

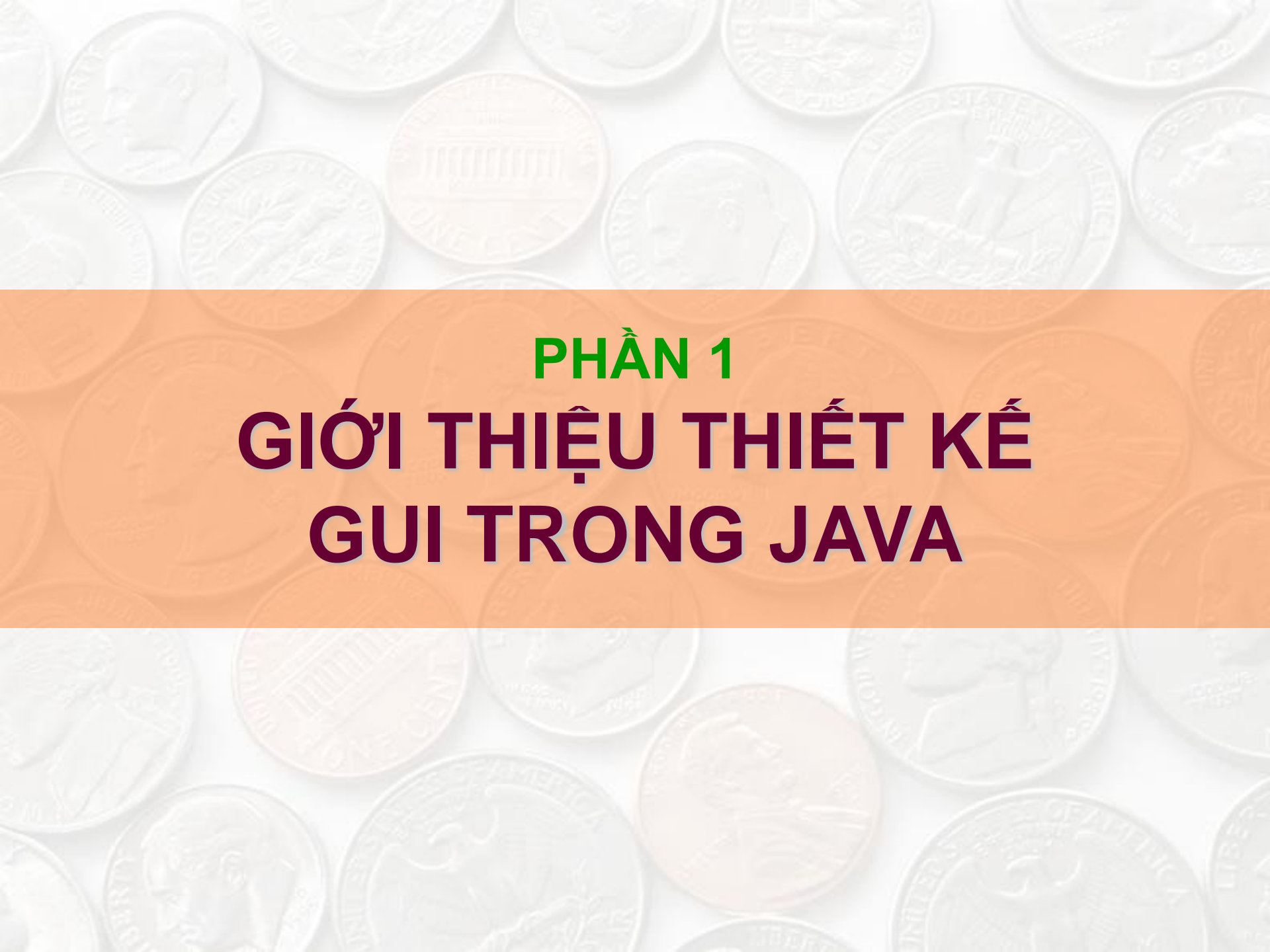
The background of the slide is a dense, overlapping pattern of various US coins, including pennies, nickels, and quarters, rendered in a light, faded grey tone. A solid orange horizontal band spans the middle of the image, serving as a backdrop for the text.

NHẬP MÔN JAVA
BÀI 4

LẬP TRÌNH GIAO DIỆN (GUI)

NỘI DUNG ĐƯỢC TRÌNH BÀY GỒM:

- Giới thiệu thiết kế GUI trong java
- Các thành phần cơ bản (Component)
- Đối tượng khung chứa (Container)
- Bộ quản lý trình bày (Layout Manager)

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

GIỚI THIỆU THIẾT KẾ GUI TRONG JAVA

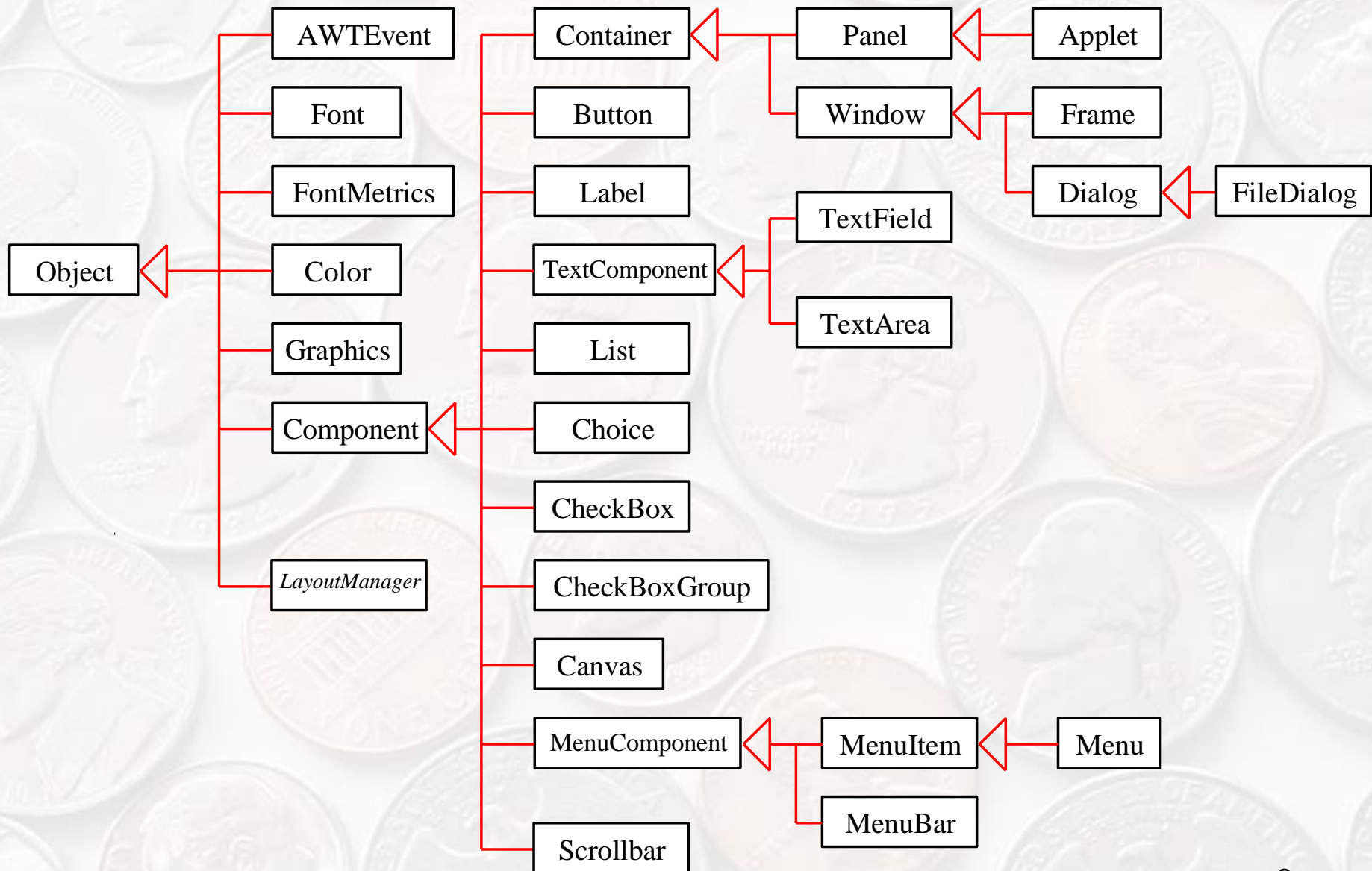
GIỚI THIỆU VỀ THIẾT KẾ GUI

- Thư viện hỗ trợ: tập hợp các lớp java cung cấp hỗ trợ thiết kế, xây dựng GUI (Graphic User Interface) là:
 - awt (java.awt.*)
 - swing (javax.swing.*)

GIỚI THIỆU AWT

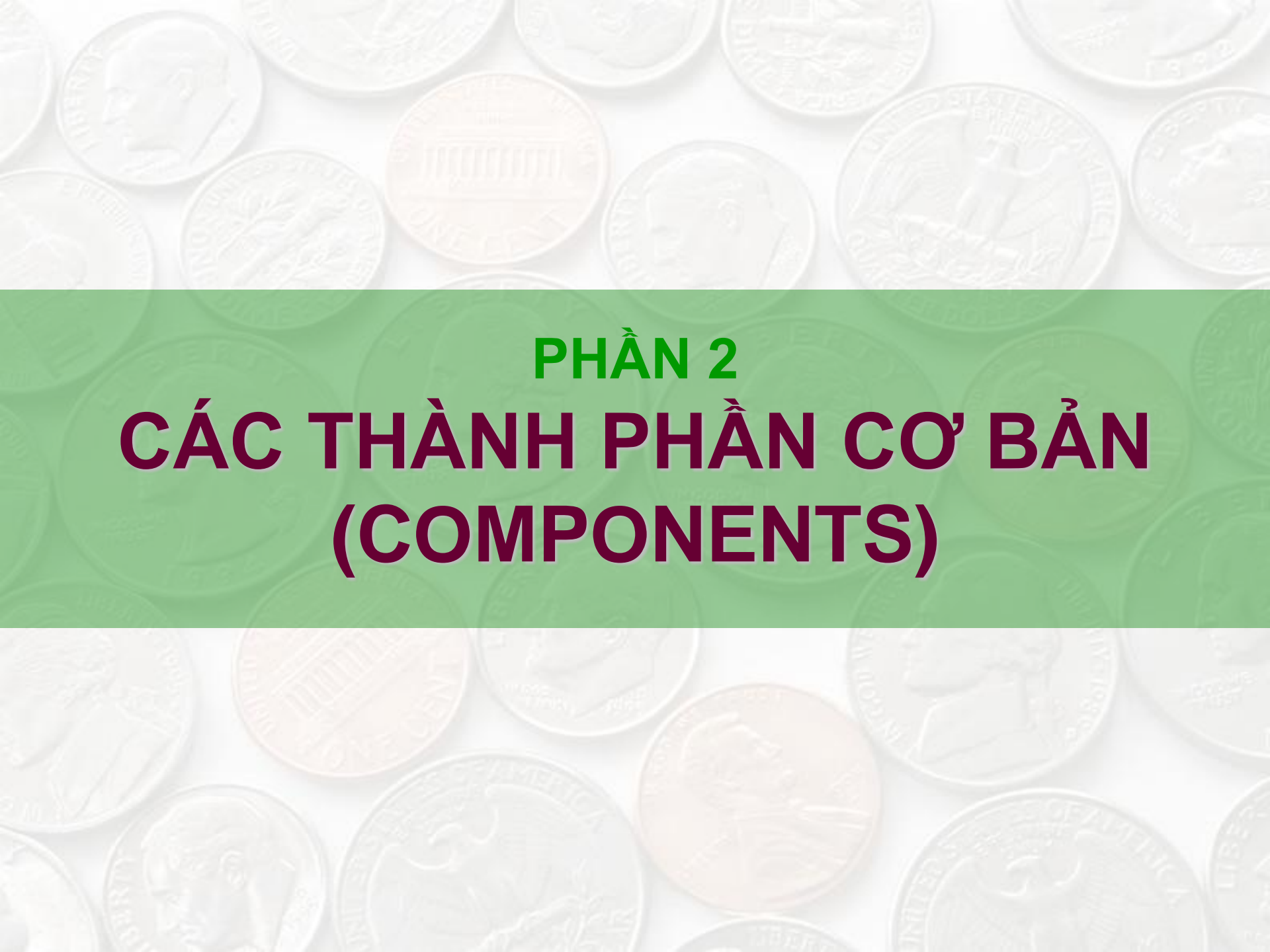
- AWT viết tắt của **Abstract Windowing Toolkit**
- AWT là tập hợp các lớp Java cho phép chúng ta tạo một GUI.
- Cung cấp các mục khác nhau để tạo hoạt động và hiệu ứng GUI
 - `import java.awt.*;`
 - `import java.awt.event.*;`

GIỚI THIỆU AWT



NGUYÊN TẮC XÂY DỰNG GUI

- Lựa chọn một container: Frame, Window, Dialog, Applet,...
- Tạo các control: (buttons, text areas, list, choice, checkbox,...)
- Đưa các control vào vùng chứa
- Sắp xếp các control trong vùng chứa (Layout).
- Thêm các xử lý sự kiện (Listeners)

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.

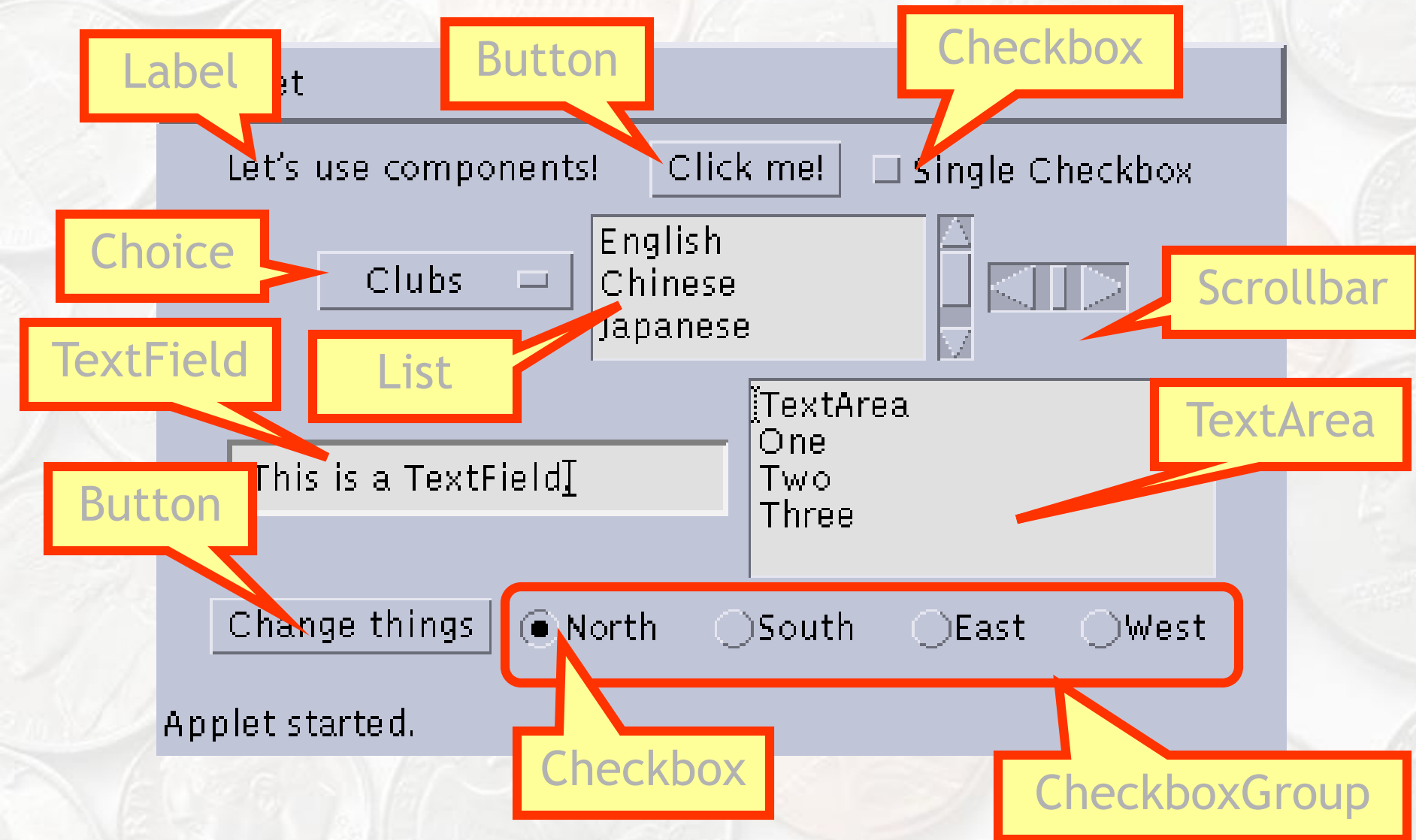
PHẦN 2

CÁC THÀNH PHẦN CƠ BẢN (COMPONENTS)

CÁC COMPONENTS CỦA GUI

- Tất cả các thành phần cấu tạo nên chương trình GUI được gọi là component.
- Ví dụ
 - Frame, Window, Dialog, Applet,...
 - TextFields, Labels, CheckBoxes, TextArea, Button, Choice, List, Scrollbars,...

CÁC COMPONENTS CỦA GUI



NHÃN (LABEL)

- Nhãn được dùng để trình bày một chuỗi văn bản ra màn hình
- Một số phương thức của Label:

`public Label();` // tạo nhãn

`public Label(String s);` // tạo nhãn với nội dung s

`public Label(String s, int align);` // tạo và canh lề

`void setText(String s);` // đặt nội dung nhãn

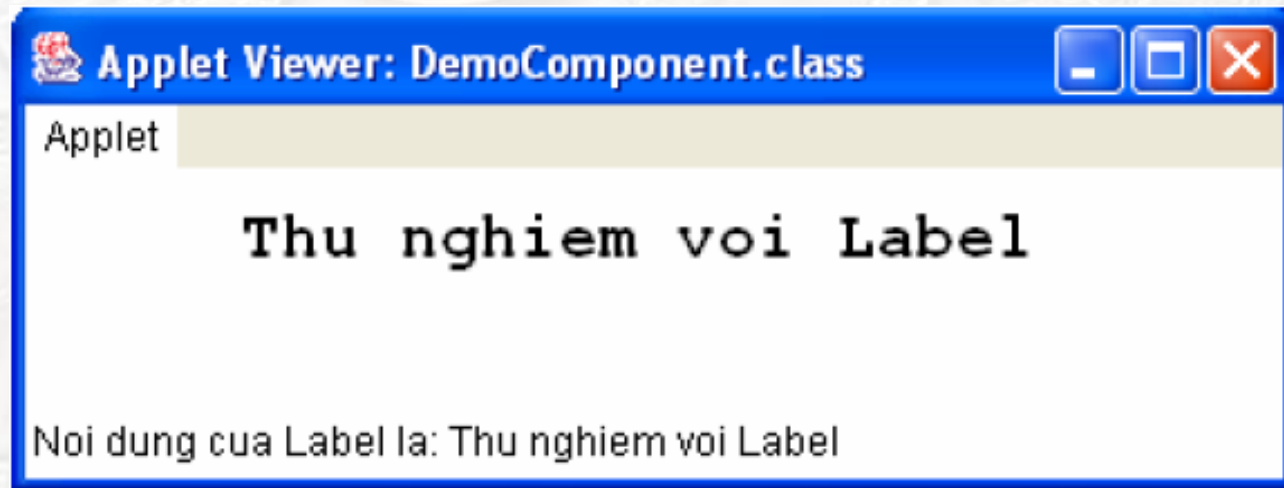
`void setAlignment(int align);` // canh lề nhãn

...

NHÃN (LABEL)

```
import java.applet.Applet;
import java.awt.*;
public class DemoLabel extends Applet
{
    private Label label;
    public void init()
    {
        Font font = new Font("Courier", Font.BOLD, 20);
        label = new Label("Thu nghiem voi Label");
        label.setFont(font);
        add(label);
    }
    public void paint(Graphics g)
    {
        showStatus("Noi dung cua Label la: " + label.getText());
    }
}
```


NHÃN (LABEL)



NÚT NHẤN (BUTTON)

- Một số phương thức của Button
 - Button(); // tạo nút nhấn
 - Button(String s); // tạo nút nhấn có tên s
 - void setLabel(String s); // đổi tên nút
 - String getLabel(); // lấy tên nút nhấn
- Để lắng nghe sự kiện nhấn nút ta cần cài đặt giao tiếp **ActionListener**.

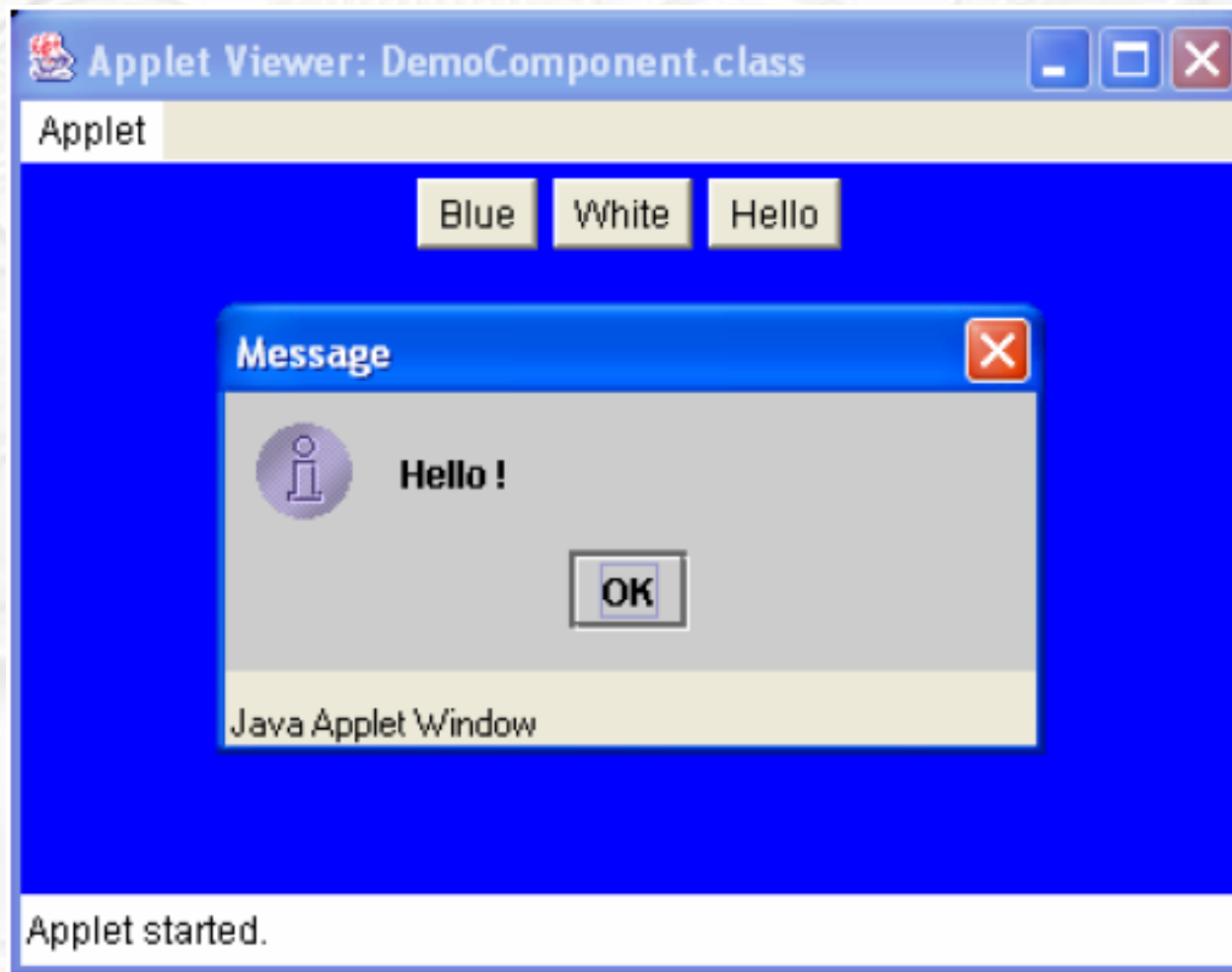
NÚT NHẤN (BUTTON)

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class DemoButton extends Applet implements ActionListener
{
    private Button blueButton;
    private Button whiteButton;
    private Button helloButton;
    public void init()
    {
        blueButton = new Button("Blue");
        whiteButton = new Button("White");
        helloButton = new Button("Hello");
        blueButton.addActionListener(this);
        whiteButton.addActionListener(this);
        helloButton.addActionListener(this);
    }
    //xem tiếp ở slide kế tiếp
}
```

NÚT NHẤN (BUTTON)

```
add(blueButton);
add(whiteButton);
add(helloButton);
}
public void actionPerformed(ActionEvent event)
{
    if(event.getSource() == helloButton)
        javax.swing.JOptionPane.showMessageDialog(this, "Hello !");
    else{
        if (event.getSource() == blueButton)
            this.setBackground(Color.BLUE);
        else if (event.getSource() == whiteButton)
            this.setBackground(Color.WHITE);
        repaint();
    }
}
}
```


NÚT NHẤN (BUTTON)



Ô VĂN BẢN (TEXT FIELD)

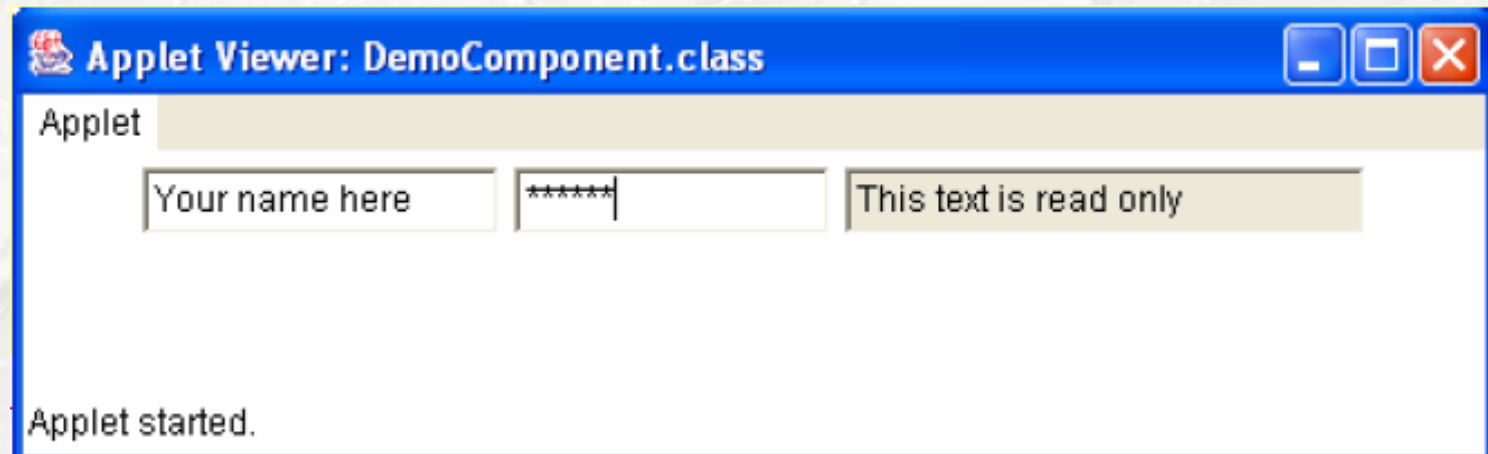
- Ô văn bản cho phép nhận dữ liệu từ bàn phím trên một dòng
- Một số phương thức
 - `TextField(...);` // các cấu tử
 - `void setEditable(boolean b);` // đặt/tắt chế độ nhập
 - `void setEchoChar(char c);` // đặt kí tự hiển thị
- Đối tượng nghe cần cài đặt 2 giao tiếp
 - ***ActionListener***
 - ***TextListener***
 - Cài đặt phương thức ***textValueChanged();***

Ô VĂN BẢN (TEXT FIELD)

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class DemoTextField extends Applet implements ActionListener
{
    private TextField txtEdit;
    private TextField txtReadOnly;
    private TextField txtPass;
    private final String PASSWORD = "Java";
    public void init()
    {
        txtEdit = new TextField("Your name here");
        txtPass = new TextField(12);
        txtPass.setEchoChar('*');
        txtPass.addActionListener(this);
        txtReadOnly = new TextField("This text is read only");
        txtReadOnly.setEditable(false);
    }
    // xem tiếp ở slide kế tiếp
```

Ô VĂN BẢN (TEXT FIELD)

```
add(txtEdit);  
add(txtPass);  
add(txtReadOnly);  
}  
public void actionPerformed(ActionEvent event)  
{  
    if(txtPass.getText().equals(PASSWORD))  
        txtReadOnly.setText("Password is valid");  
    else  
        txtReadOnly.setText("Invalid password !");  
}  
}
```



LỰA CHỌN (CHOICE)

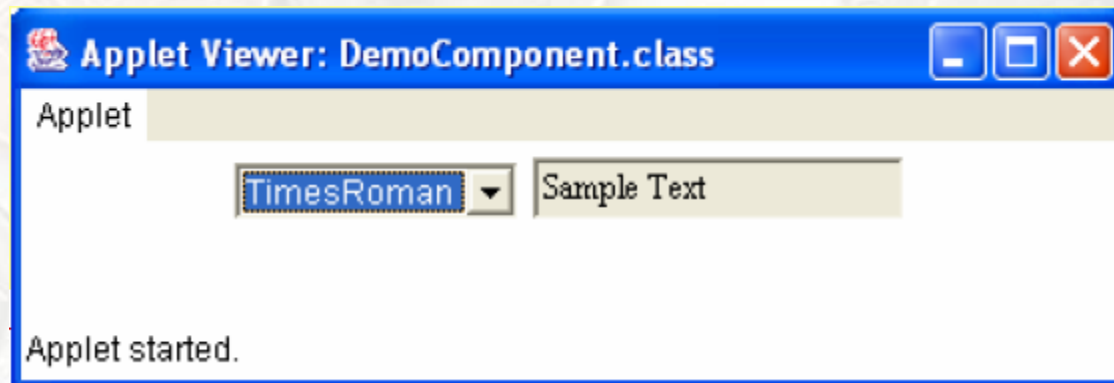
- Choice cung cấp khả năng lựa chọn một trong số các hạng mục sẵn có.
- Một số phương thức
 - Choice(); // cấu tử
 - void addItem(String s); // thêm item là s
 - String getItem(int index); // lấy item có chỉ số index
 - String getSeclectedItem(); // trả về item được chọn
 - int getSelectedIndex(); // trả về index của item được chọn
- Lớp nghe cài đặt giao tiếp ***ItemListener***
 - Cài đặt phương thức ***itemStateChanged(...)***

LỰA CHỌN (CHOICE)

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class DemoChoice extends Applet implements ItemListener
{
    private Choice choice;
    private TextField txtText;
    private Font font;
    public void init()
    {
        choice = new Choice();
        choice.addItem("TimesRoman");
        choice.addItem("Courier");
        choice.addItem("Helvetica");
        choice.addItemListener(this);
    }
    // xem tiếp ở slide kế tiếp
```

LỰA CHỌN (CHOICE)

```
txtText = new TextField("Sample Text", 16);
txtText.setEditable(false);
font = new Font(choice.getItem(0),Font.PLAIN, 12);
txtText.setFont(font);
add(choice);
add(txtText);
}
public void itemStateChanged(ItemEvent event)
{
    font = new Font(choice.getSelectedItem(), Font.PLAIN, 12);
    txtText.setFont(font);
}
}
```



CHECK BOX (HỘP ĐÁNH DẤU)

Checkbox cung cấp các hộp tùy chọn cho người dùng

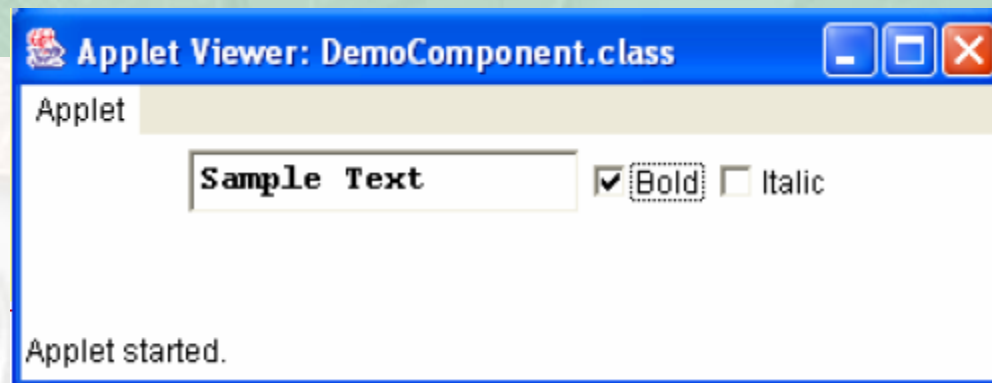
- Một số phương thức
 - `Checkbox(...);` // các cấu tử
 - `void setLabel(Strings);` // đặt nhãn mới
 - `boolean getState();` // lấy trạng thái hiện tại
- Lớp nghe cài đặt giao tiếp ***ItemListener***
 - Cài đặt phương thức ***itemStateChanged(...)***

CHECK BOX (HỘP ĐÁNH DẤU)

```
import java.applet.Applet;
Import java.awt.*;
Import java.awt.event.*;
public class DemoCheckbox extends Applet implements ItemListener
{
    private Checkbox checkBold;
    private Checkbox checkItalic;
    private TextField txtText;
    public void init()
    {
        checkBold = new Checkbox("Bold");
        checkItalic = new Checkbox("Italic");
        checkBold.addItemListener(this);
        checkItalic.addItemListener(this);
        txtText = new TextField("Sample Text", 16);
        Font font = new Font("Courier", Font.PLAIN, 14);
        txtText.setFont(font);\
//xem tiếp ở slide kế tiếp
```

CHECK BOX (HỘP ĐÁNH DẤU)

```
add(txtText);  
add(checkBold);  
add(checkItalic);  
}  
public void itemStateChanged(ItemEvent event)  
{  
    int valBold = Font.PLAIN;  
    int valItalic = Font.PLAIN;  
    if(checkBold.getState()) valBold = Font.BOLD;  
    if(checkItalic.getState()) valItalic = Font.ITALIC;  
    Font font = new Font("Courier", valBold + valItalic, 14);  
    txtText.setFont(font);  
}  
}
```



CHECK BOX GROUP & RADIO BUTTON

- Các Checkbox có thể được đặt trong một CheckboxGroup để tạo ra các Radio Button.
- Ví dụ: Tạo 3 radio button

```
// Tạo 3 radio button thuộc cùng một nhóm. Ban đầu  
// radio1 được chọn. Tại mỗi thời điểm chỉ có thể chọn một // trong 3  
radio.
```

```
CheckboxGroupg = new CheckboxGroup();  
Checkbox radio1 = new Checkbox("Radio1", g, true);  
Checkbox radio2 = new Checkbox("Radio2", g, false);  
Checkbox radio3 = new Checkbox("Radio3", g, false);
```

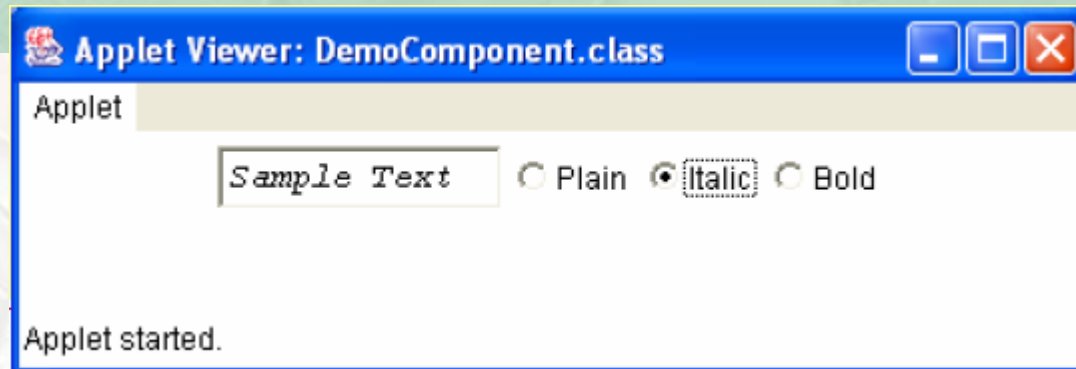

CHECK BOX GROUP & RADIO BUTTON

```
// Cac import can thiet...
public class DemoRadio extends Applet implements ItemListener
{
    private Checkbox plain, bold, italic;
    private CheckboxGroup group;
    private TextField txtText;
    public void init()
    {
        group = new CheckboxGroup();
        plain = new Checkbox("Plain", group, true);
        bold = new Checkbox("Bold", group, false);
        italic = new Checkbox("Italic", group, false);
        txtText = new TextField("Sample Text");
        txtText.setFont(new Font("Courier", Font.PLAIN, 14));
        plain.addItemListener(this);
        bold.addItemListener(this);
        italic.addItemListener(this);
    }
}
```

//xem tiếp ở slide tiếp theo

CHECK BOX GROUP & RADIO BUTTON

```
        add(txtText);  
        add(plain);  
        add(italic);  
        add(bold);  
    }  
    public void itemStateChanged(ItemEvent event)  
    {  
        int mode = 0;  
        if(event.getSource() == plain) mode = Font.PLAIN;  
        if(event.getSource() == italic) mode = Font.ITALIC;  
        if(event.getSource() == bold) mode = Font.BOLD;  
        txtText.setFont(newFont("Courier", mode, 14));  
    }  
}
```



DANH SÁCH (LIST)

- List cho phép người dùng chọn một hay nhiều item từ một danh sách các item
- Một số phương thức
 - `List();` // cấu tử mặc định
 - `List(int items, boolean ms);` // cấu tử mở rộng
 - `String getSeclectedItem();` // lấy lại thành phần được chọn
- Lớp nghe cài đặt giao tiếp `ItemListener` và/hoặc `ActionListener`

DANH SÁCH (LIST)

// Cac import can thiet...

public class DemoList extends Applet implements ItemListener, ActionListener

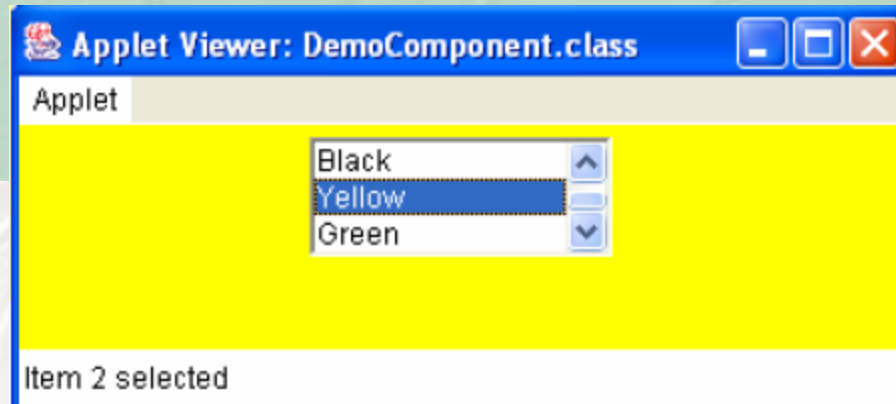
```
{  
    private List colorList;  
    public void init()  
    {  
        colorList = newList(3, false);  
        colorList.add("White");  
        colorList.add("Black");  
        colorList.add("Yellow");  
        colorList.add("Green");  
        colorList.addItemListener(this);  
        colorList.addActionListener(this);  
        add(colorList);  
    }  
}
```

//xem tiếp ở slide tiếp theo

DANH SÁCH (LIST)

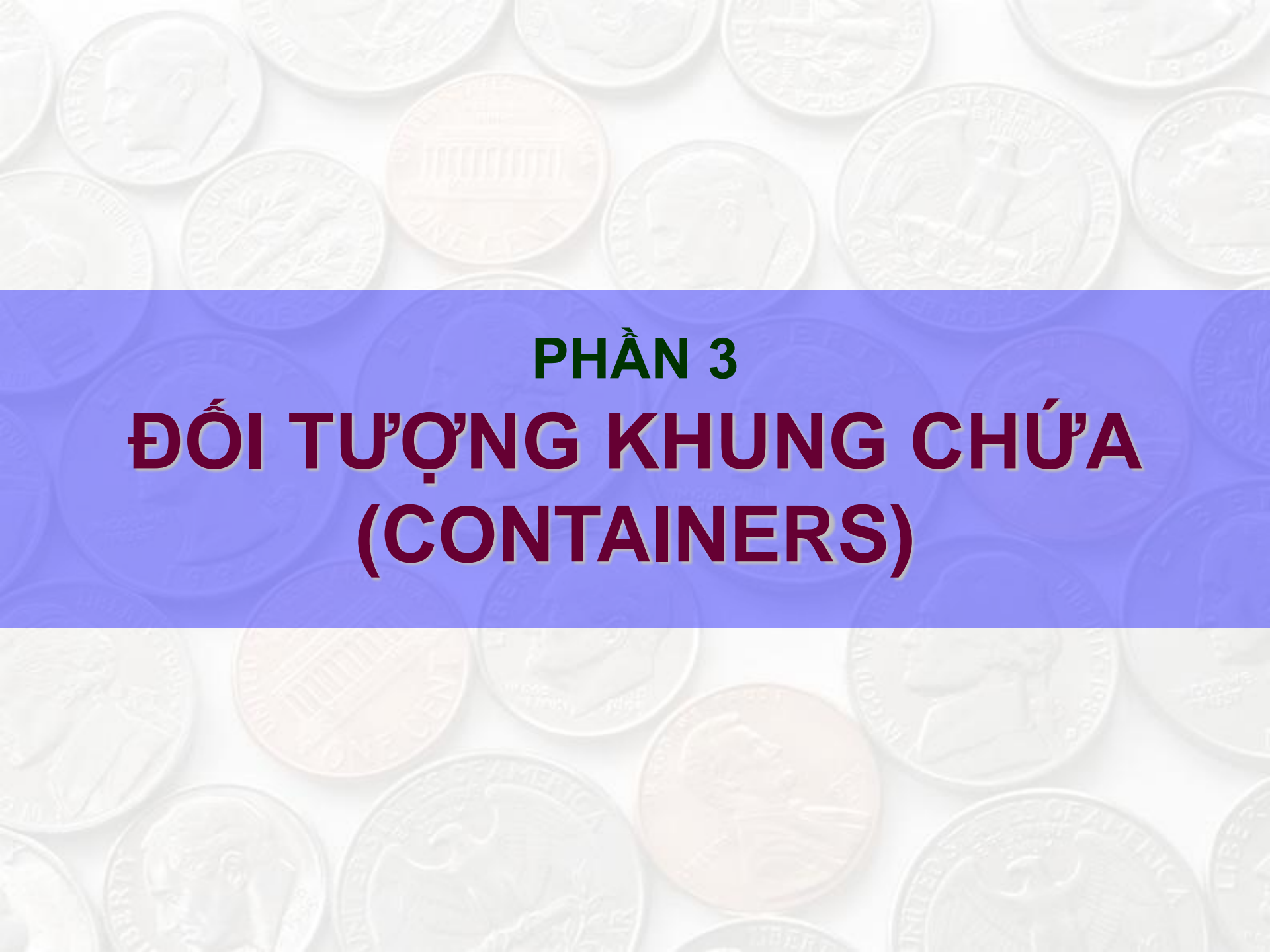
```
public void itemStateChanged(ItemEvent event)
{
    List list = (List) event.getSource();
    showStatus("Item " + list.getSelectedIndex() + " selected");
}

public void actionPerformed(ActionEvent event)
{
    List list = (List) event.getSource();
    Strings = list.getSelectedItem();
    if(s.equals("White")) setBackground(Color.WHITE);
    if(s.equals("Black")) setBackground(Color.BLACK);
    if(s.equals("Yellow")) setBackground(Color.YELLOW);
    if(s.equals("Green")) setBackground(Color.GREEN);
    repaint();
}
}
```



CÁC THÀNH PHẦN KHÁC

- Một số thành phần khác như: TextArea (vùng văn bản), Menu (thực đơn), ScrollBar (thanh trượt), Canvas (khung vẽ), Applet,... sẽ được trình bày ở các chương sau.

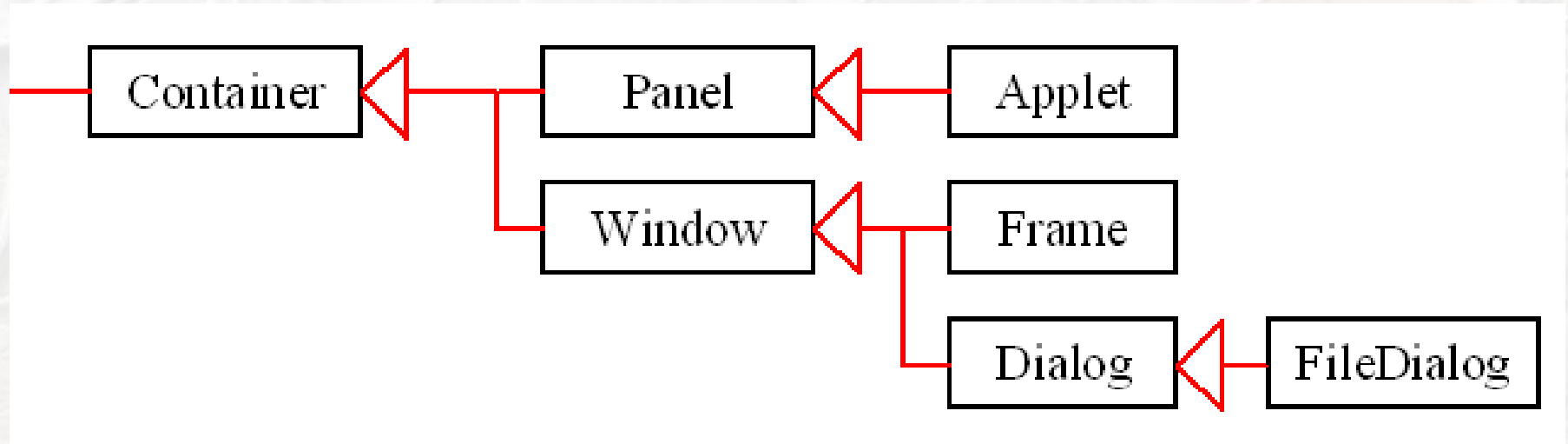
The background of the slide is a close-up, slightly blurred image of various US coins, including pennies, nickels, and quarters, scattered across the surface. A semi-transparent blue horizontal band is overlaid across the middle of the image, containing the title text.

PHẦN 3

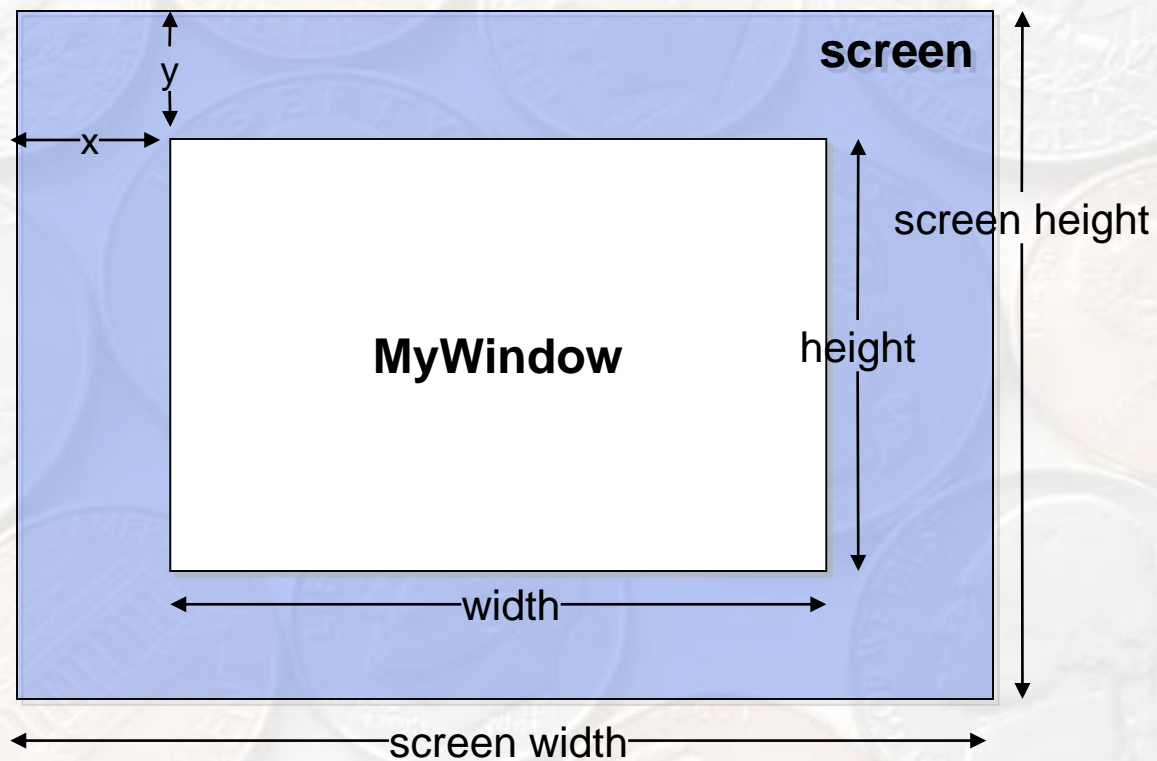
ĐỐI TƯỢNG KHUNG CHỨA (CONTAINERS)

CÁC ĐỐI TƯỢNG KHUNG CHỨA

- Là các thành phần mà có thể chứa các thành phần khác, có thể vẽ và tô màu.
- Gồm có: Frame, Applet, Panel, ScrollPane, Dialog, FileDialog.



CÁCH TÍNH TỌA ĐỘ



KHUNG CHỨA FRAME

- Frame được dùng để xây dựng các ứng dụng GUI chạy độc lập.
- Frame là một cửa sổ có thanh tiêu đề và các đường biên. Bố cục mặc định của Frame là BorderLayout.
- Frame kế thừa từ Window, nó có thể nghe các sự kiện xảy ra trên cửa sổ khi cài đặt giao tiếp **WindowListener**.
- Các ứng dụng độc lập thường tạo ra cửa sổ kế thừa từ lớp Frame.

KHUNG CHỨA FRAME

```
import java.awt.*;
import java.awt.event.*;
public class DemoFrame
{
    public static void main(String[] args)
    {
        Frame frame = new Frame("Example on Frame");
        Label label = new Label("This is a label in Frame", Label.CENTER);
        frame.add(label, BorderLayout.CENTER);
        frame.setSize(500,500);
        frame.setVisible(true);
        frame.addWindowListener(new MyWindowListener());
    }
}
//xem tiếp ở slide tiếp theo
```

KHUNG CHỨA FRAME

```
// Lop nghe doc lap (external listener)
Class MyWindowListener extends WindowAdapter
{
    public void windowClosing(WindowEvent event)
    {
        System.exit(0);
    }
}
```



KHUNG CHỨA FRAME

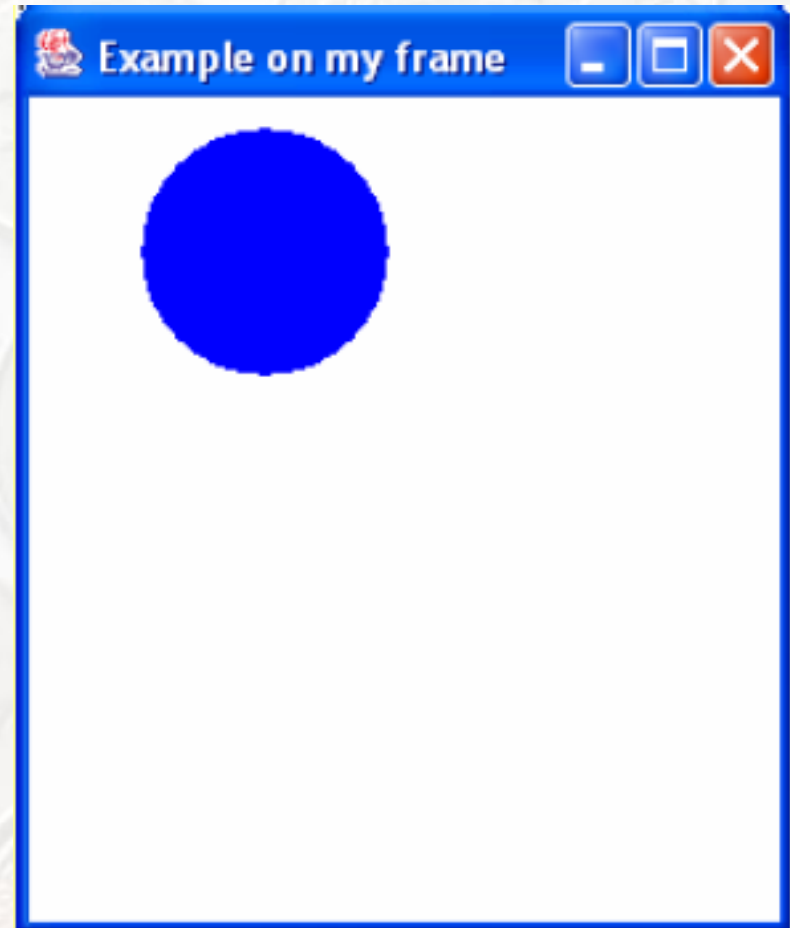
```
import java.awt.*;
import java.awt.event.*;
public class DemoFrame2
{
    public static void main(String[] args)
    {
        MyFrame myFrame = new MyFrame("Example on my frame");
        myFrame.setSize(250, 300);
        myFrame.setVisible(true);
        myFrame.addWindowListener(new WindowAdapter()
        {
            // Lop nghe noi khong ten (anonymous inner class listener)
            public void windowClosing(WindowEvent e)
            {
                System.exit(0);
            }
        });
    }
}
```

//xem tiếp ở slide tiếp theo

KHUNG CHỨA FRAME

Class MyFrame extends Frame

```
{  
    public MyFrame(String title)  
    {  
        super(title);  
    }  
    public void paint(Graphics g)  
    {  
        g.setColor(Color.BLUE);  
        g.fillOval(40, 40, 80, 80);  
    }  
}
```



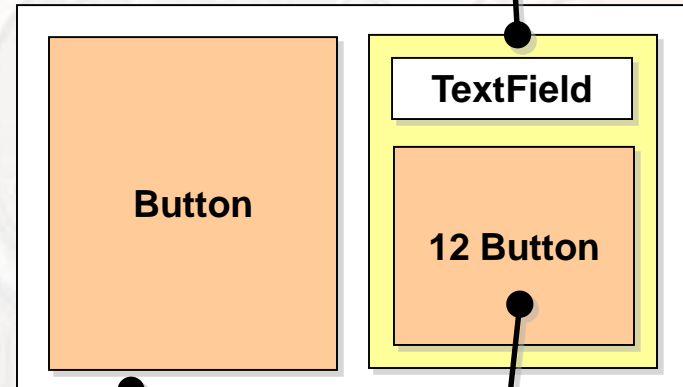
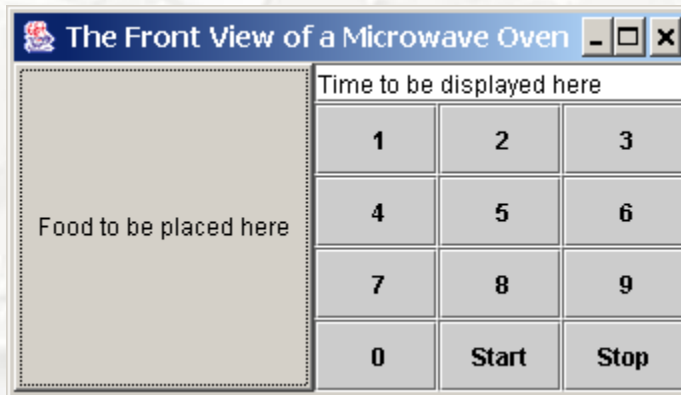
KHUNG CHỨA FRAME

- Chú ý:
 - Frame không có các phương thức `init`, `start`... như trong Applet.
 - Các ứng dụng độc lập dùng Frame phải có hàm `main` và được chạy trực tiếp bằng lệnh `java`.
 - Cần có lệnh `setSize`, `setVisible(true)` để có thể hiển thị Frame.
 - Ở cuối chương trình nên có lệnh: `System.exit(0);`

LỚP PANEL (VÙNG CHỨA)

- Lớp Panel kế thừa từ Container. Nó có thể được dùng để tạo ra các giao diện theo ý muốn.
- Ví dụ: Một giao diện có thể có nhiều panel sắp xếp theo một layout nhất định, mỗi panel lại có các component sắp xếp theo một layout riêng.
- Chú ý: Panel có bố cục mặc định là FlowLayout.

LỚP PANEL (VÙNG CHỨA)



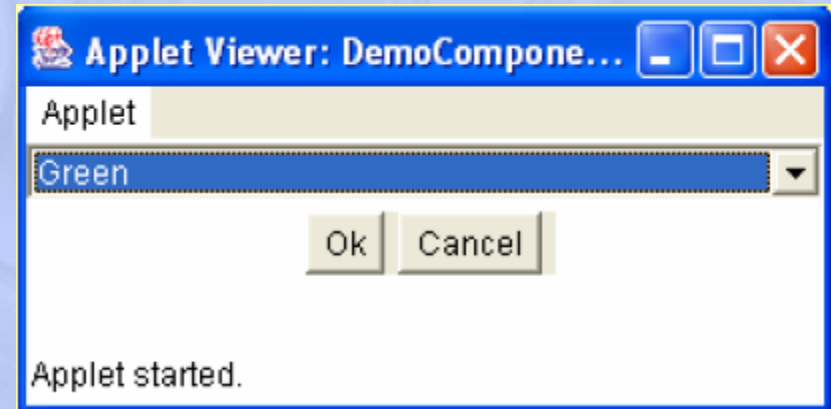
Panel (BorderLayout)

Panel (GridLayout)

Frame (BorderLayout)

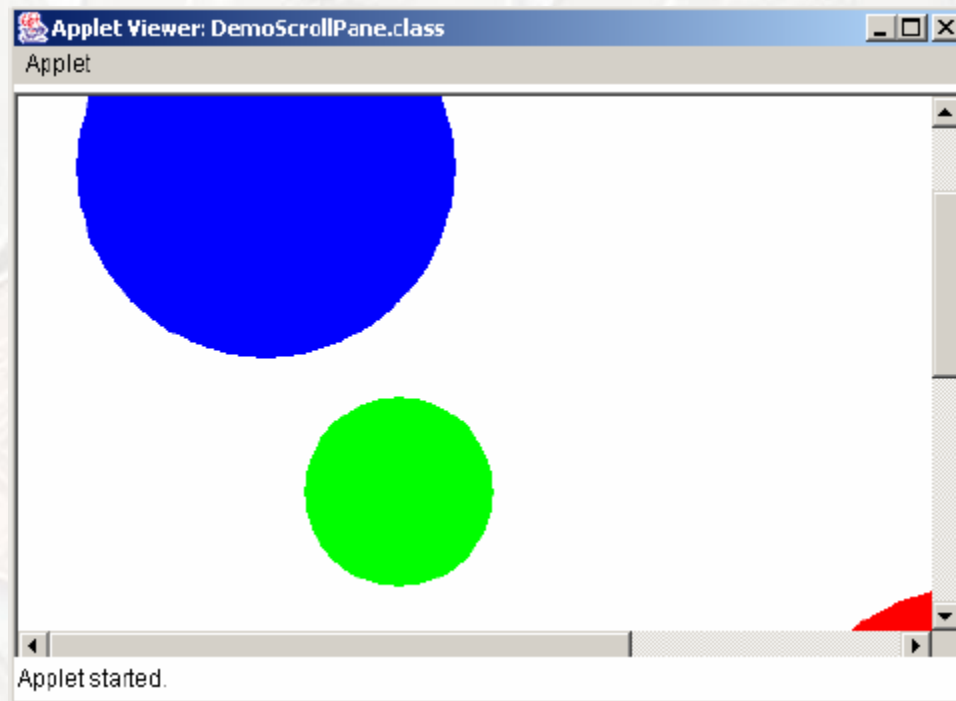
LỚP PANEL (VÙNG CHỨA)

```
public void init()
{
    Choice choice = new Choice();
    choice.add("Red");
    choice.add("Green");
    choice.add("Blue");
    Button ok = new Button("Ok");
    Button cancel = new Button("Cancel");
    Panel panel = new Panel();
    panel.add(ok);
    panel.add(cancel);
    this.setLayout(new BorderLayout());
    this.add(choice, BorderLayout.NORTH);
    this.add(panel, BorderLayout.CENTER);
}
```



KHUNG CUỘN (SCROLL PANE)

- Khung cuộn là một container cho phép chứa thành phần GUI có kích thước lớn hơn chính nó.
- Bài tập: Viết chương trình cho phép vẽ trong một canvas có độ rộng lớn hơn kích thước của applet. Đặt canvas vào trong một scroll pane.



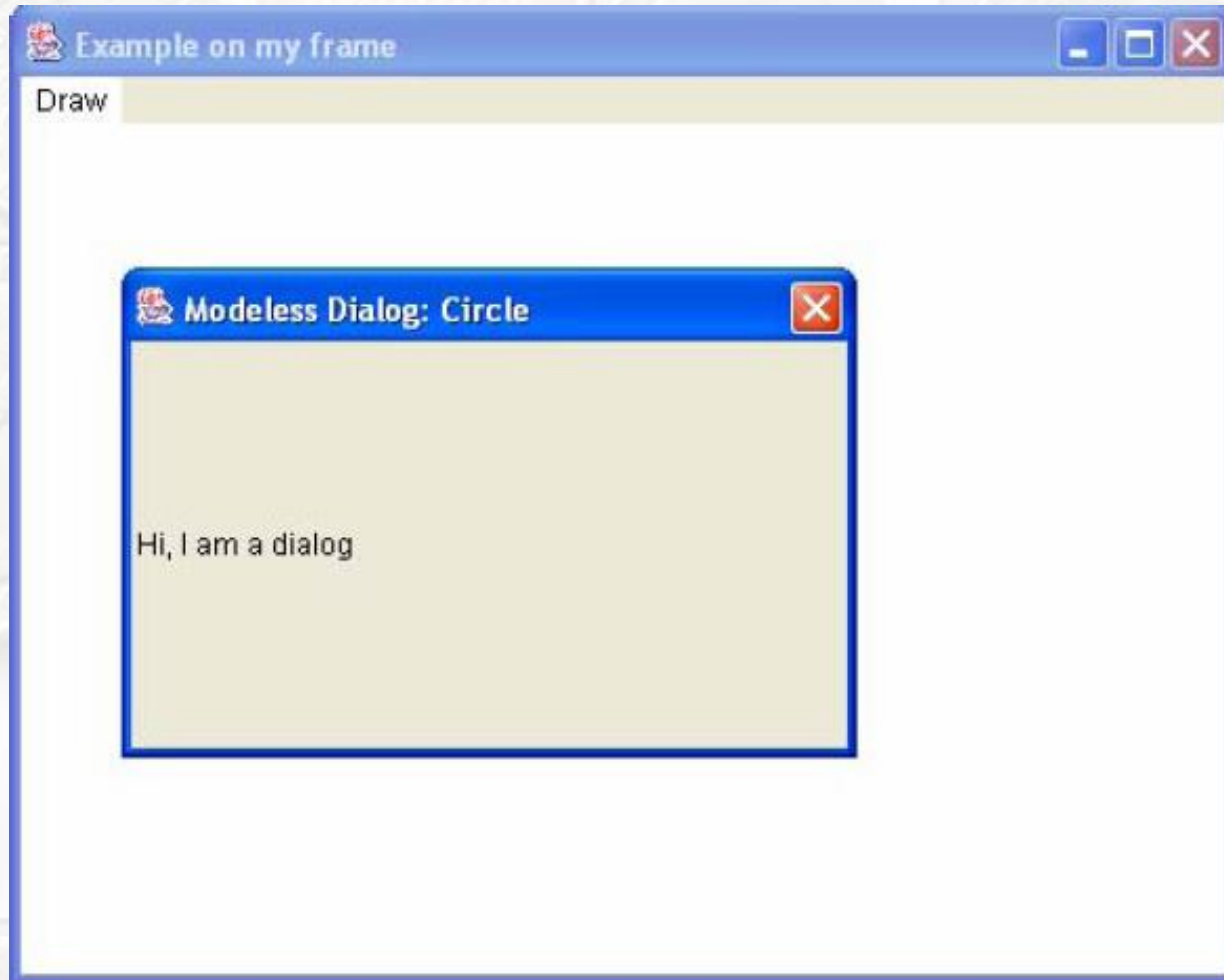
HỘP THOẠI (DIALOG)

- Dialog cũng là một cửa sổ, thường dùng để nhập hoặc hiển thị thông tin với người dùng.
- Hai loại hộp thoại
 - Modal: Phải đóng hộp thoại trước khi chuyển sang cửa sổ khác.
 - Modaleless: Có thể giữ nguyên hộp thoại và chuyển sang cửa sổ khác.

HỘP THOẠI (DIALOG)

- Dialog kế thừa từ lớp Window, nó có bố cục mặc định là BorderLayout.
- Hộp thoại có thể chứa các thành phần GUI và xử lý các sự kiện như một cửa sổ bình thường.

VÍ DỤ VỀ FRAME VÀ DIALOG



VÍ DỤ VỀ FRAME VÀ DIALOG

```
import java.awt.*;
import java.awt.event.*;
public class DemoFrame3
{
    public static void main(String[] args)
    {
        MyFrame myFrame = new MyFrame("Example on my frame");
        myFrame.setSize(500, 400);
        myFrame.setVisible(true);
        myFrame.addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e){System.exit(0);
        });
    }
}
```

VÍ DỤ VỀ FRAME VÀ DIALOG

Class MyFrame extends Frame implements ActionListener

```
{  private MenuBar menuBar;
    private Menu menu;
    private MenuItem circleItem, rectItem;
    public MyFrame(String title)
    {    super(title);
        menuBar = new MenuBar();
        setMenuBar(menuBar);
        menu = new Menu("Draw");
        menuBar.add(menu);
        circleItem = new MenuItem("Circle");
        rectItem = new MenuItem("Rectangle");
        menu.add(circleItem);  menu.add(rectItem);
        circleItem.addActionListener(this);
        rectItem.addActionListener(this);
    }
```

// xem tiếp ở slide tiếp theo

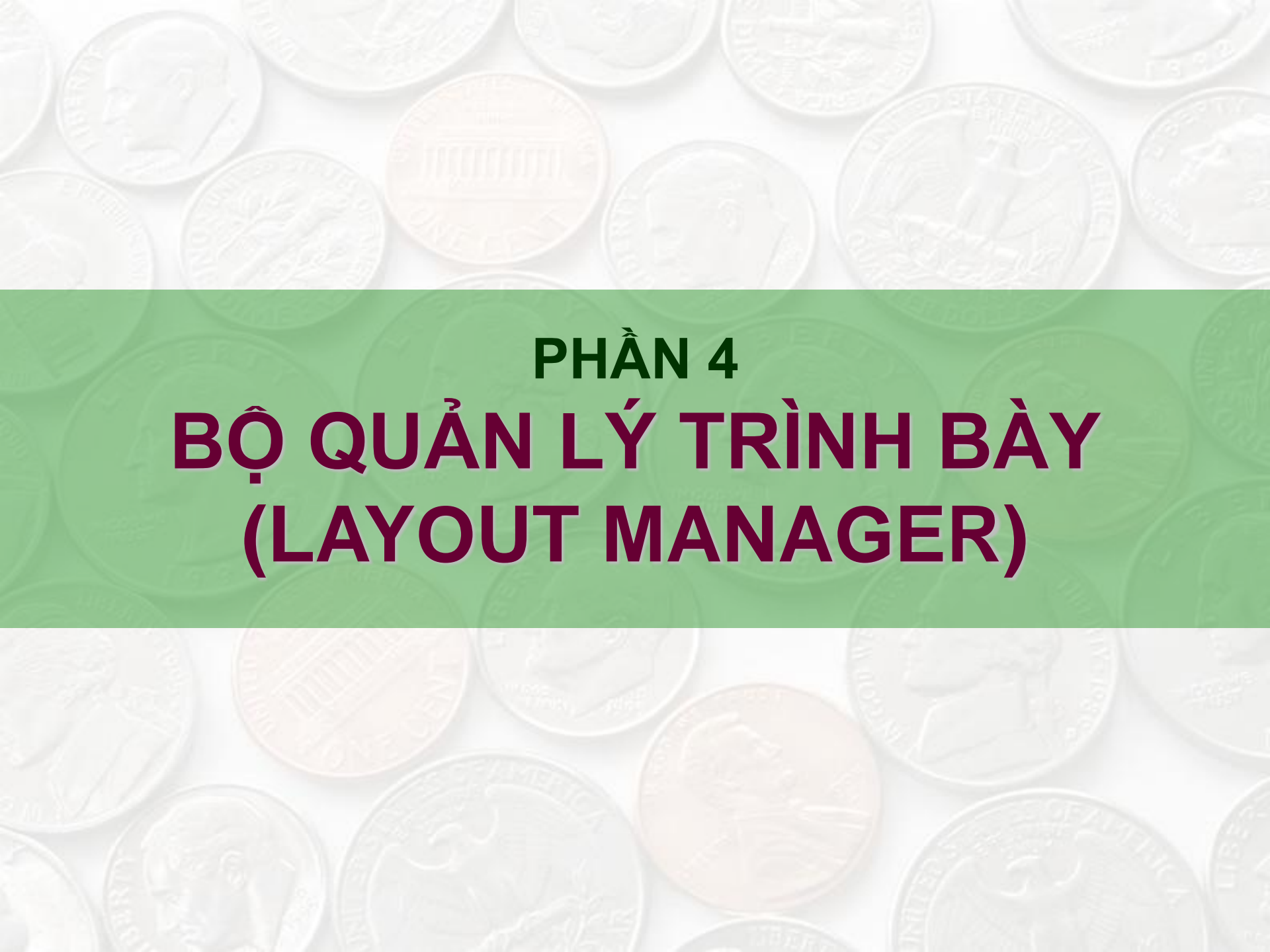
VÍ DỤ VỀ FRAME VÀ DIALOG

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == circleItem)
    {MyDialog dialog = new MyDialog(this, "Modeless Dialog: Circle", false);
    }
}
}

Class MyDialog extends Dialog
{
    MyDialog(Frame parent, String title, boolean isModel)
    {
        super(parent, title, isModel);
        add(newLabel("Hi, I am a dialog"), BorderLayout.CENTER);
        setSize(300, 200); setVisible(true);
        addWindowListener(newMyDialogListener(this));
    }
}
```


VÍ DỤ VỀ FRAME VÀ DIALOG

```
// Co the dat lop nay lam lop noi (inner class) cua lop MyDialog
class MyDialogListener extends WindowAdapter
{
    Dialog dialog;
    MyDialogListener(Dialog dia){dialog = dia;}
    public void window Closing(WindowEvent e)
    {
        dialog.setVisible(false);
        dialog.dispose();
    }
}
```

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

BỘ QUẢN LÝ TRÌNH BÀY (LAYOUT MANAGER)

BỘ QUẢN LÝ TRÌNH BÀY

Có năm bộ quản lý trình bày:

- Flow Layout
- Border Layout
- Grid Layout
- Gridbag Layout
- Null Layout

FLOW LAYOUT

Đối với một container trình bày theo kiểu FlowLayout thì:

- Các component gắn vào được sắp xếp theo thứ tự từ trái sang phải và từ trên xuống dưới.
- Các component có kích thước như mong muốn.
- Nếu chiều rộng của Container không đủ chỗ cho các component thì chúng tự động tạo ra một dòng mới.
- FlowLayout thường được dùng để sắp xếp các button trong 1 panel.
- Chúng ta có thể điều chỉnh khoảng cách giữa các component.

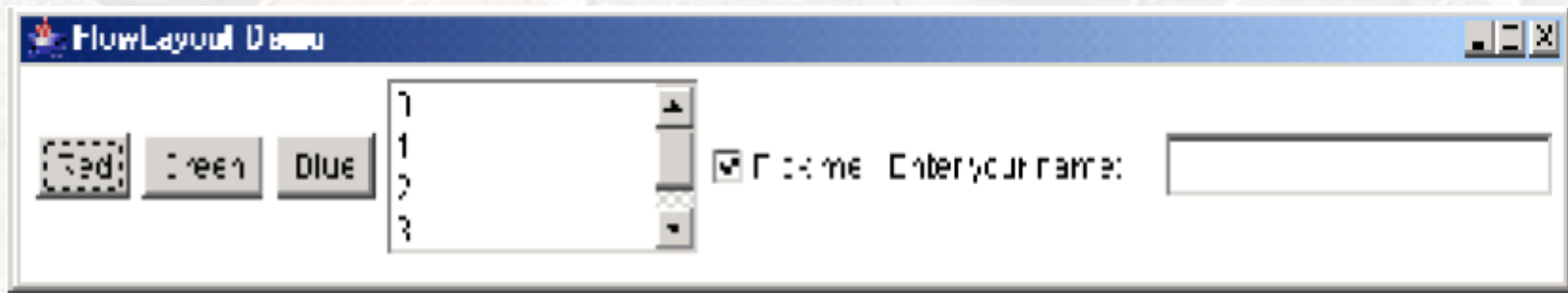
FLOW LAYOUT

Ví dụ:

```
import java.awt.*;  
import java.lang.Integer;  
class FlowLayoutDemo  
{  
    public static void main(String args[])  
    {  
        Frame fr = new Frame("FlowLayout Demo");  
        fr.setLayout(new FlowLayout());  
        fr.add(new Button("Red"));  
        fr.add(new Button("Green"));  
        fr.add(new Button("Blue"));  
        List li = new List();  
        for (int i=0; i<5; i++)  
        {  
            li.add(Integer.toString(i));  
        }  
    }  
}  
  
//xem tiếp ở slide tiếp theo
```

FLOW LAYOUT

```
fr.add(li);  
fr.add(new Checkbox("Pick me", true));  
fr.add(new Label("Enter your name:"));  
fr.add(new TextField(20));  
// phương thức pack() được gọi sẽ làm cho cửa sổ  
// hiện hành sẽ có kích thước vừa với kích thước  
// trình bày bố trí những thành phần con của nó.  
fr.pack();  
fr.setVisible(true);  
}  
}
```



BORDER LAYOUT

Đối với một container trình bày theo kiểu BorderLayout thì:

- Bộ trình bày khung chứa được chia làm 4 vùng: NORTH, SOUTH, WEST, EAST và CENTER. (Đông, Tây, Nam, Bắc và trung tâm). Bộ trình bày loại này cho phép sắp xếp và thay đổi kích thước của những components chứa trong nó sao cho vừa với 5 vùng ĐÔNG, TÂY, NAM, BẮC, TRUNG TÂM.
- Không cần phải gắn component vào cho tất cả các vùng.
- Các component ở vùng NORTH và SOUTH có chiều cao tùy ý nhưng có chiều rộng đúng bằng chiều rộng vùng chứa.
- Các component ở vùng EAST và WEST có chiều rộng tùy ý nhưng có chiều cao đúng bằng chiều cao vùng chứa.
- Các component ở vùng CENTER có chiều cao và chiều rộng phụ thuộc vào các vùng xung quanh.

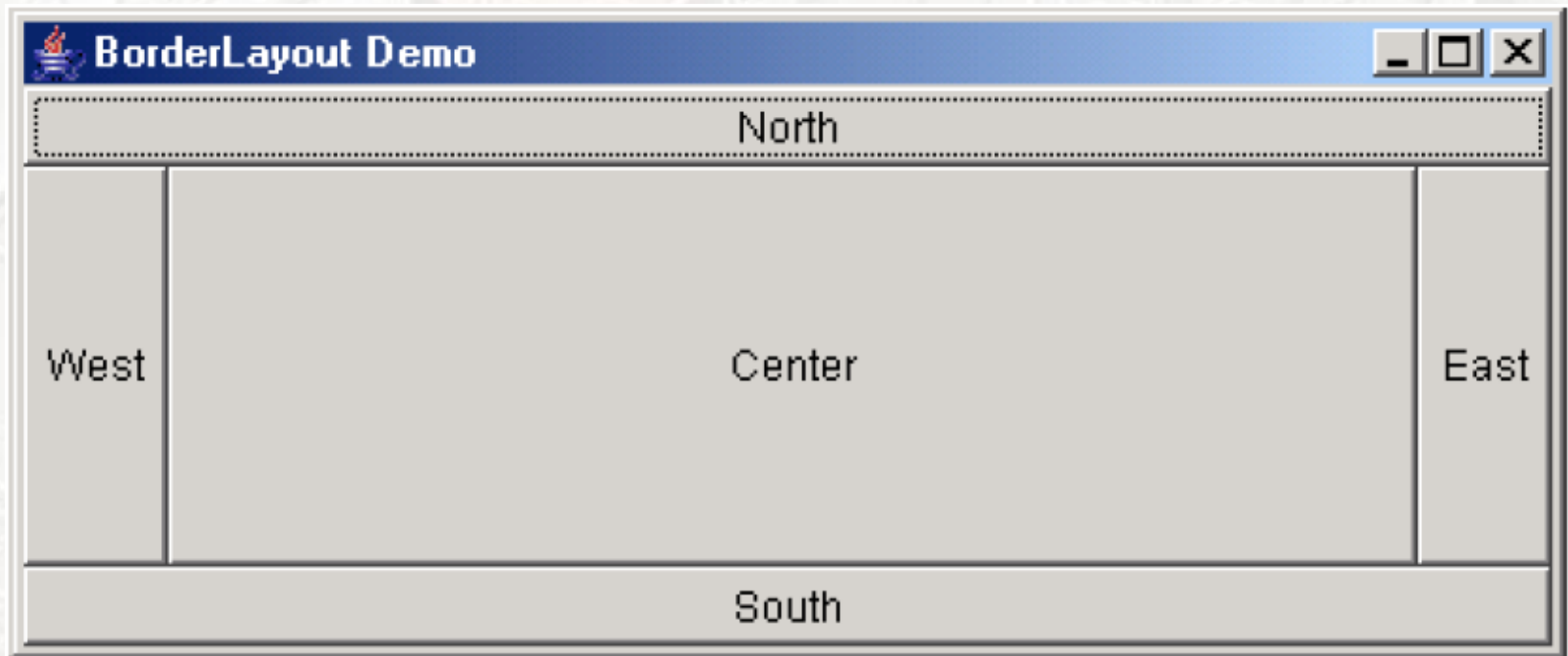
BORDER LAYOUT

Ví dụ:

```
import java.awt.*;

class BorderLayoutDemo extends Frame
{
    private Button north, south, east, west, center;
    public BorderLayoutDemo(String sTitle)
    {
        super(sTitle);
        north = new Button("North");
        south = new Button("South");
        east = new Button("East");
        west = new Button("West");
        center = new Button("Center");
        this.add(north, BorderLayout.NORTH);
        this.add(south, BorderLayout.SOUTH);
        this.add(east, BorderLayout.EAST);
        this.add(west, BorderLayout.WEST);
        this.add(center, BorderLayout.CENTER);
    }
    public static void main(String args[])
    {
        Frame fr = new BorderLayoutDemo ("Border Layout Demo");
        fr.pack();
        fr.setVisible(true);
    }
}
```

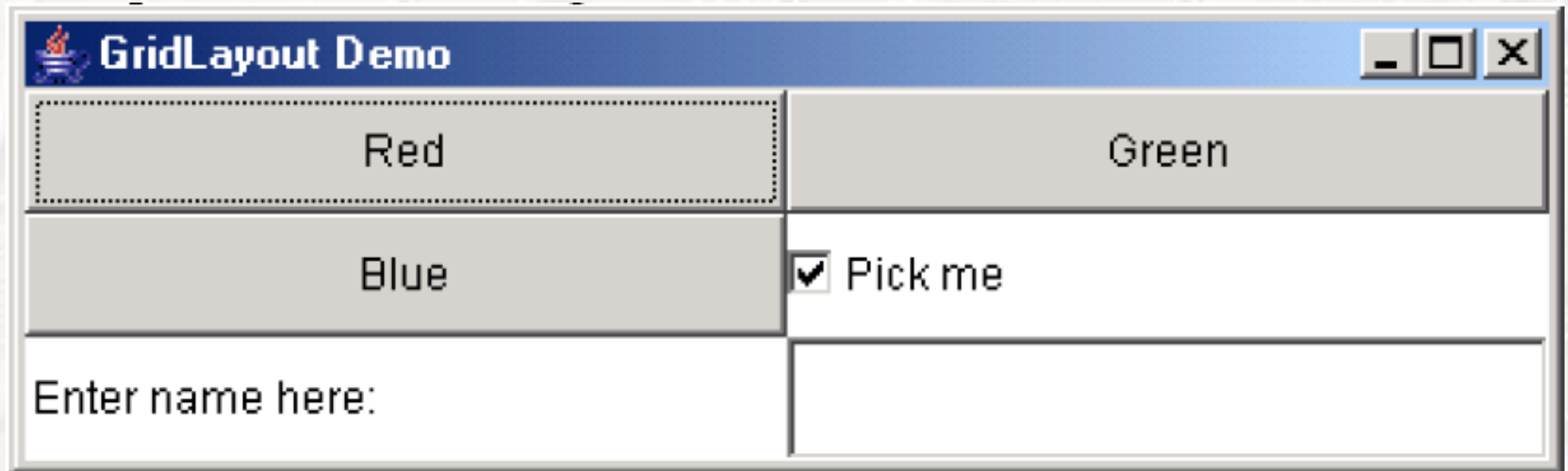

BORDER LAYOUT



GRID LAYOUT

Đối với một container trình bày theo kiểu GridLayout thì:

- Bộ trình bày tạo một khung lưới vô hình với các ô bằng nhau.
- Các đối tượng sẽ đặt vừa kích thước với từng ô đó. Thứ tự sắp xếp từ trái qua phải và từ trên xuống dưới.



GRID LAYOUT

Ví dụ:

```
import java.awt.*;  
public class GridLayoutDemo  
{  
    public static void main(String arg[])  
    {  
        Frame f = new Frame("GridLayout Demo");  
        f.setLayout(new GridLayout(3,2));  
        f.add(new Button("Red"));  
        f.add(new Button("Green"));  
        f.add(new Button("Blue"));  
        f.add(new Checkbox("Pick me", true));  
        f.add(new Label("Enter name here:"));  
        f.add(new TextField());  
        f.pack();  
        f.setVisible(true);  
    }  
}
```

GRIDBAG LAYOUT

Đối với một container trình bày theo kiểu GridBagLayout thì:

- Các componets khi được đưa vào khung chứa sẽ được trình bày trên 1 khung lưới vô hình tương tự như GridLayout. Tuy nhiên khác với GridLayout kích thước các đối tượng không nhất thiết phải vừa với 1 ô trên khung lưới mà có thể là 2, 3 ô hay nhiều hơn tùy theo các ràng buộc mà ta chỉ định thông qua đối tượng GridBagConstraints.

GRIDBAG LAYOUT

Lớp **GridBagConstraints** dẫn xuất từ lớp **Object**. Lớp **GridBagConstraints** dùng để chỉ định ràng buộc cho những components trình bày trong khung chứa container theo kiểu **GridLayout**.

- **gridx, gridy**: vị trí ô của khung lưới vô hình mà ta sẽ đưa đối tượng con vào o **gridwidth, gridheight**: kích thước hay vùng trình bày cho đối tượng con.
- **Insets**: là một biến đối tượng thuộc lớp **Inset** dùng để qui định khoảng cách biên phân cách theo 4 chiều (trên, dưới, trái, phải).
- **weightx, weighty**: chỉ định khoảng cách lớn ra tương đối của các đối tượng con với nhau

GRIDBAG LAYOUT

Ví dụ:

```
import java.awt.*;  
public class GridBagLayoutDemo  
{  
    public static void main(String arg[])  
    {  
        Frame f = new Frame("GridBagLayout Demo");  
        // Thiet lap layout manager  
        // Tao doi tuong rang buoc cho cach trinh bay  
        // GridBagLayout.  
        GridBagLayout layout = new GridBagLayout();  
        GridBagConstraints constraints = new  
        GridBagConstraints();  
        f.setLayout(layout);  
        // Tao ra 9 nut nhan  
        String[] buttName = {"Mot", "Hai", "Ba", "Bon", "Nam", "Sau", "Bay",  
                             "Tam", "Chin"};  
  
        Button[] buttons = new Button[9];  
        for(int i=0;i<9;i++)  
        {  
            buttons[i] = new Button (buttName[i]);  
        }  
  
        //xem tiếp ở slide tiếp theo
```

GRIDBAG LAYOUT

```
// Rang buoc cac nut nhan cach nhau 2 pixel
constraints.insets = new Insets(2,2,2,2);
// Qui dinh cac nut nhan se thay doi kich thuoc
// theo ca 2 chieu
constraints.fill = GridBagConstraints.BOTH;
// Rang buoc cho nut nhan thu 1
constraints.gridx = 1;           constraints.gridy = 1;
constraints.gridheight = 2;      constraints.gridwidth = 1;
layout.setConstraints(buttons[0], constraints);
// Rang buoc cho nut nhan thu 2
constraints.gridx = 2;           constraints.gridy = 1;
constraints.gridheight = 1;      constraints.gridwidth = 2;
layout.setConstraints(buttons[1], constraints);
// Rang buoc cho nut nhan thu 3
constraints.gridx = 2;           constraints.gridy = 2;
constraints.gridheight = 1;      constraints.gridwidth = 1;
layout.setConstraints(buttons[2], constraints);
// Rang buoc cho nut nhan thu 4
constraints.gridx = 1;           constraints.gridy = 3;
constraints.gridheight = 1;      constraints.gridwidth = 2;
layout.setConstraints(buttons[3], constraints);
```

//xem tiếp ở slide tiếp theo

GRIDBAG LAYOUT

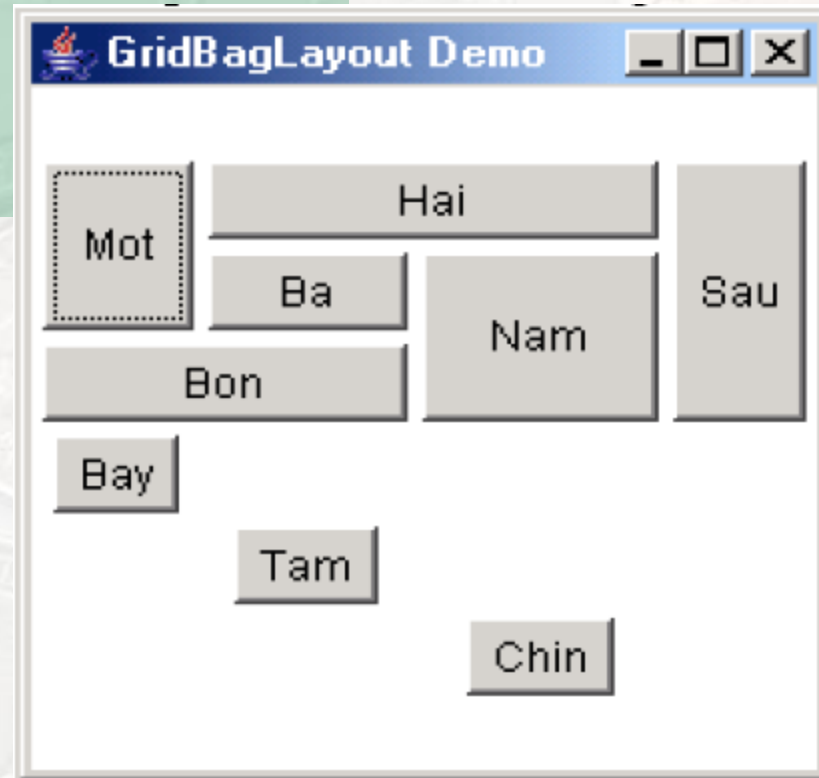
```
// Rang buoc cho nut nhan thu 5
constraints.gridx = 3;          constraints.gridy = 2;
constraints.gridheight = 2;    constraints.gridwidth = 1;
layout.setConstraints(buttons[4], constraints);
// Rang buoc cho nut nhan thu 6
constraints.gridx = 4;          constraints.gridy = 1;
constraints.gridheight = 3;    constraints.gridwidth = 1;
layout.setConstraints(buttons[5], constraints);
// Tu nut thu 7 tro di khong can rang buoc
// thay vi doi kich thuoc
constraints.fill = GridBagConstraints.NONE;
// Rang buoc cho nut nhan thu 7
constraints.gridx = 1;          constraints.gridy = 4;
constraints.gridheight = 1;    constraints.gridwidth = 1;
constraints.weightx = 1.0;
layout.setConstraints(buttons[6], constraints);
// Rang buoc cho nut nhan thu 8
constraints.gridx = 2;          constraints.gridy = 5;
constraints.gridheight = 1;    constraints.gridwidth = 1;
constraints.weightx = 2.0;
layout.setConstraints(buttons[7], constraints);
```

//xem tiếp ở slide tiếp theo

GRIDBAG LAYOUT

```
// Rang buoc cho nut nhan thu 9
constraints.gridx = 3;
constraints.gridy = 6;
constraints.gridheight = 1;
constraints.gridwidth = 1;
constraints.weightx = 3.0;
layout.setConstraints(buttons[8], constraints);
// Dua cac nut nhan khong chua chuong trinh
for (int i=0;i<9;i++)
    f.add(buttons[i]);

f.pack();
f.setVisible(true);
}
```



NULL LAYOUT

- Một khung chứa được trình bày theo kiểu Null Layout có nghĩa là người lập trình phải tự làm tất cả từ việc qui định kích thước của khung chứa, cũng như kích thước và vị trí của từng đối tượng component trong khung chứa.
- Để thiết lập cách trình bày là Null Layout cho một container ta chỉ việc gọi phương thức `setLayout(null)` với tham số là **null**.

NULL LAYOUT

Một số phương thức của lớp trừu tượng Component dùng để định vị và qui định kích thước của component khi đưa chúng vào khung chứa trình bày theo kiểu kiểu tự do:

- public void setLocation(Point p)
- public void setSize(Dimension p)
- public void setBounds(Rectangle r)

Ví dụ:

- MyButton.setSize(new Dimension(20, 10));
- MyButton.setLocation(new Point(10, 10));
- MyButton.setBounds(10, 10, 20, 10);

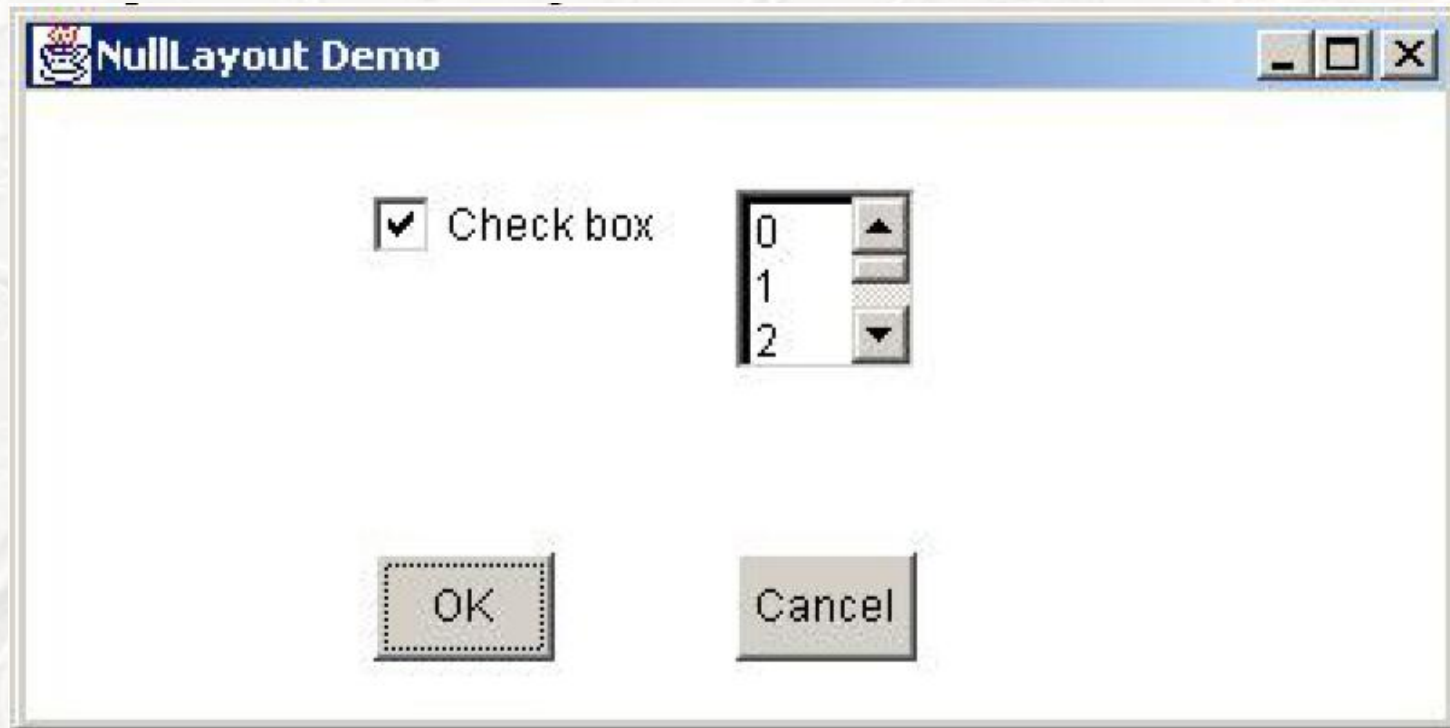
NULL LAYOUT

```
import java.awt.*;
class NullLayoutDemo
{
    public static void main(String args[])
    {
        Frame fr = new Frame("NullLayout Demo");
        fr.setLayout(null);
        Button buttOk = new Button("OK");
        buttOk.setBounds(100, 150, 50, 30);
        Button buttCancel = new Button("Cancel");
        buttCancel.setBounds(200, 150, 50, 30);
        Checkbox checkBut = new Checkbox("Check box", true);
        checkBut.setBounds(100, 50, 100, 20);
        List li = new List();
        for (int i=0; i<5; i++)
        {
            li.add(Integer.toString(i));
        }
        li.setBounds(200, 50, 50, 50);
        fr.add(buttOk);
        fr.add(buttCancel);
    }
}
```

//xem tiếp ở slide tiếp theo

NULL LAYOUT

```
fr.add(checkBut);  
fr.add(li);  
fr.setBounds(10, 10, 400, 200);  
fr.setVisible(true);  
}  
}
```



The background of the slide is a dense, overlapping pattern of various US coins, including pennies, nickels, and quarters, in shades of copper, silver, and gold. A semi-transparent purple rectangular band is centered horizontally across the image, serving as a backdrop for the title text.

HẾT **CHƯƠNG 4**