

The background of the slide is a dense, overlapping pattern of various US coins, including pennies, nickels, and quarters, rendered in a light, faded grey tone. A solid orange horizontal band spans the middle of the image, serving as a backdrop for the text.

NHẬP MÔN JAVA
BÀI 6

LẬP TRÌNH SỰ KIỆN

NỘI DUNG TRÌNH BÀY

- Các ví dụ mở đầu
- Mô hình xử lý sự kiện
- Các component nâng cao
- Xử lý sự kiện chuột
- Xử lý sự kiện bàn phím

The background of the slide is a dense, overlapping pattern of various US coins, including pennies, nickels, and quarters, rendered in a light, faded grey tone. A solid orange horizontal band is positioned across the middle of the image, serving as a backdrop for the title text.

PHẦN 1 **CÁC VÍ DỤ** **MỞ ĐẦU**

VÍ DỤ 1

Xây dựng một chương trình như sau:

- Khi nhấn vào button Red hoặc button Green hoặc button Blue thì nền của cửa sổ chương trình thay đổi màu tương ứng, đồng thời label bên dưới các button cũng có câu thông báo màu tương ứng.



VÍ DỤ 1 (file MyFirstAwt.java)

```
import java.awt.*;
import java.awt.event.*;
public class MyFirstAwt extends Frame
{
    Label status;
    Button button1 = new Button("Red");
    Button button2 = new Button("Green");
    Button button3 = new Button("Blue");

    MyFirstAwt()
    {
        this.setTitle("My First Awt"); //super("My First Awt");
        this.setLayout(new FlowLayout());
        this.add(button1);
        this.add(button2);
        this.add(button3);
        status = new Label();
        status.setText("Press any button, please!");
        this.add(status);
    }
}

//xem tiếp ở slide tiếp theo
```

VÍ DỤ 1 (file MyFirstAwt.java) - tt

```
button1.addActionListener(new MyListener(status,this));  
button2.addActionListener(new MyListener(status,this));  
button3.addActionListener(new MyListener(status,this));
```

```
this.addWindowListener(new WindowAdapter()  
{  
    public void windowClosing(WindowEvent evt){System.exit(0);}  
});
```

```
public static void main(String[] args)  
{  
    MyFirstAwt mfa = new MyFirstAwt();  
    mfa.resize(300,200);  
    mfa.show();
```

```
}  
}
```

VÍ DỤ 1 (file MyListener.java)

```
import java.awt.*;  
import java.awt.event.*;  
  
public class MyListener implements ActionListener  
{  
  
    Label status;  
    Component compo;  
  
    MyListener(Label status1, Component compo1)  
    {  
        this.status = status1;  
        this.compo = compo1;  
    }  
    //xem tiếp ở slide tiếp theo
```

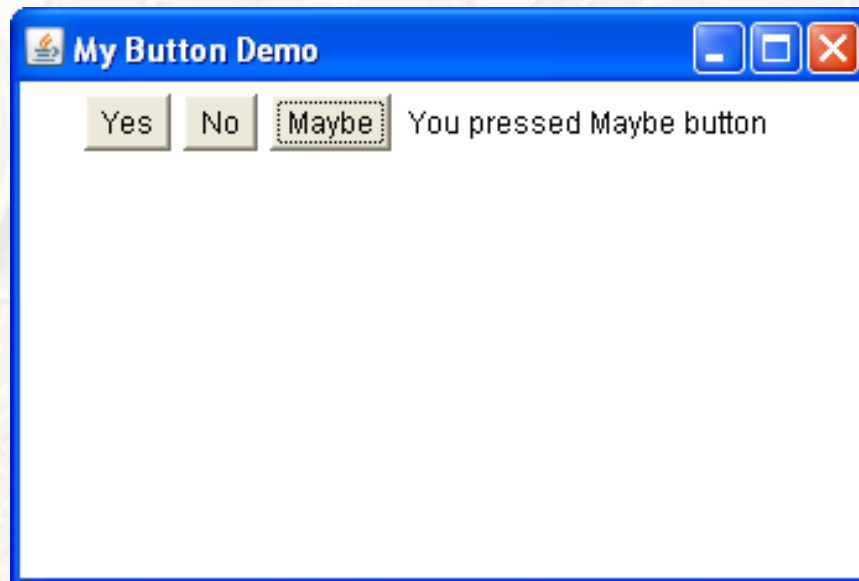

VÍ DỤ 1 (file MyListener.java) - tt

```
public void actionPerformed(ActionEvent evt)
{
    if(evt.getSource() instanceof Button)
    {
        Button temp = (Button)evt.getSource();
        status.setText("You have selected: " + temp.getLabel());
        if(temp.getLabel().equalsIgnoreCase("Red"))
        {
            compo.setBackground(new Color(255,0,0));
        }
        if(temp.getLabel().equalsIgnoreCase("Green"))
        {
            compo.setBackground(new Color(0,255,0));
        }
        if(temp.getLabel().equalsIgnoreCase("Blue"))
        {
            compo.setBackground(new Color(0,0,255));
        }
    }
}
```


VÍ DỤ 2

Xây dựng một chương trình như sau:

- Khi nhấn vào button Yes hoặc button No hoặc button Maybe thì xuất hiện câu thông báo tương ứng.



VÍ DỤ 2 (file ButtonDemo.java)

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

public class ButtonDemo extends Frame implements ActionListener
{
    String message = "";
    Button yes,no,maybe;
    Label label = new Label();
    ButtonDemo(String msg)
    {
        setTitle(msg); //super("My First Awt");
        setLayout(new FlowLayout());
        yes = new Button("Yes");
        no = new Button("No");
        maybe = new Button("Maybe");
        add(yes);
        add(no);
        add(maybe);
        add(label);
        yes.addActionListener(this);
        no.addActionListener(this);
        maybe.addActionListener(this);
    } //xem tiếp ở slide tiếp theo
}
```

VÍ DỤ 2 (file ButtonDemo.java)-tt

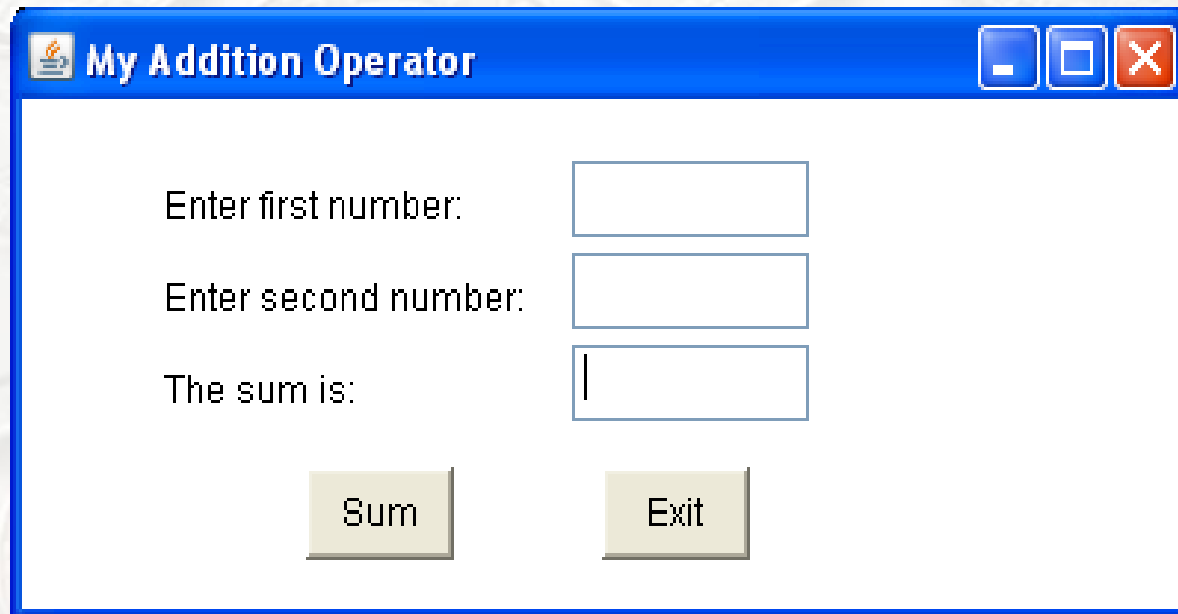
```
public void actionPerformed(ActionEvent evt)
{
    String str = evt.getActionCommand();
    if(str.equals("Yes"))
    {
        label.setText("You pressed Yes button");
    }
    if(str.equals("No"))
    {
        label.setText("You pressed No button");
    }
    if(str.equals("Maybe"))
    {
        label.setText("You pressed Maybe button");
    }
}

public static void main(String[] args)
{
    ButtonDemo btdm = new ButtonDemo("My Button Demo");
    btdm.setSize(300,200);
    btdm.show();
}
}
```

VÍ DỤ 3

Xây dựng một chương trình như sau:

- Nhập vào hai số rồi nhấp button Sum để tính tổng



The screenshot shows a Windows application window titled "My Addition Operator". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is white and contains three text labels with corresponding input fields: "Enter first number:" followed by a text box, "Enter second number:" followed by a text box, and "The sum is:" followed by a text box. At the bottom of the window, there are two buttons: "Sum" and "Exit".

My Addition Operator

Enter first number:

Enter second number:

The sum is:

Sum Exit

VÍ DỤ 3 (AddOperator.java)

```
import java.awt.*;
import java.awt.event.*;
public class AddOperator extends Frame implements ActionListener
{
    Label firstLabel = new Label("Enter first number:");
    Label secondLabel = new Label("Enter second number:");
    Label resultLabel = new Label("The sum is:");
    TextField firstTextField = new TextField(5);
    TextField secondTextField = new TextField(5);
    TextField resultTextField = new TextField(5);
    Button sumButton = new Button("Sum");
    Button exitButton = new Button("Exit");

    AddOperator()
    {
        this.setTitle("My Addition Operator");
        this.setLayout(null);
        sumButton.setBounds(100,150,50,30);
        this.add(sumButton);
        sumButton.addActionListener(this);
    }
    //xem tiếp ở slide tiếp theo
}
```

VÍ DỤ 3 (AddOperator.java) - tt

```
exitButton.setBounds(200,150,50,30);
this.add(exitButton);
exitButton.addActionListener(this);
firstLabel.setBounds(50,50,130,30);
this.add(firstLabel);
secondLabel.setBounds(50,80,130,30);
this.add(secondLabel);
resultLabel.setBounds(50,110,130,30);
this.add(resultLabel);
firstTextField.setBounds(190,50,80,25);
this.add(firstTextField);
secondTextField.setBounds(190,80,80,25);
this.add(secondTextField);
resultTextField.setBounds(190,110,80,25);
this.add(resultTextField);
this.addWindowListener(new WindowAdapter()
{
    public void windowClosing(WindowEvent evt){System.exit(0);}
});
```

} //xem tiếp ở slide tiếp theo

VÍ DỤ 3 (AddOperator.java) - tt

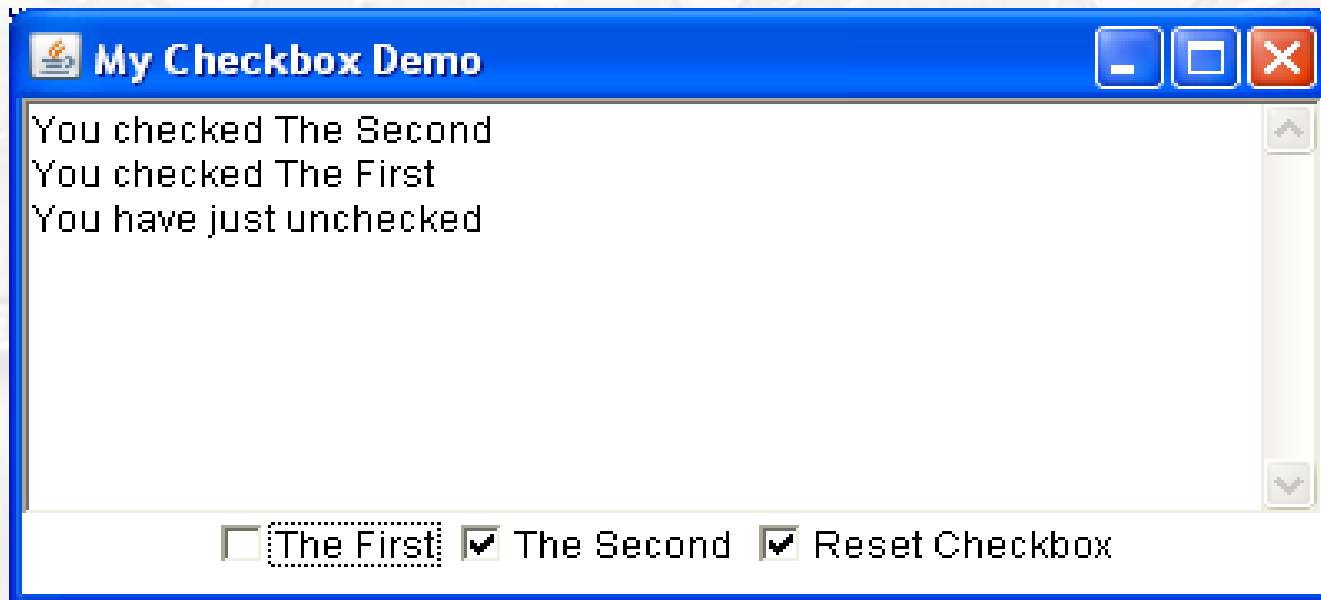
```
public void actionPerformed(ActionEvent evt)
{
    if(evt.getSource()==sumButton)
    {
        int firstNum = Integer.parseInt(firstTextField.getText());
        int secondNum = Integer.parseInt(secondTextField.getText());
        int resultNum = firstNum + secondNum;
        resultTextField.setText(String.valueOf(resultNum));
    }
    if(evt.getSource()==exitButton)
    {
        System.exit(0);
    }
}

public static void main(String[] args)
{
    AddOperator ao = new AddOperator();
    ao.setBounds(10,10,400,200);
    ao.setVisible(true);
}
}
```


VÍ DỤ 4

Xây dựng một chương trình như sau:

- Khi nhấp để chọn hoặc nhấp để bỏ chọn các checkbox thì xuất hiện câu thông báo tương ứng trong vùng TextArea.



VÍ DỤ 4 (file CheckBoxDemo.java)

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

public class CheckBoxDemo extends Frame implements ItemListener
{
    TextArea txtArea = new TextArea(8,50); //5 rows and 40 columns
    //CheckboxGroup g = new CheckboxGroup();
    Checkbox checkBox1 = new Checkbox("The First");
    Checkbox checkBox2 = new Checkbox("The Second");
    Checkbox checkBox3 = new Checkbox("Reset Checkbox");
    CheckBoxDemo()
    {
        this.setTitle("My Checkbox Demo");
        this.setLayout(new BorderLayout());
        this.add(txtArea,BorderLayout.NORTH);
        Panel panel = new Panel();
        panel.setLayout(new FlowLayout());
        panel.add(checkBox1); panel.add(checkBox2); panel.add(checkBox3);
        this.add(panel,BorderLayout.SOUTH);
        checkBox1.addItemListener(this);
        checkBox2.addItemListener(this);
        checkBox3.addItemListener(this);
    }
}
```

//xem tiếp ở slide tiếp theo

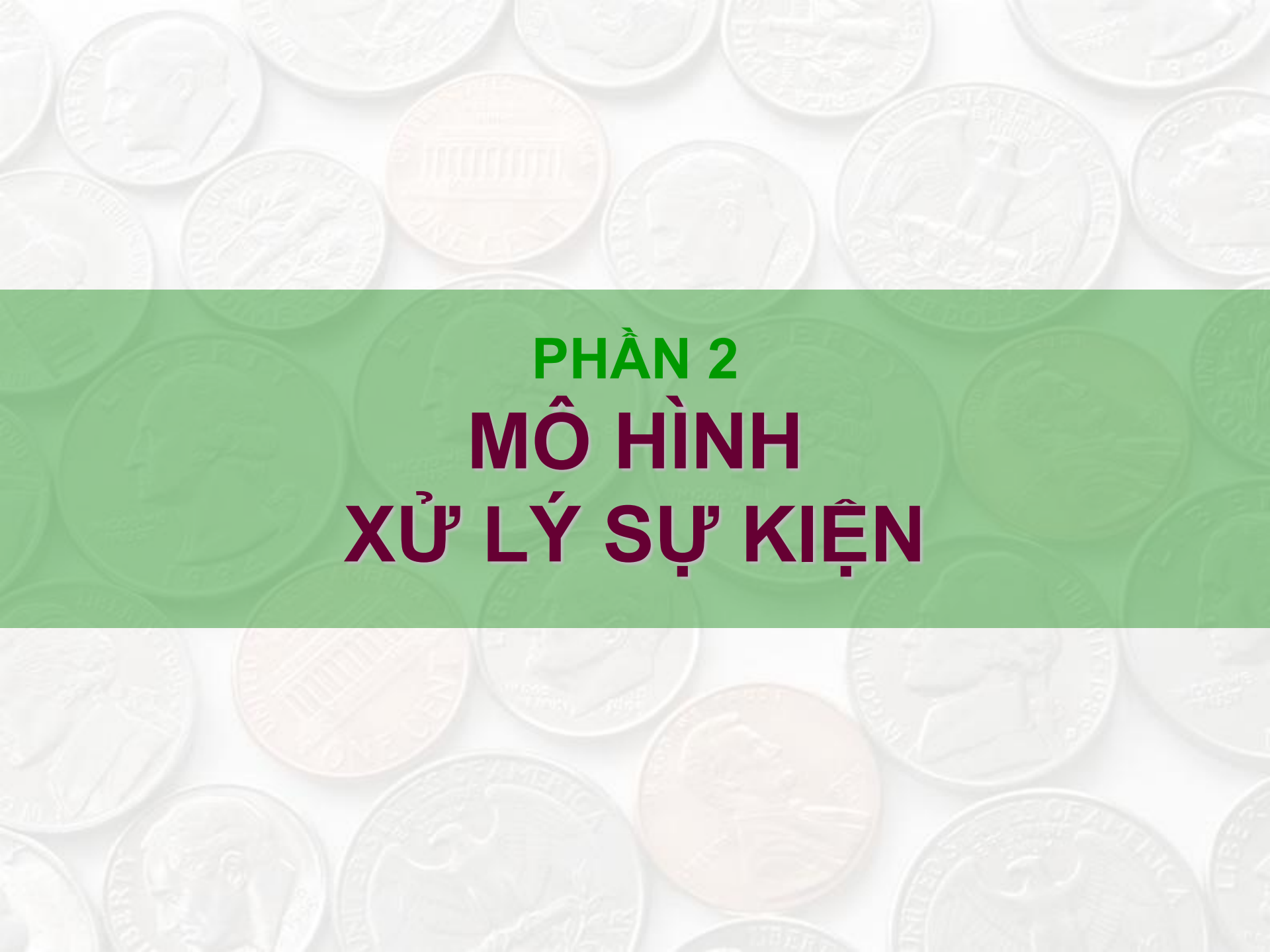
VÍ DỤ 4 (file CheckBoxDemo.java)-tt

```
public void itemStateChanged(ItemEvent evt)
{
    if(evt.getStateChange()==ItemEvent.SELECTED)
    {
        String itemLabel = (String)evt.getItem();
        if(itemLabel=="The First")
        {
            txtArea.appendText("You checked " + itemLabel + "\n");
            System.out.println(itemLabel);
        }
        if(itemLabel=="The Second")
        {
            txtArea.appendText("You checked " + itemLabel + "\n");
            System.out.println(itemLabel);
        }
        if(itemLabel=="Reset Checkbox")
        {
            txtArea.setText(""); System.out.println(itemLabel); }
    }
    if(evt.getStateChange()==ItemEvent.DESELECTED)
    {
        txtArea.appendText("You have just unchecked\n");
        System.out.println("You have just unchecked\n");
    }
}
```

//xem tiếp ở slide tiếp theo

VÍ DỤ 4 (file CheckBoxDemo.java)-tt

```
public static void main(String[] arg)
{
    CheckBoxDemo chkdemo = new CheckBoxDemo();
    chkdemo.setSize(400,200);
    chkdemo.setVisible(true);
}
} //hết
```


The background of the slide is a close-up, slightly blurred image of various US coins, including pennies, nickels, and quarters, scattered across the surface. A semi-transparent green rectangular banner is centered horizontally across the middle of the image, serving as a backdrop for the title text.

PHẦN 2

MÔ HÌNH

XỬ LÝ SỰ KIỆN

MÔ HÌNH XỬ LÝ SỰ KIỆN

Ba thành phần chính của mô hình

- Event source: nguồn gây ra sự kiện, thường là các thành phần GUI trong chương trình
- Event object: đối tượng lưu thông tin về sự kiện đã xảy ra
- Event listener: đối tượng sẽ nhận được thông tin khi có sự kiện xảy ra

MÔ HÌNH XỬ LÝ SỰ KIỆN

- Sự kiện (event) được phát sinh khi người dùng tương tác với GUI, ví dụ: di chuyển chuột, ấn nút, nhập dữ liệu văn bản, chọn menu...
- Thông tin về sự kiện được lưu trong một đối tượng sự kiện thuộc lớp con của lớp `AWTEvent` (gói `java.awt.event`).
- Chương trình có thể xử lý các sự kiện bằng cách đặt “lắng nghe sự kiện” trên các thành phần GUI.

MÔ HÌNH XỬ LÝ SỰ KIỆN

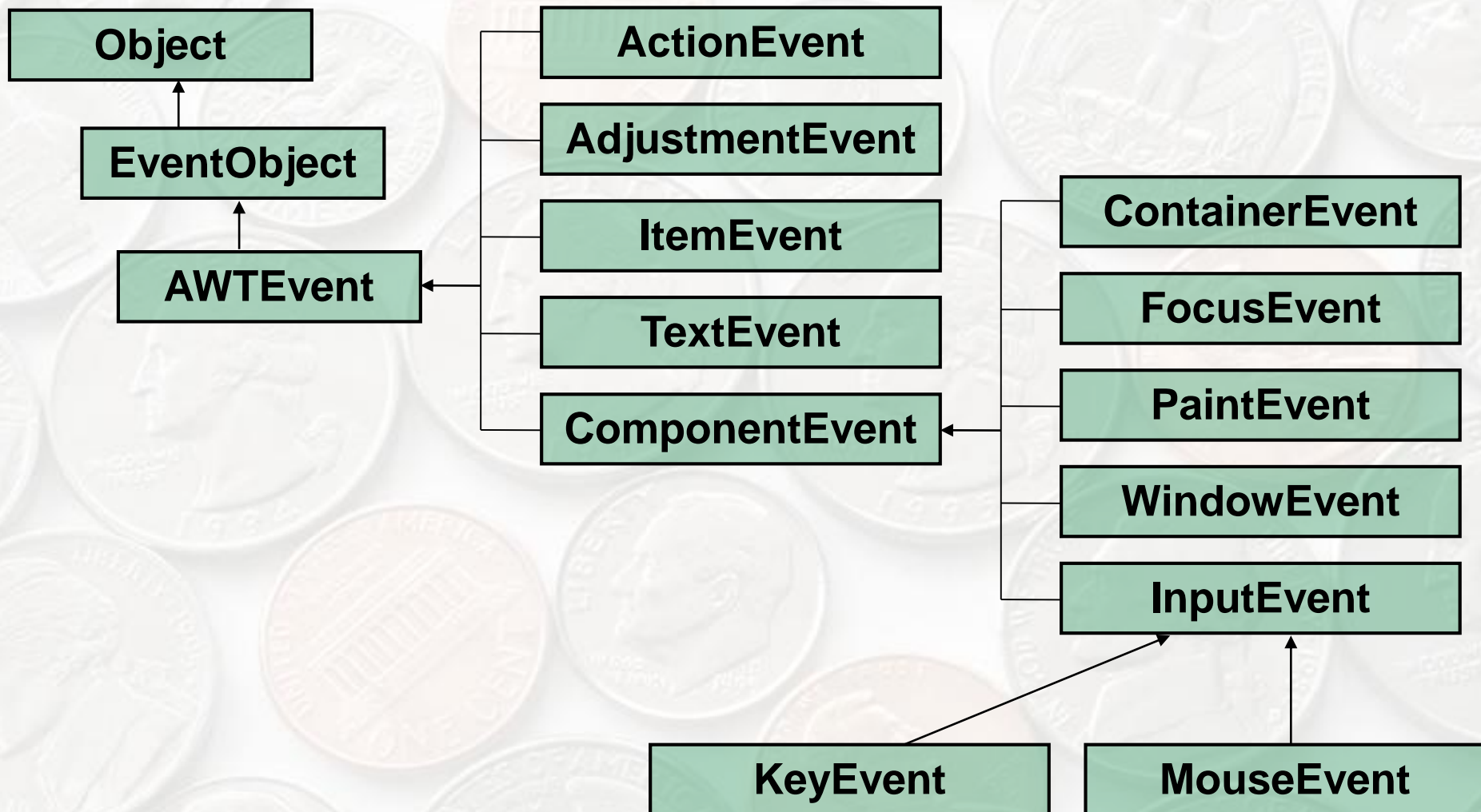


- Việc thông báo sự kiện xảy ra thực chất là việc gọi một phương thức của EventListener với đối số truyền vào là EventObject.
- Các lớp con của EventListener có thể cài đặt các phương thức để xử lý sự kiện

MÔ HÌNH XỬ LÝ SỰ KIỆN

- Nguồn sự kiện
 - Các lớp thành phần GUI mà người sử dụng tương tác.
 - Bạn có thể đăng ký “Listener” đáp ứng với những sự kiện nhất định
- Bộ lắng nghe (Listener)
 - Nhận đối tượng sự kiện khi được thông báo và thực hiện đáp ứng thích hợp.
 - Nhiều kiểu của bộ lắng nghe tồn tại cho các sự kiện cụ thể như `MouseListener`, `ActionListener`, `KeyListener`,...
 - Các giao tiếp được hiện thực và cài đặt các hành động
- Đối tượng sự kiện (Event)
 - Đóng gói thông tin về sự kiện xuất hiện
 - Các đối tượng sự kiện được gửi tới bộ lắng nghe khi sự kiện xuất hiện trên thành phần GUI

GÓI java.awt.event.*



MỘT SỐ LỚP SỰ KIỆN

Sự kiện cấp thấp: dùng cho hầu hết các thành phần

- FocusEvent: đặt/chuyển focus
- InputEvent: sự kiện phím (KeyEvent) hoặc chuột (MouseEvent)
- ContainerEvent: thêm hoặc xoá các component
- WindowEvent: đóng, mở, di chuyển cửa sổ
- ...

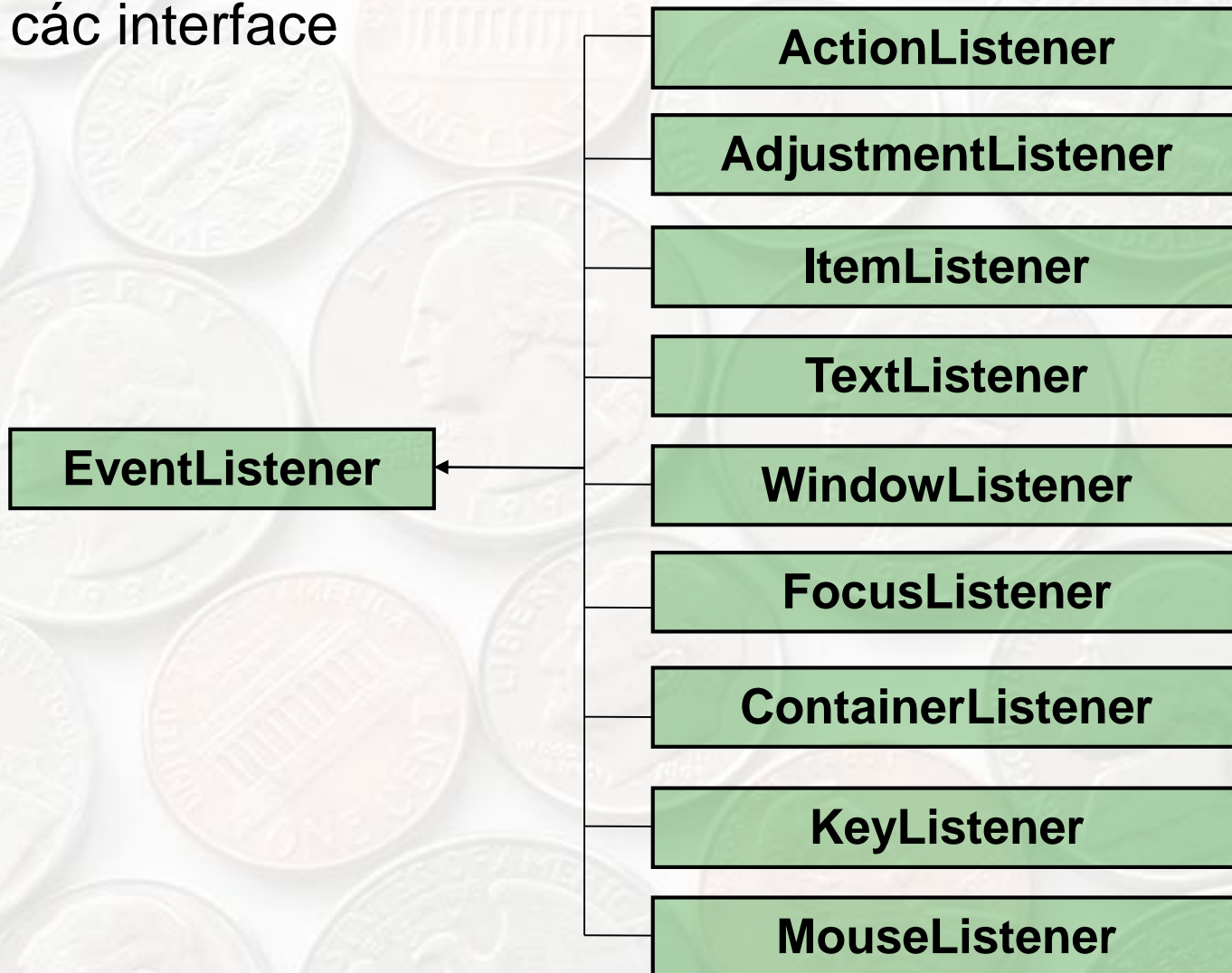
MỘT SỐ LỚP SỰ KIỆN

Sự kiện cấp cao: dùng cho một số thành phần đặc thù

- **ActionEvent**: sự kiện sinh ra từ các thành phần giao tiếp với người dùng như nhấn một nút, chọn menu...
- **ItemEvent**: lựa chọn một item trong danh sách
- **TextEvent**: thay đổi giá trị của hộp text
- ...

MỘT SỐ BỘ LẮNG NGHE SỰ KIỆN

Là các interface



CÀI ĐẶT VÀ QUẢN LÝ SỰ KIỆN

- Xác định đối tượng sẽ gây ra sự kiện (event source). Ví dụ: nút bấm.
- Xác định sự kiện cần xử lý trên đối tượng gây sự kiện. Ví dụ: ấn nút.
- Xác định đối tượng nghe sự kiện (event listener) và cài đặt các phương thức tương ứng. Ví dụ: chính applet sẽ nghe sự kiện.
- Đăng ký đối tượng nghe trên đối tượng gây ra sự kiện. Ví dụ: `button.addActionListener(...)`;

CÁC EVENT SOURCE & EVENT OBJECT

Event source	Event	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

CÁC EVENT SOURCE & EVENT OBJECT

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

Các Listener Method

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

Các Listener Method

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()

ĐĂNG KÝ ĐỐI TƯỢNG LẮNG NGHE

- Để đăng ký đối tượng nghe ta sử dụng tên phương thức có cấu trúc như sau:

add + loại sự kiện + Listener(lớp nghe sự kiện)

- Ví dụ với nút Button

addActionListener(ActionListener)

- Ví dụ với danh sách List

addActionListener(ActionListener)

addItemListener(ItemListener)

The background of the slide is a close-up, slightly blurred image of various US coins, including pennies, nickels, and quarters, scattered across the surface. A semi-transparent blue horizontal band is overlaid across the middle of the image, serving as a backdrop for the title text.

PHẦN 3

CÁC COMPONENT

NÂNG CAO

VÙNG VĂN BẢN (TextArea)

- Cho phép người dùng nhập vào nhiều dòng văn bản.

Constructors

TextArea()

TextArea(int cols, int rows)

TextArea(String S)

TextArea(String S, int rows, int cols)

TextArea(String, int cols, int rows, int Scrollbars)

Common methods

void setText(String)

String getText()

void setEditable(boolean)

boolean **isEditable()**

void **insert**(String S, int Index)

void **replaceRange**(String S, int begin, int end)

VÙNG VĂN BẢN (TextArea)

Ví dụ:

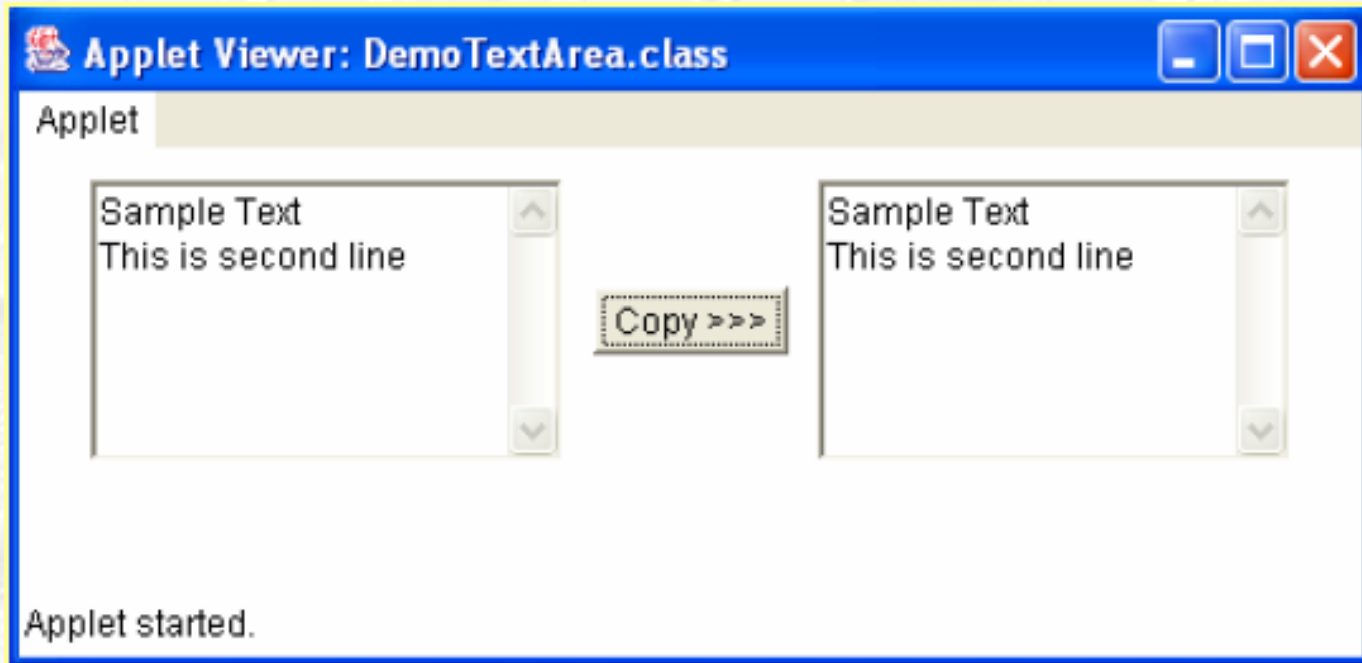
// Cac import can thiet...

```
public class DemoTextArea extends Applet implements ActionListener
{
    private TextArea textArea1, textArea2;
    private Button copy;
    public void init()
    {
        textArea1 = new TextArea("Sample Text", 5, 20);
        textArea2 = new TextArea(5, 20);
        copy = new Button("Copy >>>");
        setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
        copy.addActionListener(this);
        add(textArea1); add(copy); add(textArea2);
    }
}
```

//xem tiếp ở slide tiếp theo

VÙNG VĂN BẢN (TextArea)

```
public void actionPerformed(ActionEvent event)
{
    textArea2.setText(textArea1.getText());
}
}
```



KHUNG VẼ (Canvas)

- Khung vẽ là một vùng chuyên để vẽ đồ hoạ, nó không bị che bởi các thành phần giao diện khác.
- Khung vẽ có thể xử lý các sự kiện giống như Applet.
- Để sử dụng khung vẽ, cần tạo một lớp khác dẫn xuất từ Canvas và cài đặt nạp chồng phương thức paint().
- Nên gọi setSize cho khung vẽ. Toạ độ vẽ là (0,0) tính trong khung vẽ.

KHUNG VẼ (Canvas)

// Cac import can thiet...

public class DemoCanvas extends Applet implements ActionListener

{ private Buttonrect Button;

private Buttoncircle Button;

private MyCanvas canvas;

public void init()

{ setLayout(new BorderLayout());

rectButton = new Button("Draw Rectangle");

circleButton = new Button("Draw Circle");

rectButton.addActionListener(this);

circleButton.addActionListener(this);

Panel panel = new Panel();

panel.add(rectButton);

panel.add(circleButton);

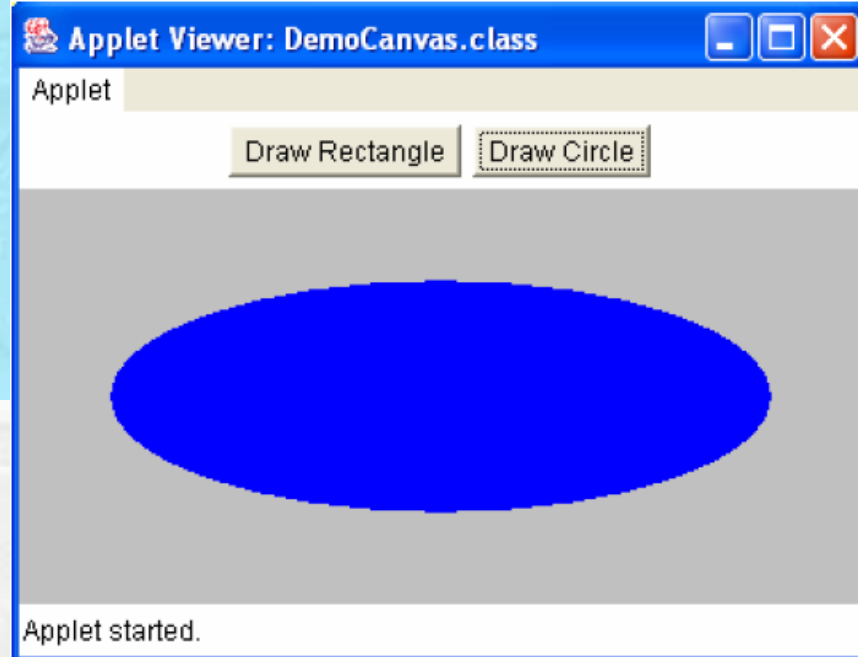
//xem tiếp ở slide tiếp theo

KHUNG VẼ (Canvas)

```
canvas = new MyCanvas();
canvas.setBackground(Color.lightGray);
add(panel, BorderLayout.NORTH);
add(canvas, BorderLayout.CENTER);
}
public void actionPerformed(ActionEvent event)
{
    if(event.getSource() == rectButton)
        canvas.draw(1);
    else if(event.getSource() == circleButton)
        canvas.draw(2);
}
}
```

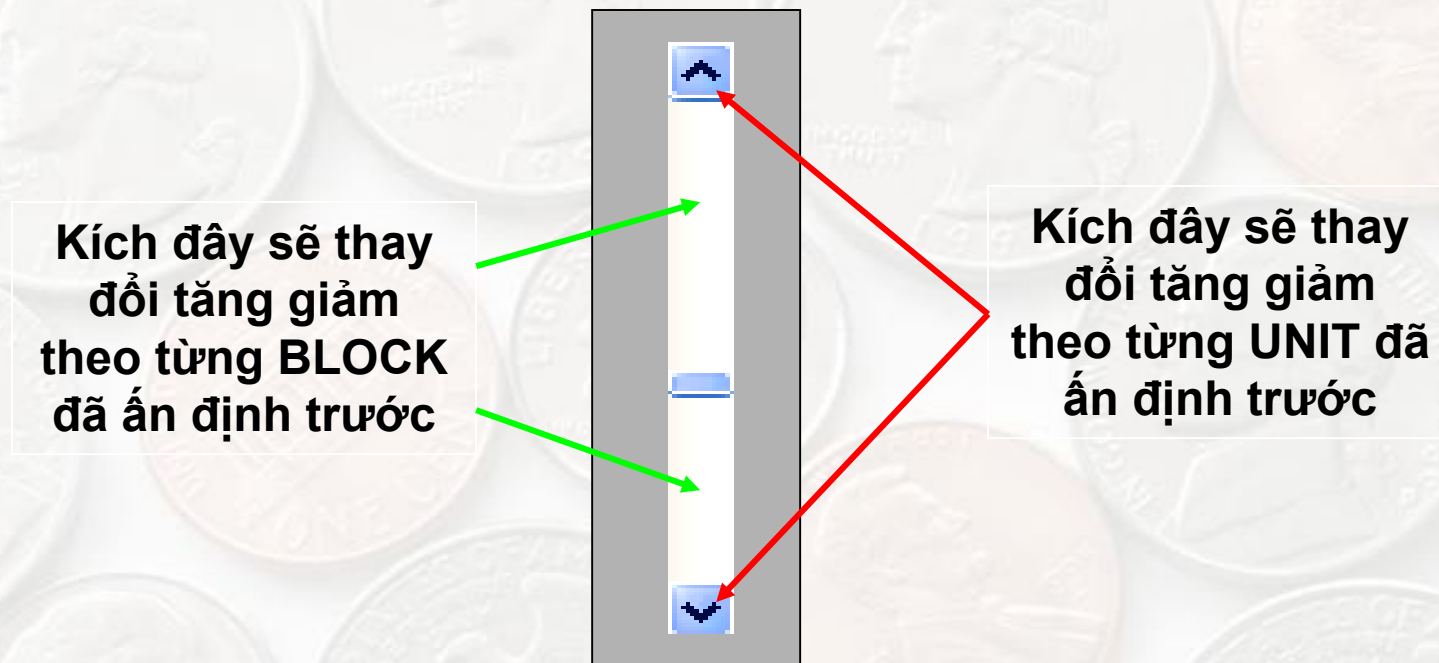
KHUNG VẼ (Canvas)

```
class MyCanvas extends Canvas
{
    private int shape;
    public void paint(Graphics g)
    {
        Dimension size = getSize();
        g.setColor(Color.BLUE);
        if(shape == 1) g.fillRect(40, 40, size.width-80, size.height-80);
        else if(shape == 2) g.fillOval(40, 40, size.width-80, size.height-80);
    }
    public void draw(int shape)
    {
        this.shape = shape;
        repaint();
    }
}
```



THANH TRƯỢT (Scrollbar)

- Công cụ nhập 1 trị trong 1 khoảng số (biểu diễn bằng Maximum, Minimum) bằng cách kéo con trượt.
- Tại 1 thời điểm, con trượt ở tại vị trí mô tả cho trị hiện hành (Value)
- Có thể có hướng ngang hoặc dọc (Orientation)



THANH TRƯỢT (Scrollbar)

Constructors

Scrollbar() - tạo thanh cuộn dọc

Scrollbar(int orientation) // VERTICAL|HORIZONTAL

Scrollbar(int orientation, int value, int visible, int minimum, int maximum)

Common methods

void setMaximum(int v)

void setMinimum(int v)

int getOrientation()

void setUnitIncrement(int v)

void setBlockIncrement(int v)

void setValue(int v)

void setVisibleAmount(int newAmount)

int getVisibleAmount()

int getMaximum() ;

int getMinimum()

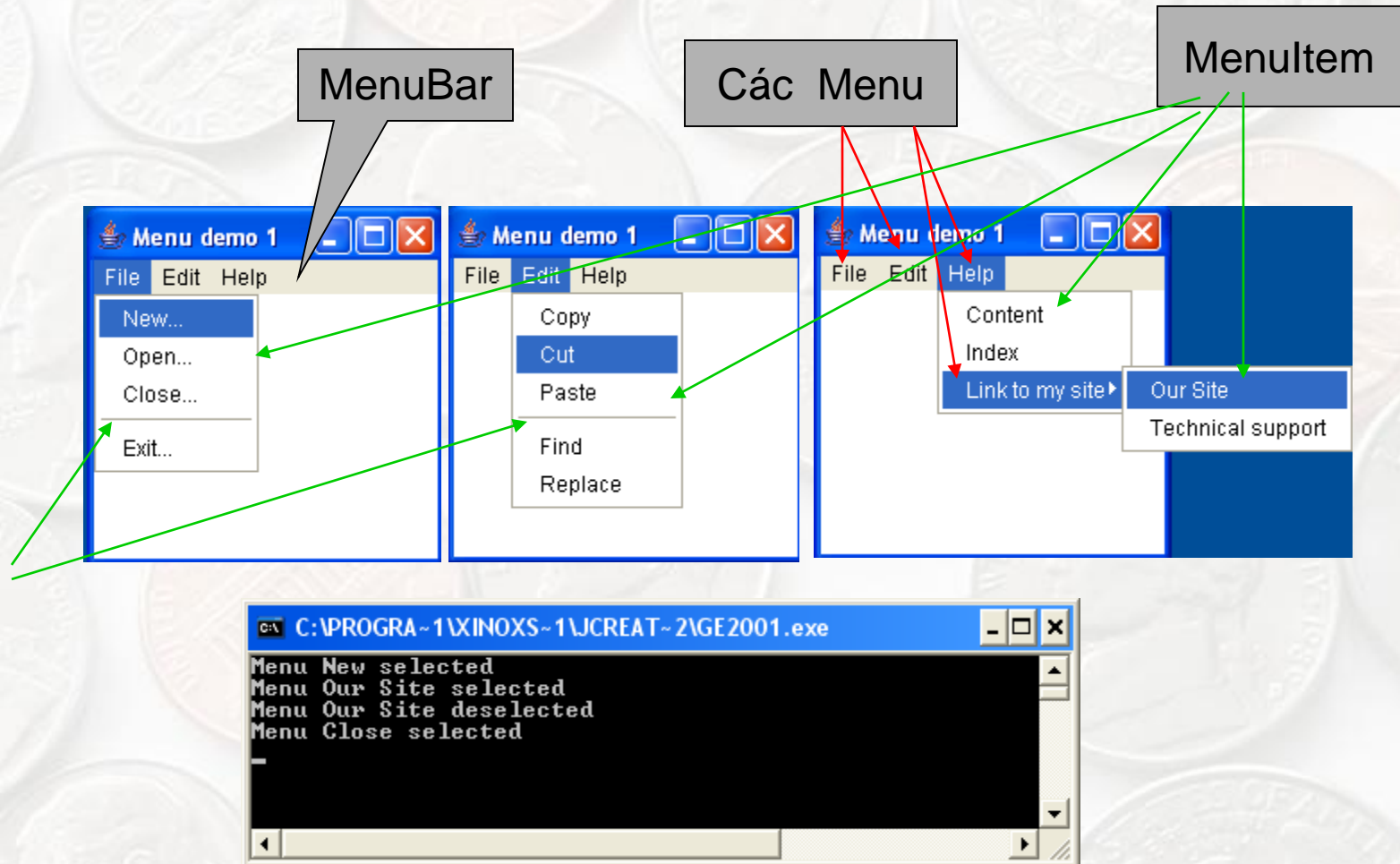
int getUnitIncrement()

int getBlockIncrement()

int getValue()

MENU (thực đơn)

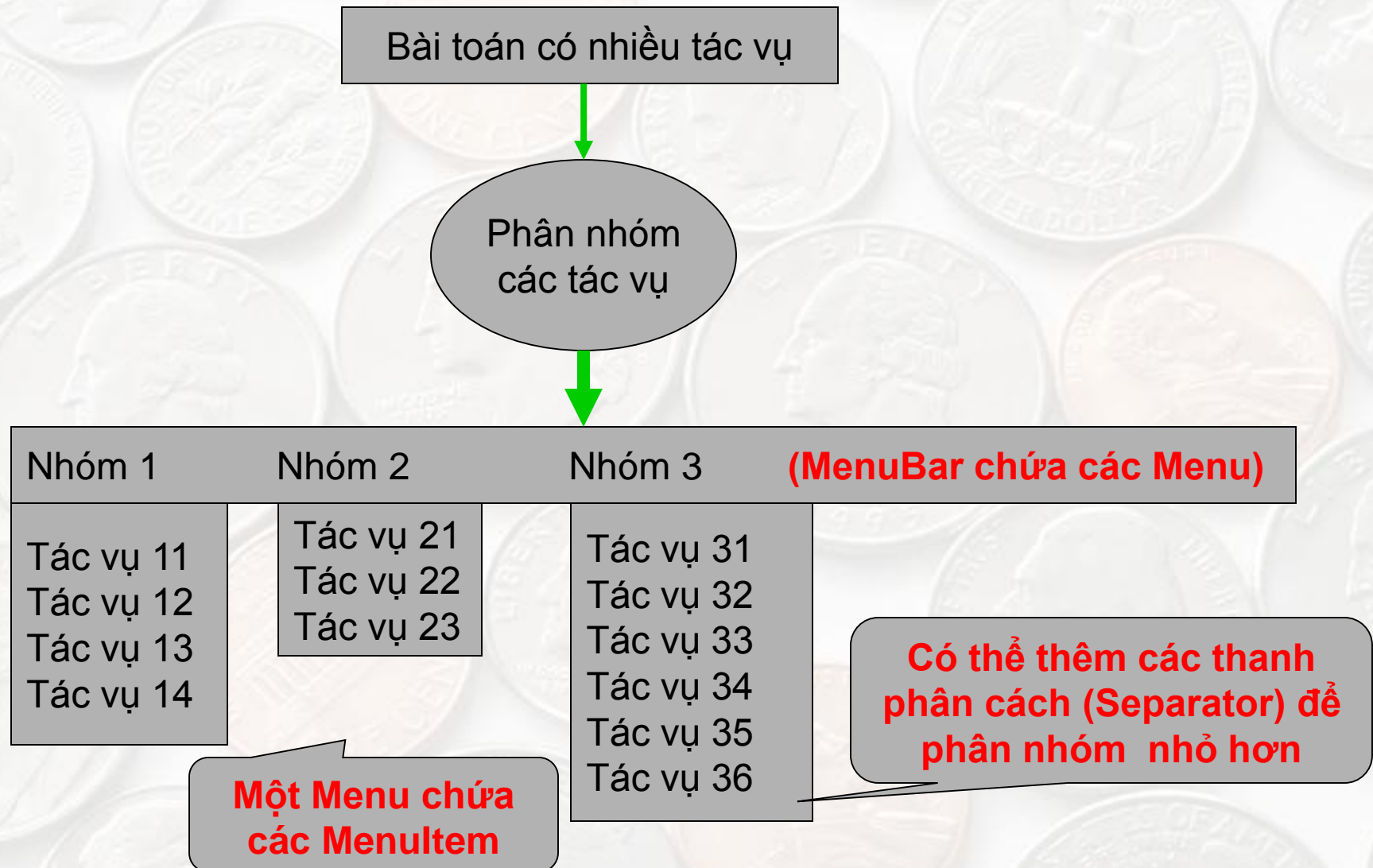
Cấu trúc một hệ menu:



MENU (thực đơn)

- Label-Chuỗi mô tả.
- Shortcut key- Phím nóng được kết hợp.
- Enable/ Disable- Cho user tác động?
- Action Command- Chuỗi tên lệnh được kết hợp.
- Ủy thác xử lý sự kiện : ActionListener

MENU (thực đơn)



VÍ DỤ 1: MENU ĐƠN GIẢN

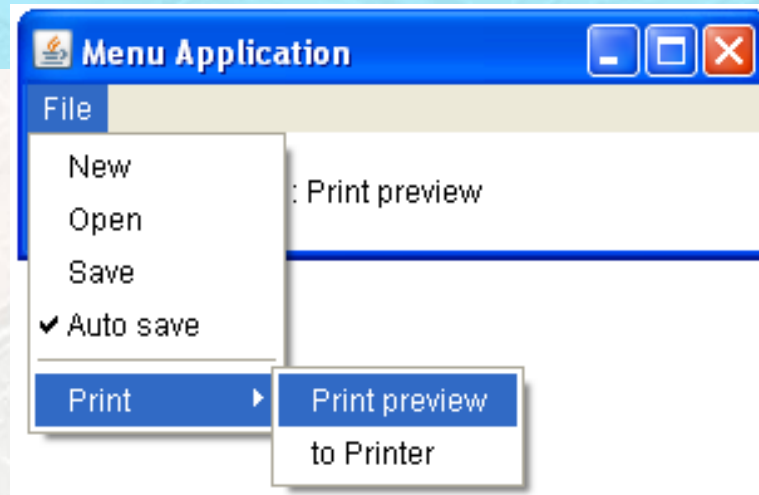
```
import java.awt.*; import java.applet.*; import java.awt.event.*;
public class MenuDemo
{   public static void main(String args[]){
    Frame myWindow = new Frame("Menu Application");
    Label status=new Label("Pleased select an item on menu");
    myWindow.add(status);
    MenuBar AppMenu =new MenuBar();
    myWindow.setMenuBar(AppMenu);
    Menu FileMenu=new Menu("File");
    AppMenu.add(FileMenu);
    MenuItem newItem=new MenuItem("New");
    MenuItem openItem=new MenuItem("Open");
    MenuItem saveItem=new MenuItem("Save");
    CheckboxMenuItem AutosaveItem=new CheckboxMenuItem("Auto save");
    AutosaveItem.setState(true);
    Menu printItem = new Menu("Print");
    MenuItem Item1=new MenuItem("Print preview");
    MenuItem Item2=new MenuItem("to Printer");
    printItem.add(Item1); printItem.add(Item2);
```


VÍ DỤ 1: MENU ĐƠN GIẢN

```
FileMenu.add(newItem); FileMenu.add(openItem); FileMenu.add(saveItem);
FileMenu.add(AutoSaveItem); FileMenu.addSeparator();
FileMenu.add(printItem);
newItem.addActionListener(new processItem(status));
openItem.addActionListener(new processItem(status));
saveItem.addActionListener(new processItem(status));
AutoSaveItem.addActionListener(new processItem(status));
//For print sub menu
Item1.addActionListener(new processItem(status));
Item2.addActionListener(new processItem(status));
myWindow.addWindowListener(new WindowAdapter(){
    public void windowClosing(WindowEvent event){
        System.exit(0);
    }
});
myWindow.setSize(new Dimension(300, 100));
myWindow.show();
}
```

VÍ DỤ 1: MENU ĐƠN GIẢN

```
class processItem implements ActionListener{
    Label status;
    processItem(Label status){
        this.status=status;
    }
    public void actionPerformed(ActionEvent evt){
        if (evt.getSource() instanceof MenuComponent){
            MenuItem Item= (MenuItem)evt.getSource();
            status.setText("You have selected : " +Item.getLabel());
        }
    }
}
```



VÍ DỤ 2: (Checkbox Menu)

```
import java.awt.*;
import java.awt.event.*;
public class MenuDemo
    extends Frame
        implements ActionListener, ItemListener {
    MenuBar mb; //File, Options, Help
    Menu fm, om, hm; //Options Sub-Menu
    Menu opSubm; //The MenuItem for exiting
    MenuItem exitItem; //An option that can be on or off
    CheckboxMenuItem cb;
    Label label;
    MenuDemo(String s) {
        super("MenuDemo: " + s);
        Container cp = this;
        cp.setLayout(new FlowLayout());
        mb = new MenuBar();
        setMenuBar(mb); // Frame implements MenuContainer
    }
    //xem tiếp ở slide tiếp theo
```


VÍ DỤ 2: (Checkbox Menu)

```
MenuItem mi;  
// The File Menu...  
fm = new Menu("File");  
    fm.add(mi = new MenuItem("Open", new MenuShortcut('O')));  
    mi.addActionListener(this);  
    fm.add(mi = new MenuItem("Close", new MenuShortcut('W')));  
    mi.addActionListener(this);  
    fm.addSeparator();  
    fm.add(mi = new MenuItem("Print", new MenuShortcut('P')));  
    mi.addActionListener(this);  
    fm.addSeparator();  
    fm.add(mi = new MenuItem("Exit", new MenuShortcut('Q')));  
    exitItem = mi;      // save for action handler  
    mi.addActionListener(this);  
    mb.add(fm);  
//xem tiếp ở slide tiếp theo
```


VÍ DỤ 2: (Checkbox Menu)

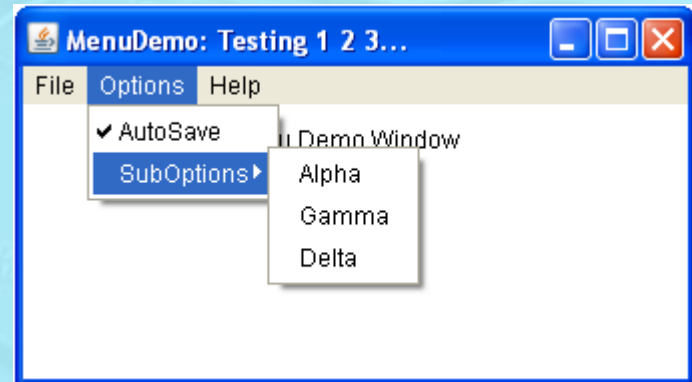
```
// The Options Menu...
om = new Menu("Options");
cb = new CheckboxMenuItem("AutoSave");
cb.setState(true); cb.addItemListener(this);
om.add(cb);
opSubm = new Menu("SubOptions"); opSubm.add(new MenuItem("Alpha"));
opSubm.add(new MenuItem("Gamma")); opSubm.add(new MenuItem("Delta"));
om.add(opSubm);
mb.add(om);
// The Help Menu...
hm = new Menu("Help");
    hm.add(mi = new MenuItem("About")); mi.addActionListener(this);
    hm.add(mi = new MenuItem("Topics")); mi.addActionListener(this);
mb.add(hm);
mb.setHelpMenu(hm);    // needed for portability (Motif, etc.).
// the main window
label = new Label("Menu Demo Window"); label.setSize(200, 150); cp.add(label);
pack();
} //xem tiếp ở slide tiếp theo
```

VÍ DỤ 2: (Checkbox Menu)

```
/** Handle action events. */
public void actionPerformed(ActionEvent evt) {
    // System.out.println("Event " + evt);
    String cmd;
    if ((cmd = evt.getActionCommand()) == null) System.out.println("You chose a menu shortcut");
    else System.out.println("You chose " + cmd);
    Object cmp = evt.getSource();
    // System.out.println("Source " + cmp);
    if (cmp == exitItem)
        System.exit(0);
}

/** The CheckBoxMenuItems send a different message */
public void itemStateChanged(ItemEvent e) {
    System.out.println("AutoSave is set " + cb.getState());
}

public static void main(String[] arg) {
    new MenuDemo("Testing 1 2 3...").setVisible(true);
}
}
```



VÍ DỤ 3: (Popup Menu)

```
class PopupMenuDemo extends Frame
{
    PopupMenu pMenu = new PopupMenu();
    MenuItem mnuCopy = new MenuItem("Copy");
    MenuItem mnuCut = new MenuItem("Cut");
    MenuItem mnuPaste = new MenuItem("Paste");
    PopupMenuDemo() // Constructor of a frame
    { ...
        pMenu.add(mnuCopy); // setup popup menu
        pMenu.addSeparator();
        pMenu.add(mnuCut);
        pMenu.addSeparator();
        pMenu.add(mnuPaste);
        // Add popup menu to the frame
        this.add(pMenu);
    }
    //xem tiếp ở slide tiếp theo
```



VÍ DỤ 3: (Popup Menu)

```
// In constructor of a frame
// Add mouse Listener for showing popup menu
addMouseListener ( new MouseAdapter()
{ public void mouseReleased(MouseEvent e)
{ if (e.isPopupTrigger()) // check right clicked
    pMenu.show(e.getComponent(),
                e.getX(),e.getY());
}
} );
```

**The
right-clicked
position**

The background of the slide is a close-up, slightly blurred image of various US coins, including pennies, nickels, and quarters, scattered across the surface. A solid green horizontal band is overlaid in the center, containing the title text.

PHẦN 4

XỬ LÝ SỰ KIỆN CHUỘT

XỬ LÝ SỰ KIỆN CHUỘT

- Java cung cấp hai interfaces lắng nghe (bộ lắng nghe sự kiện chuột) là `MouseListener` và `MouseMotionListener` để quản lý và xử lý các sự kiện liên quan đến thiết bị chuột.
- Những sự kiện chuột có thể “bẫy” cho bất kỳ component nào trên GUI mà dẫn xuất từ `java.awt.component`.

XỬ LÝ SỰ KIỆN CHUỘT

Các phương thức của interface `MouseListener`:

- **`public void mousePressed(MouseEvent event)`**: được gọi khi một nút chuột được nhấn và con trỏ chuột ở trên component.
- **`public void mouseClicked(MouseEvent event)`**: được gọi khi một nút chuột được nhấn và nhả trên component mà không di chuyển chuột.
- **`public void mouseReleased(MouseEvent event)`**: được gọi khi một nút chuột nhả sau khi kéo rê.
- **`public void mouseEntered(MouseEvent event)`**: được gọi khi con trỏ chuột vào trong đường biên của một component.
- **`public void mouseExited(MouseEvent event)`**: được gọi khi con trỏ chuột ra khỏi đường biên của một component.

XỬ LÝ SỰ KIỆN CHUỘT

Các phương thức của interface `MouseEventListener`:

- **`public void mouseDragged(MouseEvent event)`**: phương thức này được gọi khi người dùng nhấn một nút chuột và kéo trên một component.
- **`public void mouseMoved(MouseEvent event)`**: phương thức này được gọi khi di chuyển chuột trên component.

Mỗi phương thức xử lý sự kiện chuột có một tham số. `MouseEvent` chứa thông tin về sự kiện chuột phát sinh chẳng hạn như: tọa độ x, y nơi sự kiện chuột xảy ra. Những phương thức tương ứng trong các interfaces sẽ tự động được gọi khi chuột tương tác với một component.

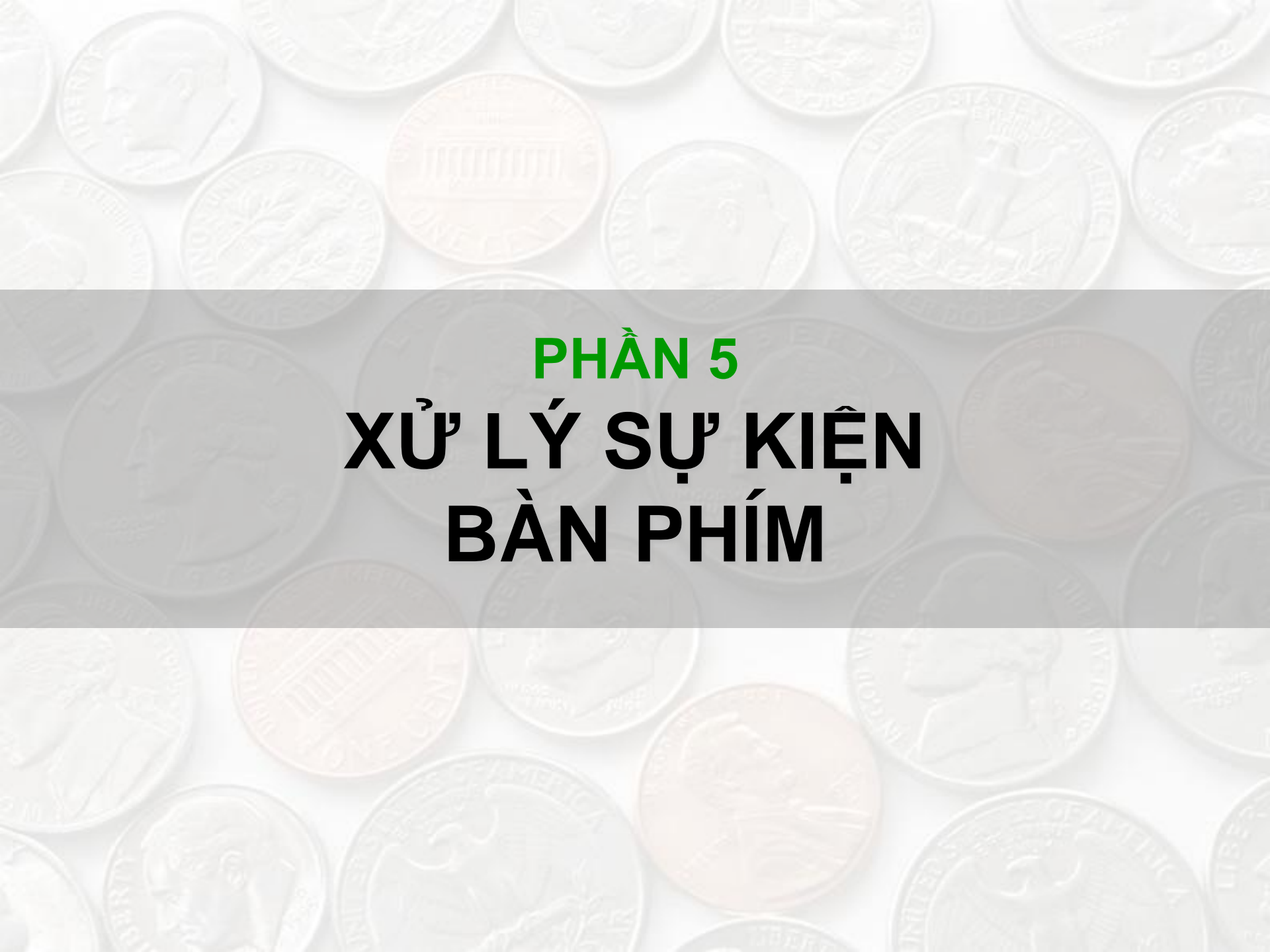
XỬ LÝ SỰ KIỆN CHUỘT

Ví dụ: Chương trình tên MouseTracker bên dưới minh họa việc dùng những phương thức của các interfaces `MouseListener` và `MouseMotionListener` để “bắt” và xử lý các sự kiện chuột tương ứng.

```
import java.awt.*; import java.awt.event.*;
public class MouseTracker extends Frame implements MouseListener, MouseMotionListener
{
    private Label statusBar;
    public MouseTracker()
    {
        super( "Demonstrating Mouse Events" );
        statusBar = new Label();
        this.add( statusBar, BorderLayout.SOUTH );
        addMouseListener( this );
        addMouseMotionListener( this );
        setSize( 275, 100 );
        setVisible( true );
    }
    public void mouseClicked( MouseEvent event )
    {
        String str_bt = new String();
        int count = event.getClickCount();
        int mousebutton = event.getButton();
        if(mousebutton == MouseEvent.BUTTON1) str_bt = "left mouse button";
        if(mousebutton == MouseEvent.BUTTON3) str_bt = "right mouse button";
        if(mousebutton == MouseEvent.BUTTON2) str_bt = "middle mouse button";
        statusBar.setText(str_bt + " clicked at (" + event.getX() + ","
                                + event.getY() + ")" + count + " lần");
    }
} //xem ở slide kế tiếp
```

XỬ LÝ SỰ KIỆN CHUỘT

```
public void mousePressed( MouseEvent event )
{
    statusBar.setText("Pressed at [" + event.getX() + ", " + event.getY() + "]" );
}
public void mouseReleased( MouseEvent event )
{
    statusBar.setText("Released at [" + event.getX() + ", " + event.getY() + "]" );
}
public void mouseEntered( MouseEvent event )
{
    statusBar.setText( "Mouse in window" );
}
public void mouseExited( MouseEvent event )
{
    statusBar.setText( "Mouse outside window" );
}
public void mouseDragged( MouseEvent event )
{
    statusBar.setText("Dragged at [" + event.getX() + ", " + event.getY() + "]" );
}
public void mouseMoved( MouseEvent event )
{
    statusBar.setText("Moved at [" + event.getX() + ", " + event.getY() + "]" );
}
public static void main( String args[] )
{
    MouseTracker application = new MouseTracker();
}
}
```

The background of the slide is a dense, overlapping pattern of various US coins, including pennies, nickels, and quarters, rendered in a light, semi-transparent style. A horizontal grey band is centered across the image, containing the title text.

PHẦN 5 **XỬ LÝ SỰ KIỆN** **BÀN PHÍM**

XỬ LÝ SỰ KIỆN BÀN PHÍM

- Để xử lý sự kiện bàn phím java hỗ trợ một bộ lắng nghe sự kiện đó là interface *KeyListener*. Một sự kiện bàn phím được phát sinh khi người dùng nhấn và nhả một phím trên bàn phím. Một lớp hiện thực *KeyListener* phải cài đặt các phương thức *keyPressed*, *keyReleased* và *keyTyped*. Mỗi phương thức này có một tham số là một đối tượng kiểu *KeyEvent*. *KeyEvent* là lớp con của lớp *InputEvent*.

XỬ LÝ SỰ KIỆN BÀN PHÍM

- Các phương thức của interface *KeyListener*
 - Phương thức *keyPressed* được gọi khi một phím bất kỳ được nhấn.
 - Phương thức *keyTyped* được gọi thực hiện khi người dùng nhấn một phím không phải “phím hành động” (như phím mũi tên, phím Home, End, Page Up, Page Down, các phím chức năng như: Num Lock, Print Screen, Scroll Lock, Caps Lock, Pause).
 - Phương thức *keyReleased* được gọi thực hiện khi nhả phím nhấn sau khi sự kiện *keyPressed* hoặc *keyTyped*

XỬ LÝ SỰ KIỆN BÀN PHÍM

Ví dụ: minh họa việc xử lý sự kiện chuột thông qua các phương thức của interface *KeyListener*. Lớp *KeyDemo* bên dưới hiện thực interface *KeyListener*, vì vậy tất cả 3 phương thức trong *KeyListener* phải được cài đặt trong chương trình.

```
// KeyDemo.java
// Demonstrating keystroke events.
// Java core packages
import java.awt.*;
import java.awt.event.*;
public class KeyDemo extends Frame implements KeyListener
{
    private String line1 = "", line2 = "";
    private String line3 = "";
    private TextArea textArea;
    //xem tiếp ở slide tiếp theo
```

XỬ LÝ SỰ KIỆN BÀN PHÍM

```
// set up GUI
public KeyDemo()
{
    super( "Demonstrating Keystroke Events" );
    textArea = new TextArea( 10, 15 ); // set up TextArea
    textArea.setText( "Press any key on the keyboard..." );
    textArea.setEnabled( false );
    this.add( textArea );
    addKeyListener( this ); // allow frame to process Key events
    setSize( 350, 100 );
    setVisible( true );
}
// handle press of any key
public void keyPressed( KeyEvent event )
{
    line1 = "Key pressed: " +
        event.getKeyText( event.getKeyCode() );
    setLines2and3( event );
} //xem tiếp ở slide tiếp theo
```


XỬ LÝ SỰ KIỆN BÀN PHÍM

```
// handle release of any key
public void keyReleased( KeyEvent event )
{
    line1 = "Key released: " + event.getKeyText( event.getKeyCode() );
    setLines2and3( event );
}
// handle press of an action key
public void keyTyped( KeyEvent event )
{
    line1 = "Key typed: " + event.getKeyChar();
    setLines2and3( event );
}
//xem tiếp ở slide tiếp theo
```


XỬ LÝ SỰ KIỆN BÀN PHÍM

```
// set second and third lines of output
private void setLines2and3( KeyEvent event )
{
    line2 = "This key is " + ( event.isActionKey() ? "" : "not" ) + "an action key";
    String temp = event.getKeyModifiersText(event.getModifiers() );
    line3 = "Modifier keys pressed: " + ( temp.equals( "" )?"none": temp );
    textArea.setText(line1+"\n"+line2+"\n"+ line3+"\n" );
}
// execute application
public static void main( String args[] )
{
    KeyDemo application = new KeyDemo();
}
} // end class KeyDemo
```

The background of the slide is a dense, overlapping pattern of various US coins, including pennies, nickels, and quarters, rendered in a light, faded style. A solid purple horizontal band is superimposed over the center of the image.

HẾT BÀI 6