();
commandLabel = new Label("Action Command");
commandPanel.setBackground(Color.pink);
commandTextField = new TextField(15);
commandTextField.setEditable(false);
commandPanel.add(commandLabel);
commandPanel.add(commandTextField);
add(commandPanel,BorderLayout.SOUTH);
pack();
setVisible(true);
```

Xử lý cácsựkiện

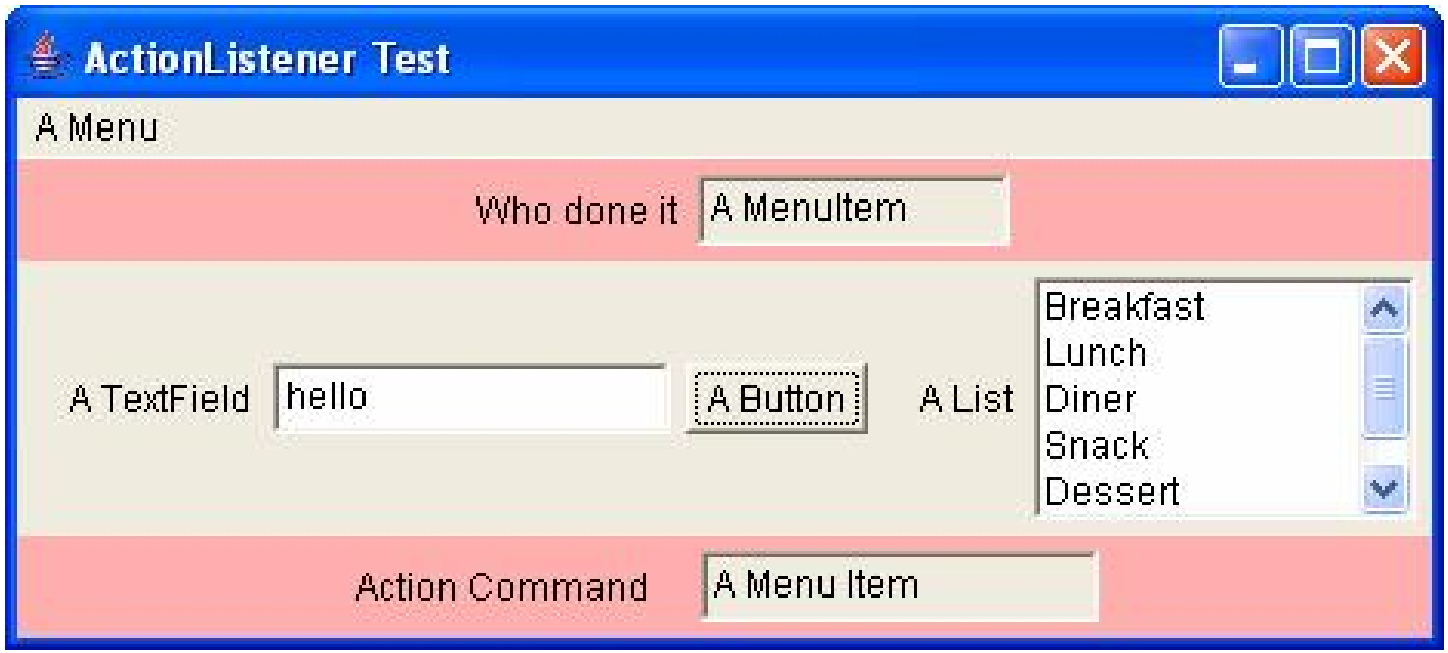
```
}
```

③ ActionEvent

```
        public void actionPerformed(ActionEvent e){
                if(e.getSource()==menuItem){
                        whoDoneItTextField.setText("A MenuItem");
                }else if(e.getSource()==textField){
                        whoDoneItTextField.setText("A TextField");
                }else if(e.getSource()==button){
                        whoDoneItTextField.setText("A Button");
                }else if(e.getSource()==list){
                        whoDoneItTextField.setText("A List");
                }
                commandTextField.setText(e.getActionCommand());
        }
        public static void main(String[] args){
                ActionListenerTest test = new ActionListenerTest();
        }
}
```

Xử lý cácsựkiện

- ActionEvent



Xử lý cácsựkiện

ItemEvent

- Được tạo ra từ các thành phần cho phép lựa chọn như Checkbox, Choice, List
- Lớp nghe ItemEvent cần cài đặt giao tiếp ItemListener
- Phương thức cần cài đặt: itemStateChanged(ItemEvent)
- Phương thức của ItemEvent
  - int getStateChange(): có thể nhận ItemEvent.SELECTED hoặc ItemEvent.DESELECTED
  - Object getItem(): item đã thay đổi trạng thái (Checkbox, Choice hoặc item được chọn của List)

Xử lý cácsựkiện

TextEvent

- Được tạo ra bởi TextComponent (TextField, TextArea)
- Lớp nghe cài đặt giao tiếp TextListener
- Phương thức cần cài đặt textValueChanged(TextEvent)
- TextEvent được sinh ra khi giá trị text của TextComponent thay đổi (thêm, xóa text)

Xử lý cácsựkiện

- MouseEvent
  - Được tạo ra bởi chuột của người dùng
  - Lớp nghe cài đặt giao tiếp
    - MouseListener
    - MouseMotionListener
  - Phương thức của MouseEvent
    - int getClickCount()
    - Point getPoint()
    - int getX()
    - int getY()

Xử lý cácsựkiện

- Các phương thức của MouseListener

- MouseEvent
  - void mouseClicked(MouseEvent)
  - void mouseEnteredMouseEvent)
  - void mouseExited(MouseEvent)
  - void mousePressed(MouseEvent)
  - void mouseReleased(MouseEvent)
- Các phương thức của MouseMotionListener
  - void mouseMoved(MouseEvent)
  - void mouseDragged(MouseEvent)

import java.awt.event.*;
public class MouseTest extend
GUIFrame

③ MouseEvent

```
import java.awt.*;
        implements MouseListener, MouseMotionListener {
        Canvas canvas;
        Label location, event;
        public MouseTest(){
                super("Mouse Event Test");
                canvas = new Canvas();
                canvas.setBackground(Color.white);
                canvas.setSize(450,450);
                canvas.addMouseListener(this);
                canvas.addMouseMotionListener(this);
                add(canvas, BorderLayout.CENTER);
                Panel infoPanel = new Panel();
                infoPanel.setLayout(new GridLayout(0, 2, 10, 0));
                location = new Label("Location: ");
```

- MouseEvent

```
                infoPanel.add(location);
                event = new Label("Event: ");
                infoPanel.add(event);
                add(infoPanel, BorderLayout.SOUTH);
                pack();
                setVisible(true);
        }
        public static void main(String[] args) {
                new MouseTest();
        }
        public void mouseClicked(MouseEvent e){
                String text = "Event: Clicked Button ";
                switch(e.getModifiers()){
                        case InputEvent.BUTTON1_MASK:
                                text += 1;
```

```
case InputEvent.BUTTON2_MASK:
```

③ MouseEvent

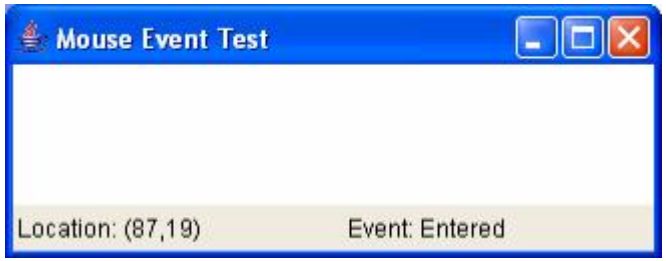
```
                                        text += 2;
                                        break;
                                case InputEvent.BUTTON3_MASK:
                                        text += 3;
                                        break;
                        }
                        text += " (" + e.getClickCount() + "x)";
                        event.setText(text);
        }
        public void mouseEntered(MouseEvent e){
                        event.setText("Event: Entered");
        }
        public void mouseExited(MouseEvent e){
                        event.setText("Event: Exited");
        }
```

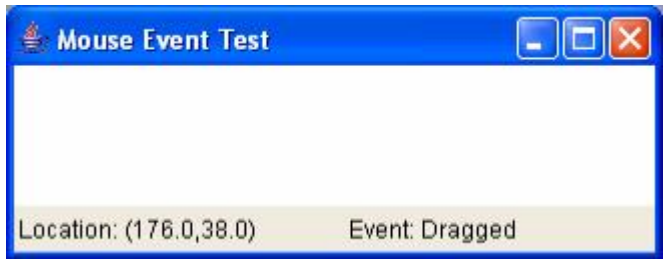
Xử lý cácsựkiện

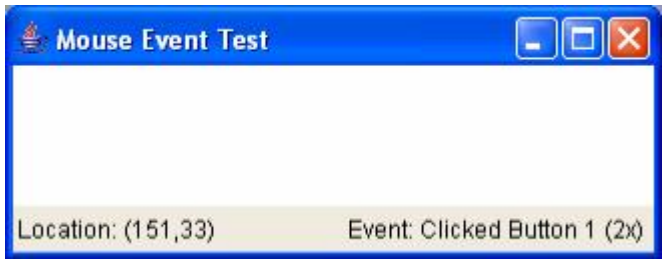
- MouseEvent

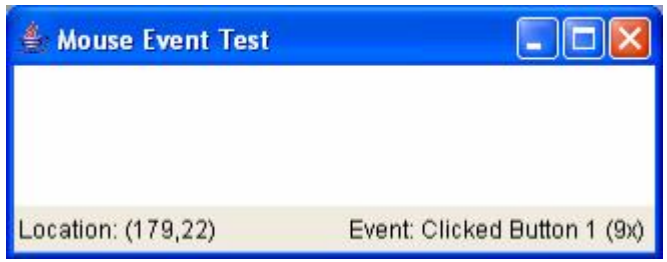
```
        public void mousePressed(MouseEvent e){
                event.setText("Event: Pressed");
        }
        public void mouseReleased(MouseEvent e){
                event.setText("Event: Released");
        }
        public void mouseMoved(MouseEvent e){
                location.setText("Location: (" + e.getX()+"," +
e.getY()+")");
        }
        public void mouseDragged(MouseEvent e){
                Point p = e.getPoint();
                event.setText("Event: Dragged");
                location.setText("Location: (" + p.getX()+"," +
p.getY()+")");
        }
}
```

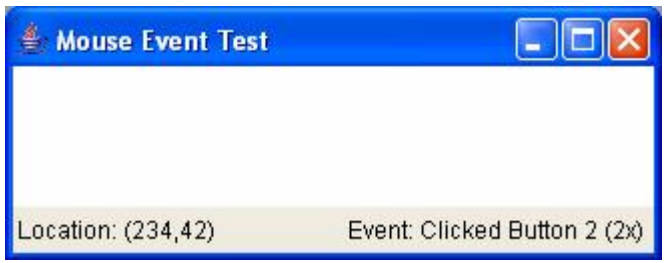
- MouseEvent

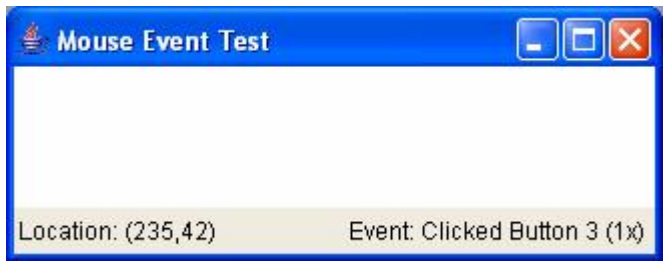

Mouse Event Test
Location: (87,19)
Event: Entered


Mouse Event Test
Location: (176.0,38.0)
Event: Dragged


Mouse Event Test
Location: (151,33)
Event: Clicked Button 1 (2x)


Mouse Event Test
Location: (179,22)
Event: Clicked Button 1 (9x)


Mouse Event Test
Location: (234,42)
Event: Clicked Button 2 (2x)


Mouse Event Test
Location: (235,42)
Event: Clicked Button 3 (1x)

Xử lý cácsựkiện

- KeyEvent
  - Được tạo ra khi người dùng gõ bàn phím
  - Lớp nghe cài đặt giao tiếp KeyListener
  - Các phương thức của KeyListener
    - void keyPressed(KeyEvent)
    - void keyReleased(KeyEvent)
    - void keyTyped(KeyEvent)
  - Một số phương thức của KeyEvent
    - char getKeyChar()
    - int getKeyCode()
    - String getKeyText(int keyCode)

- Nạp ảnh

  - Image image = Toolkit.getDefaultToolkit().
    getImage(fileName);

```
import java.awt.*;
public class ImageTest extends Canvas {
        public ImageTest(){
                super();
                setSize(300,200);
                setBackground(Color.white);
        }
        public void paint(Graphics g){
                Image image =
                        Toolkit.getDefaultToolkit().getImage("image.jpg");
```

```
        g.drawImage(image, 50,50, this);
    }
```

③ Nạp ảnh

- Nạp ảnh

```
        public static void main(String[] args) {
                ImageTest image = new ImageTest();
                GUIFrame frame = new GUIFrame("ImageTest");
                frame.add(image);
                frame.pack();
                frame.setVisible(true);
        }
}
```

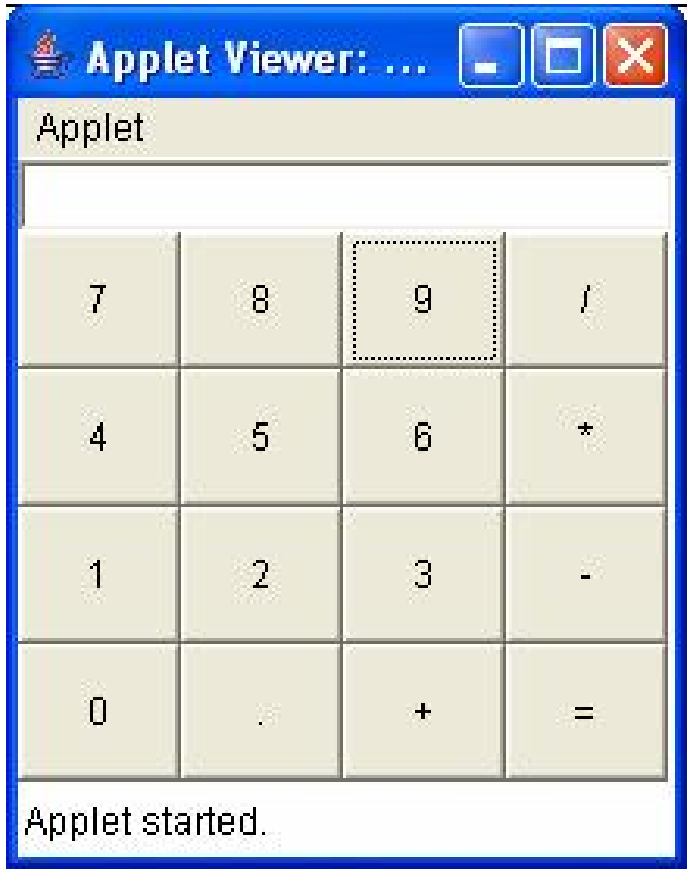
- Nạp ảnh

CÂU HỎI

Bài tập

- Tạo chương trình (Frame) có chứa một Canvas, kích thước 200, 200 màu đỏ. Khi đưa chuột vào Canvas chương trình hiện dòng chữ “In canvas”, khi đưa chuột ra ngoài sẽ hiện dòng chữ “Not in canvas”
- Viết chương trình cho phép vẽ đường tròn tại một vị trí bất kỳ mà người dùng nhấn chuột trên màn hình
- Viết một chương trình với menu “Draw” có 3 menu item: line, circle, square: khi người dùng kích vào từng menu item, 50 line hoặc circle hoặc square sẽ được vẽ ra

- Viết một chương trình tính toán đơn giản



Bài tập

- Viết chương trình cho phép người dùng điều khiển một quả bóng. Trên màn hình có các nút là: To, Nhỏ, Trái, Phải, Lên, Xuống. Khi người dùng ấn 1 nút thì kích cỡ/vị trí của quả bóng sẽ thay đổi theo. Yêu cầu tạo một lớp Ball riêng biệt. (Mở rộng bài toán cho trường hợp người dùng nhấn chuột trực tiếp trên màn hình)
- Viết chương trình mô tả trò chơi dò mìn. Trên màn hình có 3x3 nút bấm và mỗi nút có thể là có mìn hoặc không (ngẫu nhiên). Khi người dùng nhấn một nút, nếu nút đó không có mìn thì cho phép người dùng ấn tiếp, còn không thì thông báo “mìn nổ” và dừng lại. Lưu ý là mỗi nút có một số và người dùng có thể nhấn phím số tương ứng thay vì nhấn chuột vào nút.

- Sự kiện và đối tượng gây ra sự kiện

Event source	Sự kiện	Chú thích
Button	ActionEvent	Nhấn nút
Checkbox	ItemEvent	Chọn, bỏ chọn một item
Choice	ItemEvent	Chọn, bỏ chọn một item
Component	ComponentEvent	Ẩn, hiện, di chuyển
	FocusEvent	Được chọn
	MouseEvent	Tương tác chuột
	KeyEvent	Tương tác bàn phím
Container	ContainerEvent	Thêm, bớt component
List	ActionEvent	Nhấp kép chuột một item
	ItemEvent	Chọn, bỏ chọn một item

- Sự kiện và đối tượng gây ra sự kiện

Event source	Sự kiện	Chú thích
MenuItem	ActionEvent	Chọn một menu item
Scrollbar	AdjustmentEvent	Di chuyển thanh cuộn
TextComponent	TextEvent	Thay đổi văn bản
TextField	ActionEvent	Kết thúc thay đổi văn bản
Window	WindowEvent	Thay đổi cửa sổ

Tham khảo

- Đối tượng nghe và phương thức cần cài đặt

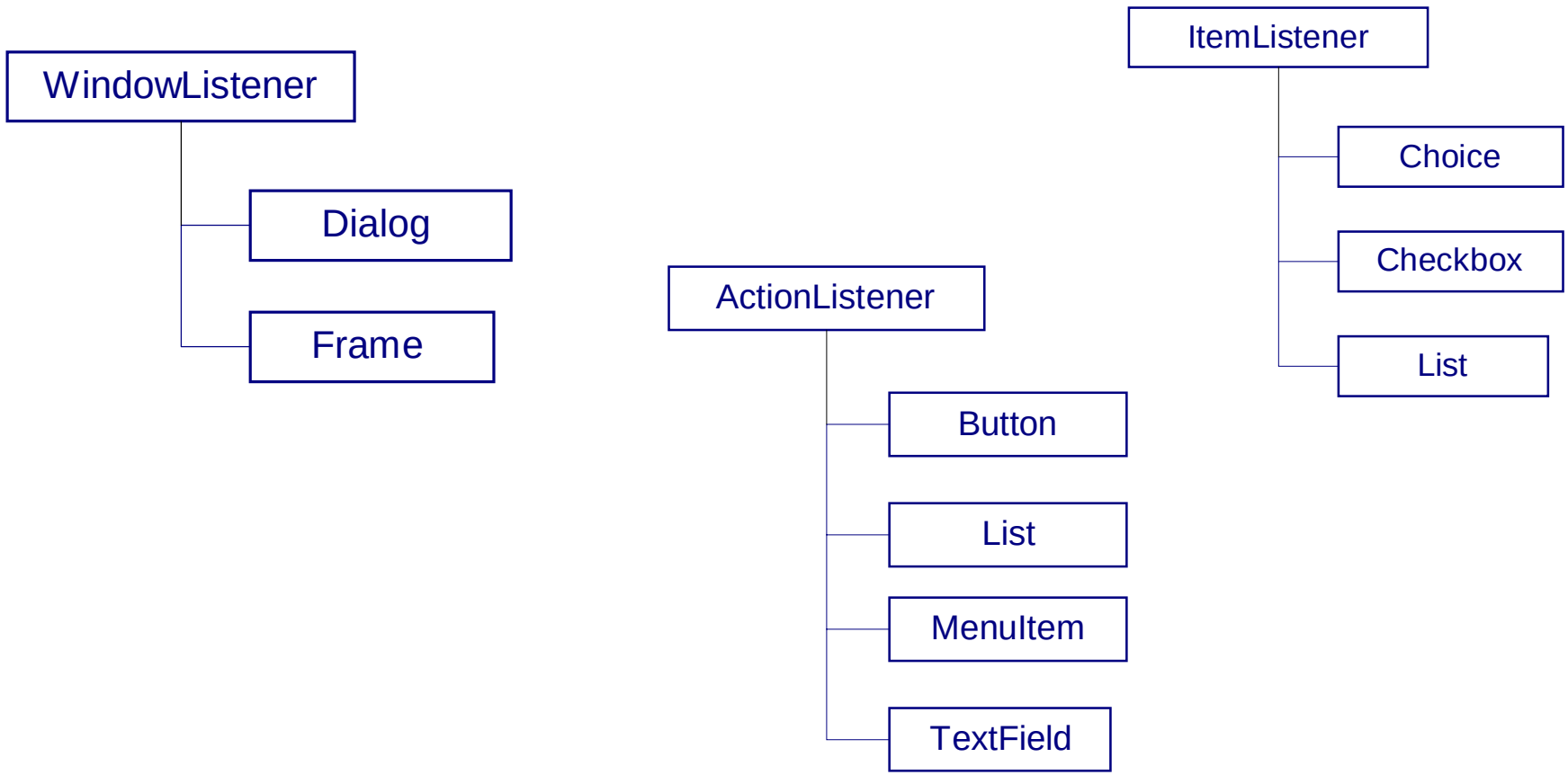
Event Class	Listener Interface	Listener Methods
ActionEvent	ActionEvent	actionPerformed()
AdjustmentEvent	AdjustmentListener	adjustmentValueChanged()
ComponentEvent	ComponentListener	componentHidden()
		componentMoved()
		componentResized()
		componentShown()
ContainerEvent	ContainerListener	componentAdded()
		componentRemoved()
FocusEvent	FocusListener	focusGained()
		focusLost()
ItemEvent	ItemListener	itemStateChanged()

Tham khảo

- Đối tượng nghe và phương thức cần cài đặt

Event Class	Listener Interface	Listener Methods
KeyEvent	KeyListener	keyPressed()
		keyReleased()
		keyTyped()
MouseEvent	MouseListener	mouseClicked()
		mousePressed()
		mouseReleased()
	MouseMotionListener	mouseDragged()
		mouseMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowClosed()
		windowActivated()

- Listener và các thành phần tương ứng



③ Listener cho Component