

BÀI 8. XÂY DỰNG GIAO DIỆN CHƯƠNG TRÌNH

1

Xây dựng giao diện chương trình

- Giới thiệu các gói lập trình giao diện trong Java: Java AWT, Java Swing
- Lập trình giao diện cơ bản với Java AWT
- Lập trình giao diện cơ bản với Java Swing

2

1. TỔNG QUAN

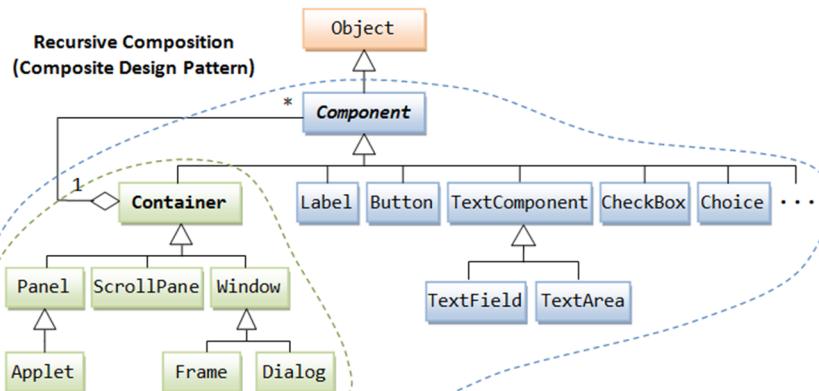
3

Các gói lập trình giao diện trong Java

- Java cung cấp các gói lập trình giao diện chính sau:
 - java.awt: cung cấp các lớp cơ bản để lập trình giao diện
 - javax.swing: cung cấp các lớp mới để xây dựng giao diện chương trình dễ dàng, mềm dẻo hơn
 - java.swt: được phát triển bởi IBM
- Giao diện chương trình gồm cửa sổ và các thành phần điều khiển (nút bấm, ô nhập dữ liệu...) đặt lên trên
- Có thể sử dụng plug-in để hỗ trợ:
 - Eclipse: WindowsBuilder Pro
 - Netbean: đã tích hợp

4

Java AWT



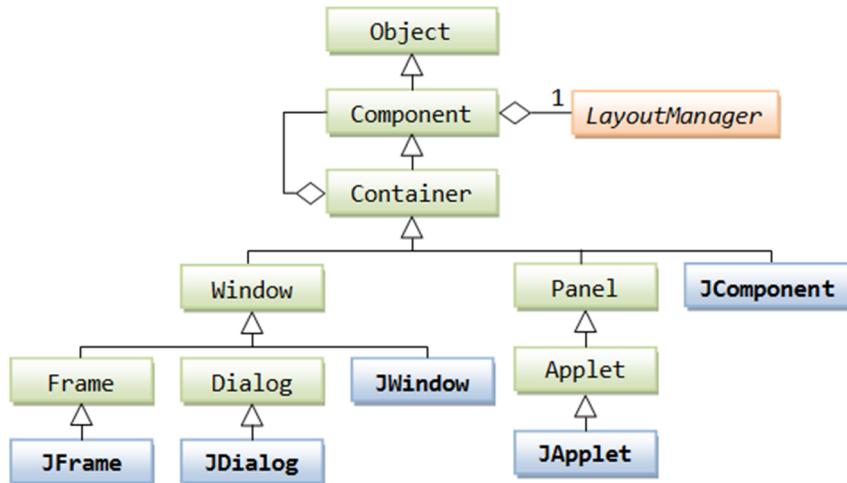
5

Các thành phần cơ bản trong Java AWT

- Component: một thành phần có thể hiển thị trên màn hình đồ họa
- Container: lớp chứa, bao chứa các thành phần khác
 - Một đối tượng Container có thể chứa các đối tượng Container khác
- Label: Nhãn
- Button: nút bấm
- Checkbox: ô lựa chọn
- TextComponent: nhập xuất dữ liệu dạng text
- ...

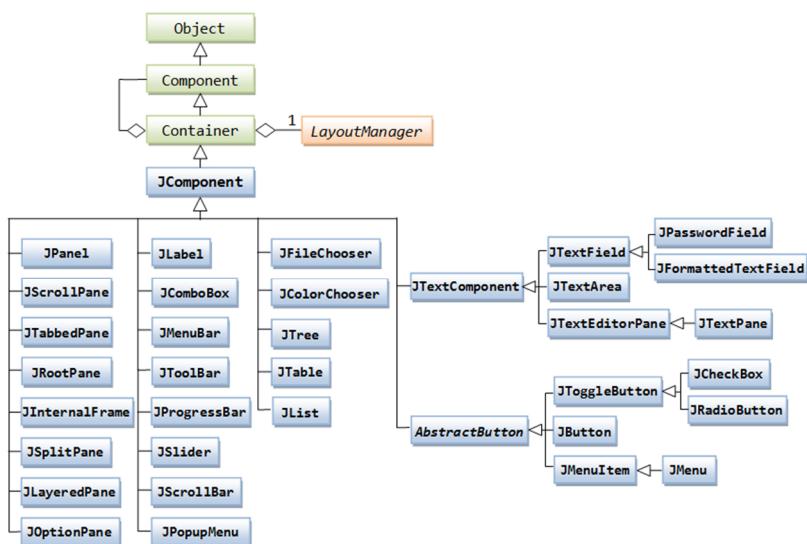
6

Java Swing



7

Java Swing (tiếp)



8

2. XÂY DỰNG GIAO DIỆN VỚI JAVA AWT

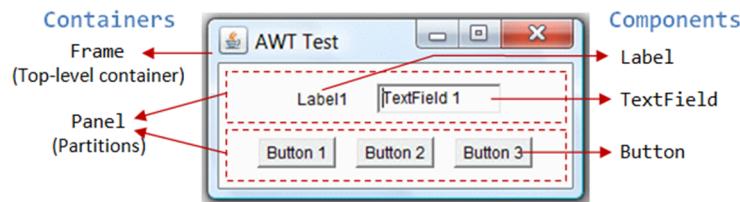
9

Các gói trong Java AWT

- Java AWT có 12 gói cung cấp các lớp để xây dựng giao diện đồ họa (GUI)
- 2 gói được sử dụng thường xuyên
- `java.awt` gồm các lớp GUI cơ bản
 - Các lớp Component (như Button, TextComponent, Label)
 - Các lớp Container – lớp chứa (Frame, Panel, Dialog, ScrollPanel)
 - Các lớp quản lý layout(FlowLayout, BorderLayout, GridLayout)
 - Các lớp đồ họa tùy biến(Graphics, Color, Font)
- `java.awt.event` gồm các lớp xử lý sự kiện trên giao diện:
 - Các lớp sự kiện (ActionEvent, MouseEvent, KeyEvent và WindowEvent)
 - Các giao diện nghe sự kiện(MouseListener, KeyListener...)
 - Các lớp Adapter (MouseAdapter, KeyAdapter)

10

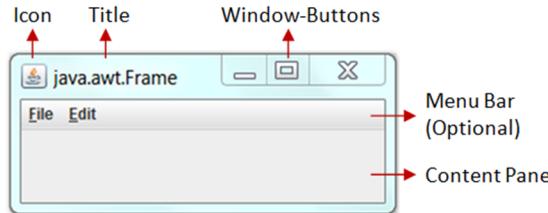
Container và Component



- Component: là đối tượng cơ bản tạo nên giao diện
 - Cho phép người dùng tương tác với chương trình
- Container: là đối tượng chứa các component
 - Bản thân container cũng là một component
 - Một container có thể chứa các container khác
 - Phương thức `add(Component)`: thêm một component vào container

11

Top-Level Containers: Frame, Dialog và Applet



- Frame: cửa sổ chính của giao diện chương trình
 - Xây dựng cửa sổ chương trình bằng cách kế thừa lớp Frame hoặc kết hợp 1 đối tượng Frame
 - Kế thừa để sử dụng ngay các thành viên của Frame
- Dialog: cửa sổ pop-up được sử dụng để tạo ra các tương tác nằm ngoài cửa sổ chính
- Applet: sử dụng trên xây dựng chương trình chạy trên trình duyệt Web

12

Frame – Ví dụ

```

import java.awt.Frame

// Một chương trình với giao diện đồ họa kế thừa lớp Frame
public class MyGUIProgram extends Frame {
    // Phương thức khởi tạo. Các đối tượng component cũng
    // được tạo ra tại đây
    public MyGUIProgram() { ..... }

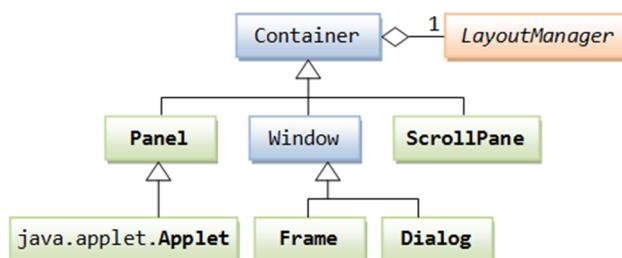
    // Các phương thức khác

    // Phương thức main
    public static void main(String[] args) {
        // Gọi hàm khởi tạo
        new MyGUIProgram();
    }
}

```

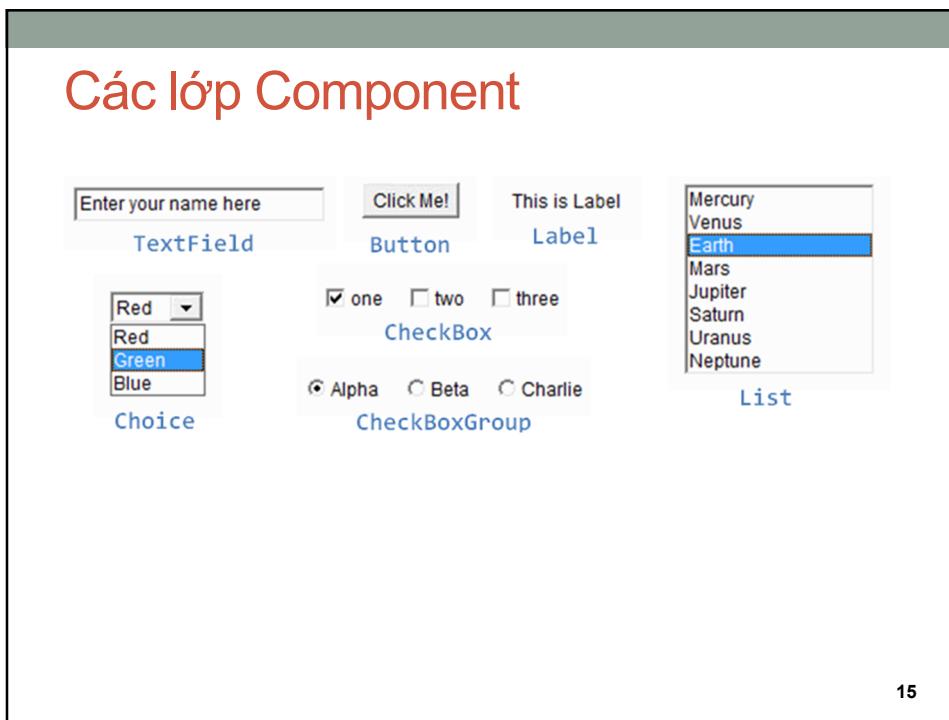
13

Secondary Containers: Panel và ScrollPane

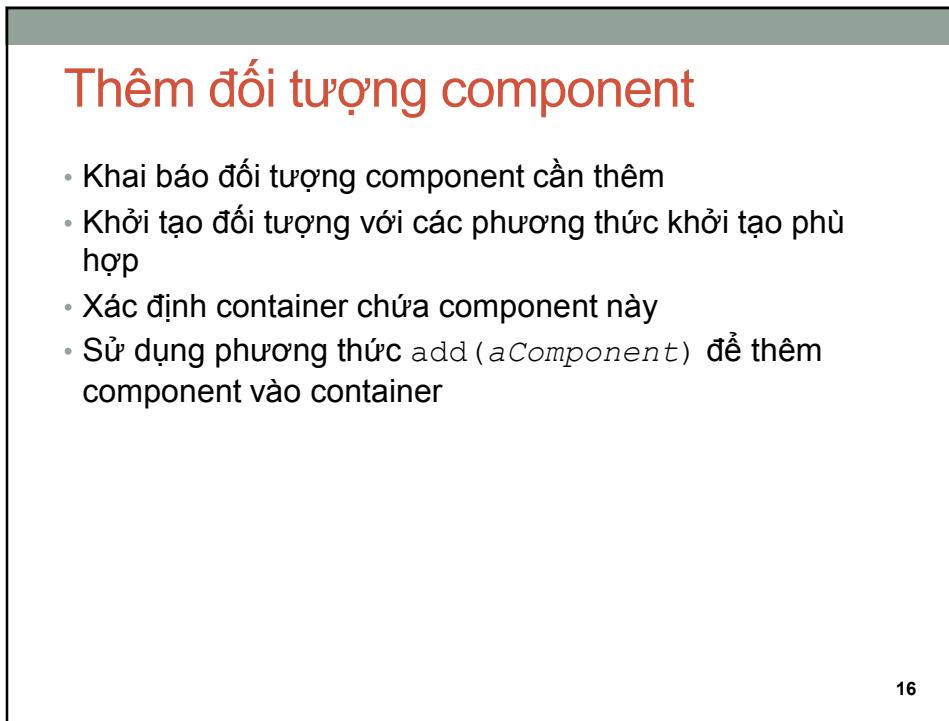


- Panel: khung chữ nhật nằm trong một top-level container, được sử dụng để tạo layout cho chương trình
- ScrollPane: tạo ra hiệu ứng cuộn chuột (ngang/dọc) cho một component

14



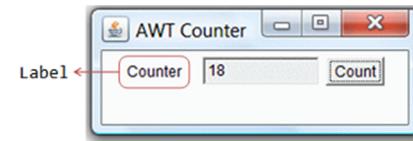
15



16

java.awt.Label

- Hiển thị một nội dung nào đó dưới dạng văn bản



- Hiển thị một nội dung nào đó dưới dạng văn bản

- Phương thức khởi tạo

```
public Label(String strLabel, int alignment);
public Label(String strLabel);
public Label();
```

- Một số phương thức:

```
public String getText();
public void setText(String strLabel);
public int getAlignment();
public void setAlignment(int alignment);
```

17

java.awt.Label - ví dụ

```
Label lblInput;
lblInput = new Label("Enter ID");
add(lblInput);
lblInput.setText("Enter password");
lblInput.getText();

/* nhãn ẩn danh (anonymous), không thẻ tương tác
add(new Label("Enter Name: ", Label.RIGHT));
```

18

java.awt.Button

- Tạo ra một hành động nào đó của chương trình qua sự kiện nhấp chuột

- Phương thức khởi tạo

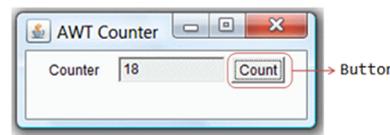
```
public Button(String buttonLabel);
public Button();
```

- Một số phương thức:

```
public String getLabel();
public void setLabel(String buttonLabel);
public void setEnable(boolean enablevt);
```

- Ví dụ:

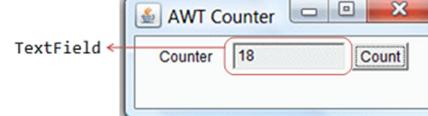
```
Button btnColor = new Button("Red");
add(btnColor);
btnColor.setLabel("green");
btnColor.getLabel();
add(new Button("Blue")); // anonymous
Button
```



19

java.awt.TextField

- Sử dụng để nhập/xuất một dòng văn bản



- Sử dụng để nhập/xuất một dòng văn bản

- Khi đang ở trong TextField, nhấp phím Enter có thể kích hoạt một hành động nào đó của chương trình

- Phương thức khởi tạo:

```
public TextField(String strInitText, int columns);
public TextField(String strInitText);
public TextField(int columns);
```

- Một số phương thức: `getText()`, `setText(String strText)`, `setEditable(boolean editablevt)`

20

java.awt.TextField – Ví dụ

```
TextField tfInput = new TextField(30);
add(tfInput);
TextField tfResult = new TextField();
tfResult.setEditable(false) ; // Set to read-only
add(tfResult);
//do something
//...
int number = Integer.parseInt(tfInput.getText());
number *= number;
tfResult.setText(number + "");
```

21

Một chương trình đơn giản với giao diện

```
import java.awt.*;
import java.awt.event.*;
/** The Countdown class illustrating a countdown allows the
 * user enter a positive value and press the button until
 * the value is zero
public class Countdown extends Frame implements
                                ActionListener {
    private Label lblCount;
    private TextField tfCount;
    private Button btnCount;
    /** Constructor to setup GUI components and event
     handling */
    public Countdown () {
        setLayout(new FlowLayout());
```

22

Countdown (tiếp)

```

lblCount = new Label("Counter");
add(lblCount);
tfCount = new TextField(10);
add(tfCount);
btnCount = new Button("Countdown");
add(btnCount);
btnCount.addActionListener(this);
setTitle("Countdown");
setSize(250, 100);
setLocationRelativeTo(null); //appear at center
setVisible(true);
}
public static void main(String[] args) {
    Countdown app = new Countdown();
}

```

23

Countdown (tiếp)

```

/** ActionEvent handler - Called back upon button-click.
 */
@Override
public void actionPerformed(ActionEvent evt) {
    int count = 0;
    count = Integer.parseInt(tfCount.getText());

    if (tfCount.isEditable())
        tfCount.setEditable(false);
    count--;
    if(count < 1)
        btnCount.setEnabled(false);
    tfCount.setText(count + "");
}
}

```

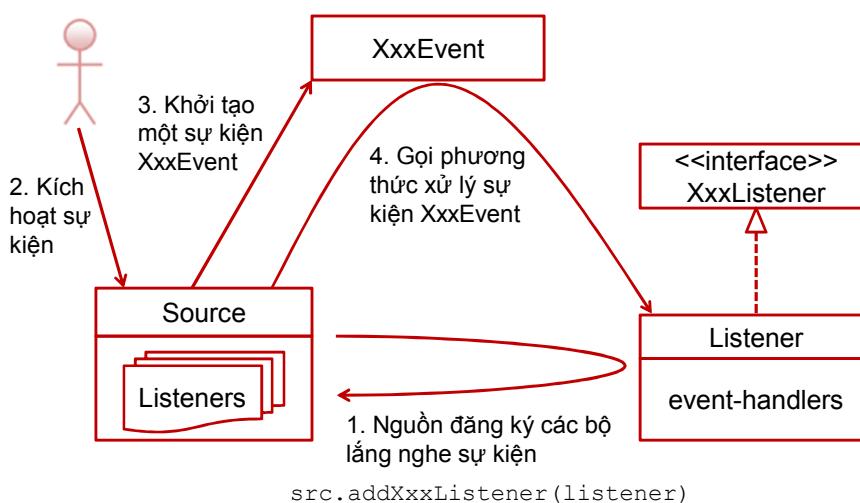
24

Nghe và xử lý sự kiện trên giao diện

- Người dùng tương tác với chương trình qua giao diện
- Chương trình phải nghe được các sự kiện trên giao diện (nhập dữ liệu, nhấn phím Enter, nhấp chuột, đóng cửa sổ chương trình...) để thực hiện hành động tương ứng → lập trình hướng sự kiện
- Tham gia sự kiện luôn có 3 đối tượng: nguồn (source) sinh sự kiện, bộ nghe sự kiện (listener), và sự kiện (event)
- Nguồn (source): là nơi phát sinh sự kiện(button, textfield...)
- Mỗi nguồn sẽ đăng ký các bộ nghe sự kiện khác nhau
- Khi có sự kiện nào đó xảy ra từ nguồn, phương thức xử lý sự kiện (event handler) trên bộ nghe sự kiện sẽ được gọi để xử lý

25

Nghe và xử lý sự kiện



26

Một số Listener

```
public interface WindowListener{  
    public void windowClosing(WindowEvent evt);  
    public void windowOpened(WindowEvent evt);  
    public void windowClosed(WindowEvent evt);  
    public void windowActivated(WindowEvent evt);  
    public void windowDeactivated(WindowEvent evt);  
    public void windowIconified(WindowEvent evt);  
    public void windowDeiconified(WindowEvent evt);  
}  
  
public interface MouseListener {  
    public void mousePressed(MouseEvent evt);  
    public void mouseReleased(MouseEvent evt);  
    public void mouseClicked(MouseEvent evt);  
    public void mouseEntered(MouseEvent evt);  
    public void mouseExited(MouseEvent evt);  
}
```

27

Một số Listener(tiếp)

```
public interface KeyListener {  
    public void keyPressed(KeyEvent evt);  
    public void keyReleased(KeyEvent evt);  
    public void keyTyped(KeyEvent evt);  
}
```

28

Nghe và xử lý sự kiện – Ví dụ

```

import java.awt.*;
import java.awt.event.*;
/** The Countdown class illustrating a countdown allows the
/* user enter a positive value and press the button until
*/ the value is zero
public class Countdown extends Frame implements
    ActionListener, WindowListener {
    private Label lblCount;
    private TextField tfCount;
    private Button btnCount;
    /** Constructor to setup GUI components and event
        handling */
    public Countdown () {
        setLayout(new FlowLayout());

```

29

Nghe và xử lý sự kiện – Ví dụ

```

lblCount = new Label("Counter");
add(lblCount);
tfCount = new TextField(10);
add(tfCount);
btnCount = new Button("Countdown");
add(btnCount);
btnCount.addActionListener(this);
addWindowListener(this);
setTitle("Countdown");
setSize(250, 100);
setLocationRelativeTo(null); //appear at center
setVisible(true);
}
public static void main(String[] args) {
    Countdown app = new Countdown();
}

```

30

Nghe và xử lý sự kiện – Ví dụ

```
/** ActionEvent handler - Called back upon button-click.
 */
@Override
public void actionPerformed(ActionEvent evt) {
    int count = 0;
    count = Integer.parseInt(tfCount.getText());

    if (tfCount.isEditable())
        tfCount.setEditable(false);
    count--;
    if(count < 1)
        btnCount.setEnabled(false);
    tfCount.setText(count + "");
}
```

31

Nghe và xử lý sự kiện – Ví dụ

```
/** WindowEvent handler - Called back upon clicking close-/*
window button*/
@Override
public void windowClosing(WindowEvent e) {
    System.exit(0); // Terminate the program
}
// Not used, but need to provide an empty body
@Override
public void windowOpened(WindowEvent e) { }
@Override
public void windowClosed(WindowEvent e) { }
@Override
public void windowIconified(WindowEvent e) { }
@Override
public void windowDeiconified(WindowEvent e) { }
@Override
public void windowActivated(WindowEvent e) { }
@Override
public void windowDeactivated(WindowEvent e) { }
}
```

32

Ví dụ - Xử lý sự kiện chuột

```

import java.awt.*;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.MouseMotionListener;
/** The MouseEventDemo listens the moving of mouse pointer
 * and invokes the handler to display the its coordinates
public class MouseEventDemo extends Frame implements
    MouseListener, MouseMotionListener, WindowListener{
    private TextField tfMouseClickX;
    private TextField tfMouseClickY;
    private TextField tfMousePositionX;
    private TextField tfMousePositionY;

```

33

Ví dụ - Xử lý sự kiện chuột(tiếp)

```

/** Constructor to setup the GUI */
public MouseEventCatcher() {
    setLayout(new FlowLayout());

    add(new Label("X-Click: "));
    tfMouseClickX = new TextField(10);
    tfMouseClickX.setEditable(false);
    add(tfMouseClickX);
    add(new Label("Y-Click: "));
    tfMouseClickY = new TextField(10);
    tfMouseClickY.setEditable(false);
    add(tfMouseClickY);

```

34

Ví dụ - Xử lý sự kiện chuột(tiếp)

```

add(new Label("X-Position: "));
tfMousePositionX = new TextField(10);
tfMousePositionX.setEditable(false);
add(tfMousePositionX);
add(new Label("Y-Position: "));
tfMousePositionY = new TextField(10);
tfMousePositionY.setEditable(false);
add(tfMousePositionY);
addMouseListener(this);
addMouseMotionListener(this);
addWindowListener(this);
setTitle("MouseEvent Demo");
setSize(400, 120);
setLocationRelativeTo(null);
setVisible(true);
}

```

35

Ví dụ - Xử lý sự kiện chuột(tiếp)

```

/** WindowEvent handler - Called back upon clicking close-*/
window button*/
@Override
public void windowClosing(WindowEvent e) {
    System.exit(0); // Terminate the program
}
// Not used, but need to provide an empty body
@Override
public void windowOpened(WindowEvent e) { }
@Override
public void windowClosed(WindowEvent e) { }
@Override
public void windowIconified(WindowEvent e) { }
@Override
public void windowDeiconified(WindowEvent e) { }
@Override
public void windowActivated(WindowEvent e) { }
@Override
public void windowDeactivated(WindowEvent e) { }
}

```

36

Ví dụ - Xử lý sự kiện chuột(tiếp)

```
/** MouseListener handlers */
// Called back when a mouse-button has been clicked
@Override
public void mouseClicked(MouseEvent e) {
    tfMouseClickX.setText(e.getX() + "");
    tfMouseClickY.setText(e.getY() + "");
}
@Override
public void mousePressed(MouseEvent e) { }
@Override
public void mouseReleased(MouseEvent e) { }
@Override
public void mouseEntered(MouseEvent e) { }
@Override
public void mouseExited(MouseEvent e) { }
```

37

Ví dụ - Xử lý sự kiện chuột(tiếp)

```
/** MouseMotionEvent handlers */
// Called back when the mouse-pointer has been moved
@Override
public void mouseMoved(MouseEvent e) {
    tfMousePositionX.setText(e.getX() + "");
    tfMousePositionY.setText(e.getY() + "");
}
@Override
public void mouseDragged(MouseEvent e) { }
```

```
public class MouseEventTest {
    public static void main(String[] args) {
        new MouseMotionDemo();
    }
}
```

38

Nhận xét các ví dụ

- Khi phải xử lý nhiều sự kiện, khai báo Frame mới cồng kềnh


```
public class MouseEventDemo extends Frame
    implements MouseListener, MouseMotionListener,
    WindowListener{
```
- Khó tái sử dụng được các phương thức xử lý sự kiện cho các sự kiện khác nhau xảy ra trên các nguồn khác nhau
→ Tách xử lý sự kiện ra khỏi sự kiện trên giao diện thành các lớp khác nhau

39

Class lồng (Nested class/Inner class)

- Là một class được khai báo trong class khác

```
public class MyOuterClass {
    .....
    private class MyNestedClass1 { ... }
    public static class MyNestedClass2 { ... }
    .....
}
```

- Là một class được khai báo trong class khác
- Có thể truy cập tới mọi thành viên của class bao nó
- Mang đầy đủ các đặc điểm của class thông thường

40

Class lồng - Ví dụ

```

import java.awt.*;
import java.awt.event.*;
/** The Countdown class illustrating a countdown allows the
/* user enter a positive value and press the button until
*/ the value is zero
public class Countdown extends Frame {
    private Label lblCount;
    private TextField tfCount;
    private Button btnCount;
    /** Constructor to setup GUI components and event
        handling */
    public Countdown () {
        setLayout(new FlowLayout());

```

41

Countdown (tiếp)

```

lblCount = new Label("Counter");
add(lblCount);
tfCount = new TextField(10);
add(tfCount);
btnCount = new Button("Countdown");
add(btnCount);
btnCount.addActionListener(new BtnCountListener());
setTitle("Countdown");
setSize(250, 100);
setLocationRelativeTo(null); //appear at center
setVisible(true);
}
public static void main(String[] args) {
    Countdown app = new Countdown();
}

```

42

Countdown (tiếp)

```
/** BtnCountListener is a "named inner class" used as
 * ActionListener. This inner class can access private
 * variables of the outer class.*/
private class BtnCountListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        int count = 0;
        count = Integer.parseInt(tfCount.getText());
        if (tfCount.isEditable())
            tfCount.setEditable(false);
        count--;
        if(count < 1)
            btnCount.setEnabled(false);
        tfCount.setText(count + "");
    }
}
```

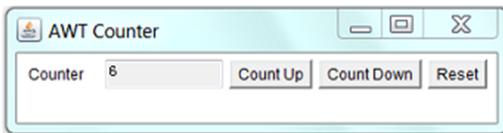
43

Lớp lồng ẩn danh(anonymous)

```
btnCount.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        int count = 0;
        count = Integer.parseInt(tfCount.getText());
        if (tfCount.isEditable())
            tfCount.setEditable(false);
        count--;
        if(count < 1)
            btnCount.setEnabled(false);
        tfCount.setText(count + "");
    }
});
//...
```

44

Sử dụng lớp lồng ẩn danh



```
import java.awt.*;
import java.awt.event.*;
public class AWTCounter3Buttons extends Frame {
    private TextField tfCount;
    private int count = 0;
    /** Constructor to setup the GUI */
    public AWTCounter3Buttons () {
        setLayout(new FlowLayout());
        add(new Label("Counter"));
        tfCount = new TextField("0", 10);
        tfCount.setEditable(false);
        add(tfCount);
```

45

Sử dụng lớp lồng ẩn danh(tiếp)

```
Button btnCountUp = new Button("Count Up");
add(btnCountUp);
btnCountUp.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        ++count;
        tfCount.setText(count + "");
    }
});
Button btnCountDown = new Button("Count Down");
add(btnCountDown);
btnCountDown.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        count--;
        tfCount.setText(count + "");
    }
});
```

46

Sử dụng lớp lồng ẩn danh(tiếp)

```
Button btnReset = new Button("Reset");
add(btnReset);
btnReset.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        count = 0;
        tfCount.setText("0");
    }
});

setTitle("AWT Counter");
setSize(400, 100);
setVisible(true);
}
```

47

Sử dụng lớp lồng xử lý cùng loại sự kiện trên các nguồn khác nhau

```
BtnListener listener = new BtnListener();
btnCountUp.addActionListener(listener);
btnCountDown.addActionListener(listener);
btnReset.addActionListener(listener);
//...
private class BtnListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        String btnLabel = e.getActionCommand();
        if (btnLabel.equals("Count Up")) {
            ++count;
        } else if (btnLabel.equals("Count Down")) {
            --count;
        } else {
            count = 0;
        }
        tfCount.setText(count + "");
    }
}
```

48

Lớp Adapter

- Trong trường hợp chỉ cần định nghĩa một số phương thức bắt sự kiện trong số các phương thức mà giao diện Listener yêu cầu triển khai, có thể sử dụng lớp Adapter
- Lớp Adapter định nghĩa sẵn các phương thức mà giao diện Interface yêu cầu triển khai với nội dung rỗng
- Khi sử dụng lớp Adapter, chỉ cần định nghĩa đè lên các phương thức cần dùng

49

Lớp Adapter - Ví dụ

- Không dùng lớp Adapter

```

@Override
public void windowClosing(WindowEvent e) {
    System.exit(0); // Terminate the program
}
// Not used, but need to provide an empty body
@Override
public void windowOpened(WindowEvent e) { }
@Override
public void windowClosed(WindowEvent e) { }
@Override
public void windowIconified(WindowEvent e) { }
@Override
public void windowDeiconified(WindowEvent e) { }
@Override
public void windowActivated(WindowEvent e) { }
@Override
public void windowDeactivated(WindowEvent e) { }
}

```

50

Lớp Adapter – Ví dụ

- Sử dụng lớp Adapter

```
addWindowListener(new WindowAdapter() {  
    @Override  
    public void windowClosing(WindowEvent e) {  
        System.exit(0); // Terminate the program  
    }  
});
```

51

Quản lý Layout và Panel

- Layout: các thức sắp xếp các phần tử (component) trên cửa sổ. Trên một cửa sổ Frame chỉ được chọn 1 layout
- Khi cần sử dụng nhiều layout khác nhau trên cửa sổ, cần sử dụng đối tượng lớp Panel
- Panel là một lớp chứa thứ cấp (secondary)
- Các layout cung cấp trong Java AWT
 - FlowLayout
 - GridLayout
 - BorderLayout
 - BoxLayout

52

FlowLayout

- Các phần tử được sắp xếp từ trái qua phải theo thứ tự trong mã nguồn
- Khi hết chiều ngang trên một hàng, các phần tử tiếp theo tự động xuống theo chiều ngang
- Phương thức khởi tạo:


```
public FlowLayout();
public FlowLayout(int align);
public FlowLayout(int align, int hgap, int vgap);
```
- align (canh lề) FlowLayout.LEFT (or LEADING), FlowLayout.RIGHT (or.TRAILING), or FlowLayout.CENTER
- hgap, vgap: khoảng trống giữa các phần tử
- Mặc định: hgap=5, vgap=5, align=CENTER

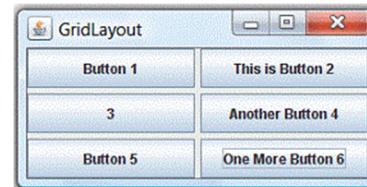


53

GridLayout

- Hiển thị theo dạng lưới, từ trái sang, từ trên xuống
- Hiển thị theo dạng lưới, từ trái sang, từ trên xuống
- Phương thức khởi tạo:

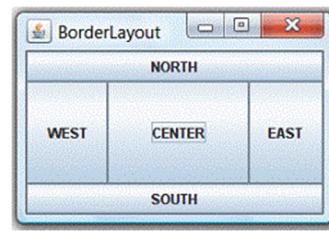

```
public GridLayout(int rows, int cols);
public GridLayout(int rows, int cols, int hgap, int vgap);
```
- Mặc định: rows = 1, cols = 0, hgap = 0, vgap = 0
- Khi một trong hai giá trị rows hoặc cols bằng 0, các thành phần được sắp xếp theo giá trị còn lại
- Khi cả hai giá trị khác 0, giá trị cols sẽ bị bỏ qua



54

BorderLayout

- Container chia thành 5 vùng: EAST, WEST, SOUTH, NORTH, và CENTER



- Phương thức khởi tạo:

```
public BorderLayout();
public BorderLayout(int hgap, int vgap);
```

- Thêm một phần tử:

```
add(aComponent, aZone)
```

- Ví dụ:

```
btnNorth = new Button("NORTH");
add(btnNorth, BorderLayout.NORTH);
```

55

Sử dụng Panel thiết kế giao diện

- Trên Frame có thể thêm nhiều Panel
- Các Panel có thể sử dụng layout khác nhau
- Phương thức khởi tạo:

```
new Panel(Layout)
```

- Thêm phần tử vào Panel: phương thức add(Component)

```
Panel panelDisplay = new Panel(new FlowLayout());
Panel panelButtons = new Panel(new GridLayout(4, 3));
setLayout(new BorderLayout());
add(panelDisplay, BorderLayout.NORTH);
add(panelButtons, BorderLayout.CENTER);
tfDisplay = new TextField("0", 20);
panelDisplay.add(tfDisplay);
```

56

3. XÂY DỰNG GIAO DIỆN VỚI JAVA SWING

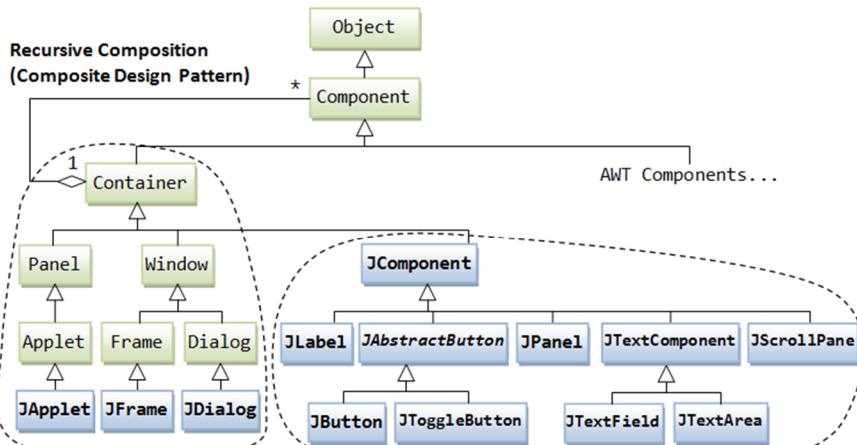
57

Các gói trong Java Swing

- Swing cung cấp 18 gói có thể sử dụng xây dựng giao diện đồ họa
 - Thường sử dụng lệnh import javax.swing.* để chương trình trở nên ngắn gọn
- **Ưu điểm của Swing so với AWT:**
 - Cung cấp thêm các đối tượng mới để xây dựng giao diện đồ họa
 - *look-and-feel*: tùy biến để các thành phần giao diện của Swing nhìn giống như các thành phần giao diện của HĐH
 - Hỗ trợ các thao tác sử dụng bàn phím thay chuột
 - Sử dụng tài nguyên hiệu quả hơn
 -

58

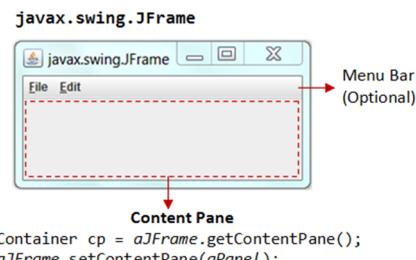
Các phần tử của Swing



59

Swing Container

- Top-level container:
 - JFrame: sử dụng cho các cửa sổ chính của chương trình
 - JApplet: sử dụng trên trình duyệt
 - JDialog: cửa sổ thông báo
- Secondary container: JPanel
- Thêm các đối tượng vào cửa sổ JFrame:
 - Không thể thêm trực tiếp
 - Phải tương tác qua Content Pane của JFrame
 - Đối tượng JFrame cung cấp 2 phương thức
 - getContentPane() trả lại một đối tượng ContentPane thuộc lớp Container
 - setContentPane(JPanel): thiết lập nội dung cho Content Pane



```
Container cp = aJFrame.getContentPane();
aJFrame.setContentPane(aPanel);
```

60

getContentPane() - Ví dụ

```
public class TestGetContentPane extends JFrame {
    // Constructor
    public TestGetContentPane() {
        // Get the content-pane of this JFrame, which
        // is a java.awt.Container
        // All operations, such as setLayout() and
        // add() operate on the content-pane
        Container cp = this.getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(new JLabel("Hello, world!"));
        cp.add(new JButton("Button"));
        .....
    }
    .....
}
```

61

setContentPane() - Ví dụ

```
public class TestSetContentPane extends JFrame {
    // Constructor
    public TestSetContentPane() {
        // The "main" JPanel holds all the GUI components
        JPanel mainPanel = new JPanel(new FlowLayout());
        mainPanel.add(new JLabel("Hello, world!"));
        mainPanel.add(new JButton("Button"));

        // Set the content-pane of this JFrame to the main
        // JPanel
        this.setContentPane(mainPanel);
        .....
    }
    .....
}
```

62

Một mẫu chương trình dùng JFrame

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

// A Swing GUI application inherits from javax.swing.JFrame
public class AnyClass extends JFrame {
    // private variables
    // .....
    /** Constructor to setup the GUI components */
    public AnyClass() {
        Container cp = this.getContentPane();
        // Content-pane sets layout
        cp.setLayout(new ....Layout());
        // Allocate the GUI components
        // .....
        // Content-pane adds components
        cp.add(...);
    }
}

```

63

Một mẫu chương trình dùng JFrame

```

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setTitle(".....");
        setSize(300, 150); //or pack();

        setVisible(true);
    }

    public static void main(String[] args) {
        // Run GUI codes in Event-Dispatching thread for
        // thread-safety
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new AnyClass();
            }
        });
    }
}

```

64

Ví dụ - Countdown

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*
/** The Countdown class using Java Swing instead Java AWT
public class Countdown extends JFrame {
    private JLabel lblCount;
    private JTextField tfCount;
    private JButton btnCount;
    /** Constructor to setup GUI components and event
        handling */
    public Countdown () {
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());

```

65

Countdown (tiếp)

```

cp.add(new JLabel("Counter"));
tfCount = new JTextField("0", 10);
tfCount.setEditable(false);
cp.add(tfCount);
JButton btnCount = new JButton("Countdown");
cp.add(btnCount);
btnCount.addActionListener(new BtnCountListener());
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("Countdown");
setSize(250, 100);
 setLocationRelativeTo(null); //appear at center
setVisible(true);
}

```

66

Countdown (tiếp)

```
public static void main(String[] args) {
    Countdown app = new Countdown();
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new Countdown();
        }
    });
}
```

67

Countdown (tiếp)

```
private class BtnCountListener implements ActionListener {
    @Override
    public void actionPerformed(ActionEvent e) {
        int count = 0;
        count = Integer.parseInt(tfCount.getText());
        if (tfCount.isEditable())
            tfCount.setEditable(false);
        count--;
        if(count < 1)
            btnCount.setEnabled(false);
        tfCount.setText(count + "");
    }
} // end of the BtnCountListener class
}//end of the Countdown class
```

68

Thiết lập thuộc tính hiển thị Component

```
// javax.swing.JComponent
// Thiết lập màu nền hậu cảnh
public void setBackground(Color bgColor)
// Thiết lập màu nền tiền cảnh
public void setForeground(Color fgcolor)
// Thiết lập font chữ
public void setFont(Font font)
// Thiết lập viền
public void setBorder(Border border)
public void setPreferredSize(Dimension dim)
public void setMaximumSize(Dimension dim)
public void setMinimumSize(Dimension dim)
// Thiết lập màu
public void setOpaque(boolean isOpaque)
// Thiết lập chỉ dẫn
public void setToolTipText(String toolTipMsg)
```

69

Thiết lập thuộc tính hiển thị Component

```
// javax.swing.JLabel, javax.swing.AbstractButton
// Thiết lập tên hiển thị trên nhãn, nút bấm
public void setText(String strText)
// Thiết lập hình biểu tượng
public void setIcon(Icon defaultIcon)
// Canh lề ngang: SwingConstants.RIGHT,
SwingConstants.LEFT...
public void setHorizontalAlignment(int alignment)
// Canh lề dọc: SwingConstants.TOP,
SwingConstants.BOTTOM...
public void setVerticalAlignment(int alignment)
// Căn lề cho phần tên
public void setHorizontalTextPosition(int
textPosition)
public void setVerticalTextPosition(int
textPosition)
```

70

Thiết lập thuộc tính hiển thị Component

```
//javax.swing.JTextField,
//javax.swing.JLabel,
//javax.swing.AbstractButton
public void setHorizontalAlignment(int
alignment)
// javax.swing.AbstractButton
//Thiết lập phím tắt
public void setMnemonic(int mnemonic)
```

71

Ví dụ - Thiết lập ảnh biểu tượng (icon)

```
ImageIcon iconDuke = null;
String imgFilename = "images/duke.gif";
URL imgURL =
getClass().getClassLoader().getResource(imgFilename);
if (imgURL != null) {
    iconDuke = new ImageIcon(imgURL);
} else {
    System.err.println("Couldn't find file: " +
imgFilename);
}

JLabel lbl = new JLabel("The Duke", iconDuke,
JLabel.CENTER);
lbl.setBackground(Color.LIGHT_GRAY);
lbl.setOpaque(true);

Container cp = getContentPane();
cp.add(lbl);
```

72

Vùng hiển thị và viền bao

```
public int getWidth()
public int getHeight()
public Dimension getSize()
public int getX()
public int getY()
//Vị trí tương đối trên lớp chứa
public Point getLocation()
//Vị trí tuyệt đối trên màn hình
public Point getLocationOnScreen()
```

73

Vùng hiển thị và viền bao – Ví dụ

```
import java.awt.*;
import javax.swing.*;
public class TestSize {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Display Area");
        Container cp = frame.getContentPane();
        cp.setLayout(new FlowLayout());
        JButton btnHello = new JButton("Hello");
        btnHello.setPreferredSize(new Dimension(100, 80));
        cp.add(btnHello);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 150); // or pack()
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

74

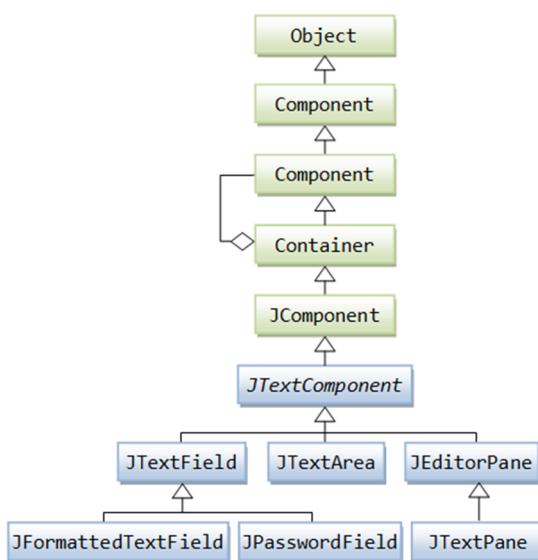
Thiết lập vị trí trên màn hình hiển thị

```
// Set methods (in java.awt.Window)
// (x, y) specifies the origin (top-left corner) of the
// window on the screen
public void setSize(int width, int height)
public void setLocation(int x, int y)
public void setBounds(int x, int y, int width, int height)
public void setSize(Dimension dim)
public void setLocation(Point origin)
public void setBounds(Rectangle r) // JDK 1.6

// The associated get methods (in java.awt.Component) are:
public int getWidth()
public int getHeight()
public int getX()
public int getY()
public Dimension getSize()
public Point getLocation()
public Rectangle getBounds()
```

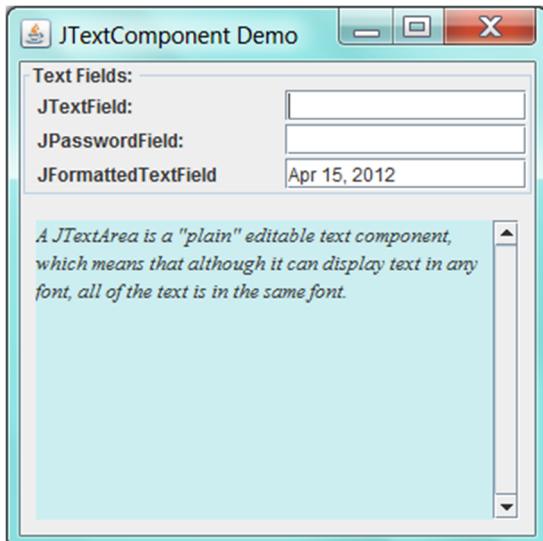
75

Các đối tượng nhập dữ liệu dạng văn bản



76

Ví dụ



77

Ví dụ

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

/** Test JTextField, JPasswordField,
JFormattedTextField, JTextArea */
@SuppressWarnings("serial")
public class JTextComponentDemo extends JFrame
{

    // Private variables of the GUI components
    JTextField tField;
    JPasswordField pwField;
    JTextArea tArea;
    JFormattedTextField formattedField;
```

78

Ví dụ (tiếp)

```
/** Constructor to set up all the GUI components */
public JTextComponentDemo() {
    // JPanel for the text fields
    JPanel tfPanel = new JPanel(new GridLayout(3, 2, 10,
        2));

    tfPanel.setBorder(BorderFactory.createTitledBorder(
        "Text Fields: "));

    // Regular text field (Row 1)
    tfPanel.add(new JLabel(" JTextField: "));
    tField = new JTextField(10);
    tfPanel.add(tField);
    tField.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            tArea.append("\nYou have typed " +
                tField.getText());
        }
    });
}
```

79

Ví dụ (tiếp)

```
// Password field (Row 2)

    tfPanel.add(new JLabel("JPasswordField: "));
    pwField = new JPasswordField(10);
    tfPanel.add(pwField);
    pwField.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e)
        {
            tArea.append("\nYour password is " + new
                String(pwField.getPassword()));
        }
    });
}
```

80

Ví dụ (tiếp)

```
// Formatted text field (Row 3)
tfPanel.add(new JLabel(" JFormattedTextField"));
formattedField = new JFormattedTextField(
    java.util.Calendar.getInstance().getTime();
tfPanel.add(formattedField);

// Create a JTextArea
tArea = new JTextArea("A JTextArea is a \"plain\""
    editable text component, which means that
    although it can display text in any font, all
    of the text is in the same font.";
tArea.setFont(new Font("Serif", Font.ITALIC, 13));
tArea.setLineWrap(true);
tArea.setWrapStyleWord(true);
tArea.setBackground(new Color(204, 238, 241));
```

81

Ví dụ (tiếp)

```
// Wrap the JTextArea inside a JScrollPane
JScrollPane tAreaScrollPane = new JScrollPane(tArea);
tAreaScrollPane.setBorder(
    BorderFactory.createEmptyBorder(10, 10, 10, 10));
tAreaScrollPane.setVerticalScrollBarPolicy(
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

// Setup the content-pane of JFrame in BorderLayout
Container cp = this.getContentPane();
cp.setLayout(new BorderLayout(5, 5));
cp.add(tfPanel, BorderLayout.NORTH);
cp.add(tAreaScrollPane, BorderLayout.CENTER);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("JTextComponent Demo");
setSize(350, 350);
setVisible(true);
}
```

82

Ví dụ (tiếp)

```
/** The entry main() method */
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new JTextComponentDemo();
        }
    });
}
```

83

Phân tích mã nguồn

- **JPasswordField:** tạo ô nhập mật khẩu

```
tfPanel.add(new JLabel("JPasswordField: "));
pwField = new JPasswordField(10);
tfPanel.add(pwField);
```

- **JFormattedTextField:** tạo ô nhập theo định dạng

```
tfPanel.add(new JLabel("JPasswordField: "));
pwField = new JPasswordField(10);
tfPanel.add(pwField);
```

84

Phân tích mã nguồn(tiếp)

- Tạo thanh cuộn (scroll bar)

```
JTextArea tArea = new JTextArea(...);
JScrollPane tAreaScrollPane = new JScrollPane(tArea);
tAreaScrollPane.setVerticalScrollBarPolicy(...);
tAreaScrollPane.setHorizontalScrollBarPolicy(...);
```

- Thay đổi thuộc tính cho JTextArea

```
// Append the str to the end of the document
public void append(String str)
// Replace with the str from startPos to endPos position
public void replaceRange(String str, int startPos,
                         int endPos)
// Insert the str after the specified position

public void insert(String str, int pos)
```

85

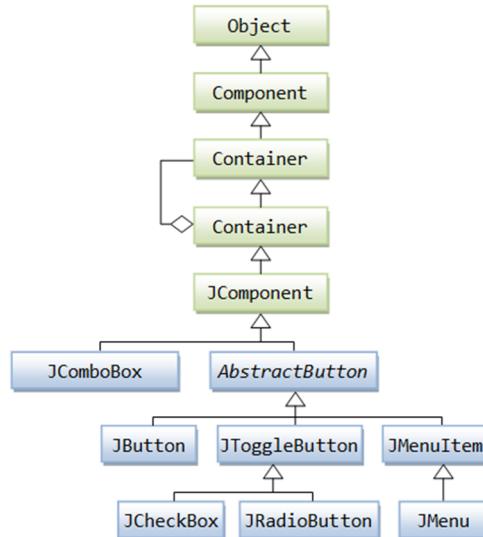
Phân tích mã nguồn(tiếp)

- Sử dụng JEditorPane như trình duyệt Web

```
JEditorPane editorPane = new JEditorPane();
editorPane.setEditable(false);
try {
    // Form a URL and display the HTML page on the editor-pane
    URL url = new URL
        ("http://www3.ntu.edu.sg/home/ehchua/programming/index.
         html");
    editorPane.setPage(url);
} catch (MalformedURLException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
// Wrap the JEditorPane inside a JScrollPane
JScrollPane editorScrollPane = new JScrollPane(editorPane);
editorScrollPane.setVerticalScrollBarPolicy(
    JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
setContentPane(editorScrollPane);
```

86

Sử dụng Button và ComboBox



87

Ví dụ



```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/** Counter with JRadioButton and JComboBox */
@SuppressWarnings("serial")
public class SwingCounterRadioCombo extends JFrame {
    private JTextField tfCount;
    private int count = 0; // counter's value
    private boolean countingUp = true;
    private int step = 1; // increment step size
  
```

88

Ví dụ (tiếp)

```
/** Constructor to setup the UI */
public SwingCounterRadioCombo () {
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());

    // Create JLabel and JTextField
    cp.add(new JLabel("Counter:"));
    tfCount = new JTextField("0", 5);
    tfCount.setEditable(false);
    tfCount.setHorizontalAlignment(JTextField.RIGHT);
    cp.add(tfCount);
```

89

Ví dụ (tiếp)

```
// Create JRadioButton for counting up and down
JRadioButton rbUp = new JRadioButton("Up",
                                    true);
rbUp.setMnemonic(KeyEvent.VK_U);
cp.add(rbUp);
rbUp.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        countingUp = true;
    }
});
```

90

Ví dụ (tiếp)

```

JRadioButton rbDown = new JRadioButton("Down");
rbDown.setMnemonic(KeyEvent.VK_D);
cp.add(rbDown);
rbDown.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        countingUp = false;
    }
});
// Setup a ButtonGroup to ensure exclusive
// selection
ButtonGroup btnGp = new ButtonGroup();
btnGp.add(rbUp);
btnGp.add(rbDown);

```

91

Ví dụ (tiếp)

```

// Create JComboBox for setting the count step size
add(new JLabel("Step:"));
final Integer[] steps = {1, 2, 3, 4, 5};
final JComboBox<Integer> comboCount = new
    JComboBox<Integer>(steps);
comboCount.setPreferredSize(
    new Dimension(60, 20));
cp.add(comboCount);
comboCount.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        if (e.getStateChange() ==
            ItemEvent.SELECTED) {
            step = (Integer)
                comboCount.getSelectedItem();
        }
    }
});

```

92

Ví dụ (tiếp)

```
// Create JButton for "Count"
JButton btnCount = new JButton("Count");
btnCount.setMnemonic(KeyEvent.VK_C);
cp.add(btnCount);
btnCount.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        if (countingUp) {
            count += step;
        } else {
            count -= step;
        }
        tfCount.setText(count + "");
    }
});
```

93

Ví dụ (tiếp)

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("Swing Counter with RadioButton &
        ComboBox");
setSize(480, 100);
setVisible(true);
}

public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new SwingCounterRadioCombo();
        }
    });
}
```

94

Bài tập trên lớp

- Thay đổi mã nguồn của lớp SwingCounterRadioCombo để mặc định ban đầu lựa chọn Down để đếm lùi

95

JCheckbox – Ví dụ



```
//In initialization code:  
chinButton = new JCheckBox("Chin");  
chinButton.setMnemonic(KeyEvent.VK_C);  
chinButton.setSelected(true);  
  
glassesButton = new JCheckBox("Glasses");  
glassesButton.setMnemonic(KeyEvent.VK_G);  
glassesButton.setSelected(true);
```

96

JCheckbox – Ví dụ(tiếp)

```

hairButton = new JCheckBox("Hair");
hairButton.setMnemonic(KeyEvent.VK_H);
hairButton.setSelected(true);

teethButton = new JCheckBox("Teeth");
teethButton.setMnemonic(KeyEvent.VK_T);
teethButton.setSelected(true);

//Register a listener for the check boxes.
chinButton.addItemListener(this);
glassesButton.addItemListener(this);
hairButton.addItemListener(this);
teethButton.addItemListener(this);

```

97

JCheckbox – Ví dụ(tiếp)

```

public void itemStateChanged(ItemEvent e) {
    ...
    Object source = e.getItemSelectable();

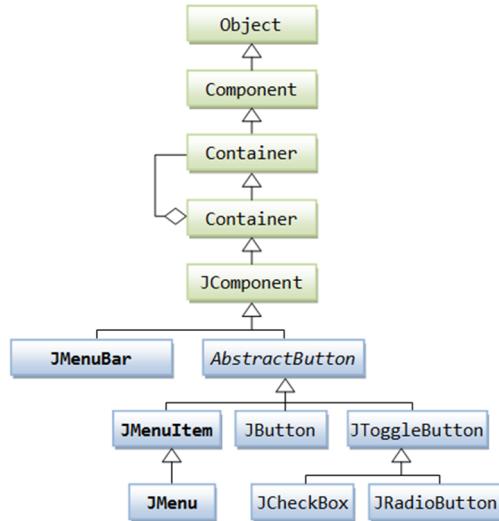
    if (source == chinButton) {
        //...make a note of it...
    } else if (source == glassesButton) {
        //...make a note of it...
    } else if (source == hairButton) {
        //...make a note of it...
    } else if (source == teethButton) {
        //...make a note of it...
    }

    if (e.getStateChange() == ItemEvent.DESELECTED)
        //...make a note of it...
    ...
    dosSomething();
}

```

98

JMenuBar, JMenu, JMenuItem



99

Ví dụ

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/** Testing menu-bar of JFrame */
public class TestJMenuBar extends JFrame {

    JTextField display;
    int count = 0;

    /** Constructor to setup the GUI */
    public TestJMenuBar() {
        // A menu-bar contains menus. A menu contains
        //menu-items (or sub-Menu)
        JMenuBar menuBar; // the menu-bar
        JMenu menu; // each menu in the menu-bar
        JMenuItem menuItem; // an item in a menu
    }
}
    
```

100

Ví dụ(tiếp)

```

menuBar = new JMenuBar();

// First Menu
menu = new JMenu("Menu-A");
menu.setMnemonic(KeyEvent.VK_A);
menuBar.add(menu);

menuItem = new JMenuItem("Up", KeyEvent.VK_U);
menu.add(menuItem); // the menu adds this item
menuItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        ++count;
        display.setText(count + "");
    }
});

```

101

Ví dụ(tiếp)

```

menuItem = new JMenuItem("Down",
                       KeyEvent.VK_D);
menu.add(menuItem); // the menu adds this item
menuItem.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) {
        --count;
        display.setText(count + "");
    }
});

```

102

Ví dụ(tiếp)

```
// Second Menu
menu = new JMenu("Menu-B");
menu.setMnemonic(KeyEvent.VK_B); // short-cut key
menuBar.add(menu); // the menu bar adds this menu

menuItem = new JMenuItem("Reset", KeyEvent.VK_R);
menu.add(menuItem); // the menu adds this item
menuItem.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        count = 0;
        display.setText(count + "");
    }
});

setJMenuBar(menuBar);
```

103

Ví dụ(tiếp)

```
Container cp = getContentPane();
cp.setLayout(new FlowLayout());
display = new JTextField("0", 10);
cp.add(display);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("Test MenuBar");
setSize(300, 100);
setVisible(true);
}
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new TestJMenuBar();
        }
    });
}
```

104

Tương tác qua hộp thoại (Dialog)

- Sử dụng phương thức:

`JOptionPane.showXxxDialog()`

```
// Prompt for user input
public static String showInputDialog(Object message, [Object
initialSelectionValue])
public static Object showInputDialog(Component parentComponent, Object
message,
    [String title], [int messageType], [Icon icon], [Object[] options],
    [Object initialValue])

// Asks a confirming question (yes/no/cancel)
public static int showConfirmDialog(Component parentComponent, Object
message,
    [String title], [int optionType], [int messageType], [Icon icon])

// Display a message
public static void showMessageDialog(Component parentComponent, Object
message,
    [String title], [int messageType], [Icon icon])

// Support all features of the above three methods
public static int showOptionDialog(Component parentComponent, Object message,
    String title, int optionType, int messageType, Icon icon, Object[]
options, Object initialValue)
```

105

Ví dụ 1

```
import javax.swing.*;
public class JOptionPaneTest {
    public static void main(String[] args) {
        // JOptionPane does not have to run under a
        // Swing Application (extends JFrame).
        // It can run directly under main().
        String inStr = JOptionPane.showInputDialog(
            null, "Ask for user input(returns a
String)", "Input Dialog",
            JOptionPane.PLAIN_MESSAGE);
        System.out.println("You have entered " +
            inStr);
        JOptionPane.showMessageDialog(null,
            "Display a message (returns void)!",
            "Message Dialog", JOptionPane.PLAIN_MESSAGE);
```

106

Ví dụ 1 (tiếp)

```

int answer = JOptionPane.showConfirmDialog(
    null, "Ask for confirmation (returns an
    int)", "Confirm Dialog",
    JOptionPane.YES_NO_CANCEL_OPTION);
switch (answer) {
    case JOptionPane.YES_OPTION:
        System.out.println("You clicked YES");
        break;
    case JOptionPane.NO_OPTION:
        System.out.println("You clicked NO");
        break;
    case JOptionPane.CANCEL_OPTION:
        System.out.println("You clicked Cancel");
        break;
}
}
}

```

107

Ví dụ 2 – Bắt buộc nhập giá trị hợp lệ

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class InputDialogWithValidation extends JFrame
{
    JTextField tfDisplay;

    /** Constructor to setup the GUI components */
    public InputDialogWithValidation() {
        Container cp = getContentPane();
        cp.setLayout(new FlowLayout());

        tfDisplay = new JTextField(10);
        tfDisplay.setEditable(false);
        cp.add(tfDisplay);
    }
}

```

108

Ví dụ 2 (tiếp)

```

 JButton btn = new JButton("Input");
 cp.add(btn);
 btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        boolean validInput = false;
        int numberIn;
        String inputStr = JOptionPane.showInputDialog(
            "Enter a number [1-9]: ");
        do {
            try {
                numberIn = Integer.parseInt(inputStr);
            } catch (NumberFormatException ex) {
                numberIn = -1;
            }
        }
    }
}

```

109

Ví dụ 2 (tiếp)

```

if (numberIn < 1 || numberIn > 9) {
    inputStr = JOptionPane.showInputDialog(
        "Invalid numner! Enter a number [1-9]: ");
} else {
    JOptionPane.showMessageDialog(
        null, "You have entered " + numberIn);
    validInput = true;
}
} while (!validInput); // repeat if input is not valid
tfDisplay.setText(numberIn + "");
}
);

```

110

JTabbedPane

```
JTabbedPane tabPane = new JTabbedPane();

JPanel generalPanel = new JPanel(new FlowLayout());
tabPane.add("General", generalPanel);

JPanel advancedTab = new JPanel(new FlowLayout());
tabPane.add("Advanced", advancedTab);

Container cp = this.getContentPane();
cp.add(tabPane);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setTitle("JTabbedPane Demo");
pack();
setVisible(true);
```

111

JTable

- JTable là đối tượng được sử dụng để hiển thị dữ liệu dưới dạng bảng

- Khởi tạo:

```
JTable tblObject = new JTable (Object[][] data,
                           Object[] colNames)
```

data: mảng 2 chiều chứa dữ liệu cần hiển thị

colNames: mảng 1 chiều chứa tên các cột

- Phương thức:

- setPreferredScrollableViewportSize(): thiết lập kích thước vùng hiển thị
- setFillsViewportHeight(true): bảng được hiển thị với chiều cao tối đa bằng chiều cao vùng hiển thị

- Tạo thanh cuộn:

```
JScrollPane scrollPane = new JScrollPane(JTable);
```

112

JTable – Ví dụ

```
String[] columnNames = {"First Name", "Last Name",
                       "Sport", "# of Years", "Vegetarian"};

Object[][] data = {{"Kathy", "Smith", "Snowboarding",
                   new Integer(5), new Boolean(false)},
                  {"John", "Doe", "Rowing",
                   new Integer(3), new Boolean(true)},
                  {"Sue", "Black", "Knitting",
                   new Integer(2), new Boolean(false)},
                  {"Jane", "White", "Speed reading",
                   new Integer(20), new Boolean(true)},
                  {"Joe", "Brown", "Pool",
                   new Integer(10), new Boolean(false)};
```

113

JTable – Ví dụ (tiếp)

```
JTable table = new JTable(data, columnNames);
table.setPreferredScrollableViewportSize(
    new Dimension(500, 70));
table.setFillsViewportHeight(true);
JScrollPane scrollPane = new JScrollPane(table);
```

114

Thay đổi các thành phần giao diện

- Các lưu ý:

- JPanel mặc định sử dụng FlowLayout, mặc định căn lề giữa, khoảng cách giữa các thành phần theo chiều ngang và dọc là 5px
- Content Pane mặc định sử dụng BorderLayout
- Các phương thức thiết lập kích thước và vị trí cửa sổ cần đặt cuối cùng, khi đã hoàn thành việc thêm các phần tử trên giao diện
- Phương thức dùng để đóng gói các phần tử trên Container: setSize(), pack(), validate(), invalidate()
- Đóng gói các phần tử cho Component: revalidate(), repaint()

115

Thay đổi các thành phần giao diện

- add(JComponent): thêm một phần tử trên phần tử đã có:
 - Mọi phần tử giao diện trong Java Swing đều có
- remove(JComponent): xóa một phần tử
- removeAll(): xóa tất cả các phần tử
- doLayout(): thay đổi các thuộc tính của Layout(khoảng cách giữa các phần tử)
- setLayout(LayoutManager): thay đổi Layout
- validate(): Hiển thị lại giao diện khi có thay đổi về Layout
- repaint(): Làm sạch lưu ảnh của các phần tử bị xóa

116

Nhiều hơn nữa về Swing

<http://docs.oracle.com/javase/tutorial/uiswing/components/index.html>

117

3. GIỚI THIỆU MÔ HÌNH MVC

118

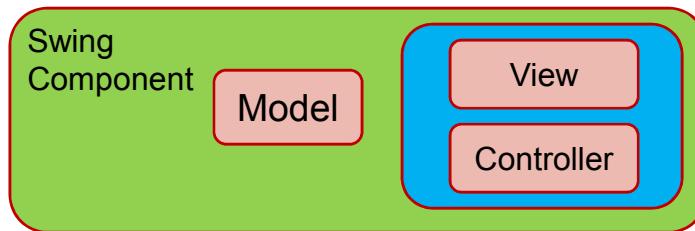
MVC là gì?

- Mô hình thiết kế phần mềm 3 thành phần: Model – View – Control
- Model:
 - Mô hình hóa các đối tượng chứa dữ liệu cần xử lý
 - Cung cấp các phương thức để truy cập dữ liệu
 - Mô hình hóa các hoạt động nghiệp vụ
- View:
 - Cung cấp giao diện cho người dùng nhập/xuất dữ liệu
 - Kiểm tra tính hợp lệ của dữ liệu vào
 - Bắt các sự kiện trên giao diện
- Controller: nhận các sự kiện được truyền từ View, gọi đến các phương thức tương ứng của Model

119

Java Swing và MVC

- Java Swing được xây dựng dựa trên mô hình MVC
- Mỗi đối tượng trong Java Swing đóng gói 3 thành phần:
 - Model: chứa dữ liệu và các phương thức thao tác trên dữ liệu đó
 - View: các phương thức để hiển thị đối tượng
 - Controller: bắt và xử lý sự kiện trên đối tượng
- Ví dụ: Xem đoạn mã tạo đối tượng ComboBox sau đây
- Mô hình thực sự của Java Swing Component



120

Ví dụ

```
// Create JComboBox for setting the count step size
add(new JLabel("Step:"));
final Integer[] steps = {1, 2, 3, 4, 5};
final JComboBox<Integer> comboBox = new
    JComboBox<Integer>(steps);
comboBox.setPreferredSize(
    new Dimension(60, 20));
cp.add(comboBox);
comboBox.addItemListener(new ItemListener() {
    @Override
    public void itemStateChanged(ItemEvent e) {
        if (e.getStateChange() ==
            ItemEvent.SELECTED) {
            step = (Integer)
                comboBox.getSelectedItem();
    }
});
```

Controller

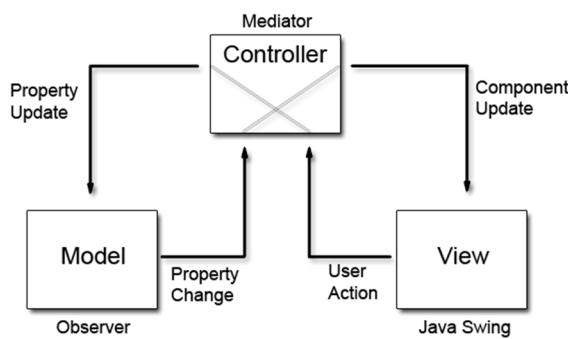
View

Model

121

Xây dựng phần mềm theo mô hình MVC

- Khi chương trình phát triển thêm nhiều tính năng, hoặc quá trình xử lý phức tạp hơn, mô hình MVC đóng gói trên đối tượng Swing không còn đáp ứng được.
- Xây dựng phần mềm theo mô hình MVC



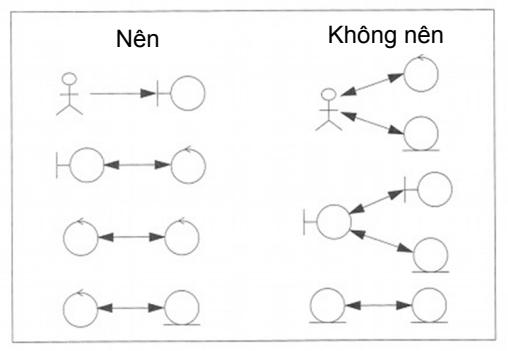
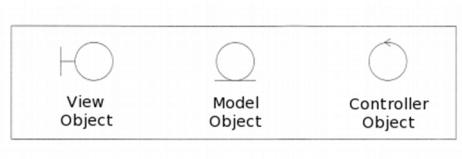
122

Lợi ích của MVC

- Cho phép phân tách hệ thống lớn thành 3 nhóm thành phần → dễ dàng hơn trong thiết kế, phát triển và bảo trì
- Các thành phần có thể phát triển đồng thời
- Từ một Model có thể hiển thị trên các View khác nhau. Ví dụ: cùng một tập số liệu có thể hiển thị dưới dạng bảng, biểu đồ cột, biểu đồ tròn...
- Để tăng đảm bảo tính cộng tác khi phát triển đồng thời, các lớp cần phải được triển khai từ các giao diện

123

Giao tiếp giữa các thành phần



124

Các bước xử lý yêu cầu người dùng

1. Người dùng thực hiện một hành vi trên View

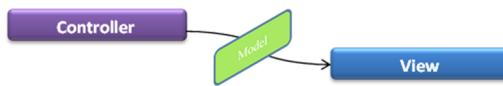
2. View bắt sự kiện, chuyển yêu cầu cho Controller xử lý



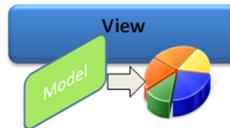
3. Controller gọi phương thức tương ứng mà Model cung cấp



4. Controller nhận kết quả trả về (có thể là một Model chứa dữ liệu) và chuyển cho View để hiển thị



5. View thay đổi khung nhìn và hiển thị kết quả



125

Ví dụ

- Một cửa hàng quản lý danh mục các mặt hàng trong file văn bản, với mỗi dòng có chứa thông tin theo dạng sau:

Product ID | Product name | Amount

- Nhân viên của cửa hàng khi muốn xem danh mục hàng phải qua thao tác đăng nhập. Thông tin tài khoản của nhân viên được lưu trên file văn bản có định dạng mỗi dòng như sau:

EmployeeID EmployeePassword Permission(0 or 1)

- Chương trình có chức năng để thêm một nhân viên mới vào cửa hàng với các thông tin như mô tả ở trên
- Xây dựng chương trình với giao diện đồ họa để đáp ứng yêu cầu trên.

126

Model

- interface IAccount: tài khoản nhân viên
 - Các phương thức getter và setter
 - Lớp Account triển khai từ IAccount: userID, password, permission
- interface IAccountList: danh sách tài khoản
 - void addAccount(IAccount)
 - int check(String, String)
 - Lớp AccountList triển khai từ IAccountList:
 - Thuộc tính là mảng IAccount[]
 - Phương thức khởi tạo: đọc dữ liệu từ file vào mảng

127

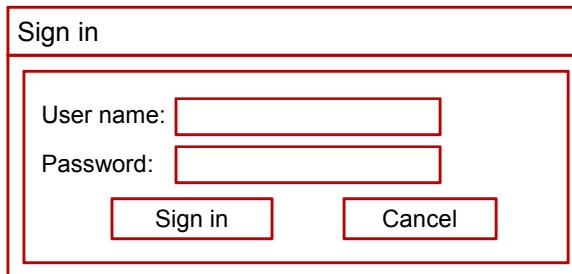
Model

- interface IProduct: một mặt hàng trong kho
 - Các phương thức getter và setter
 - Lớp Product triển khai từ IProduct: productID, productName, amount
- interface IProductList: danh sách mặt hàng
 - Phương thức getter
 - Lớp ProductList triển khai từ IProductList: thuộc tính là mảng IProduct []

128

View

- SignInForm: Hiển thị khi chương trình được khởi động



- Sự kiện và xử lý:

- Nút Signin được nhán: Chương trình kiểm tra thông tin đăng nhập
- Nút Cancel được nhán: Kết thúc chương trình
- Cửa sổ bị tắt: Kết thúc chương trình

129

View(tiếp)

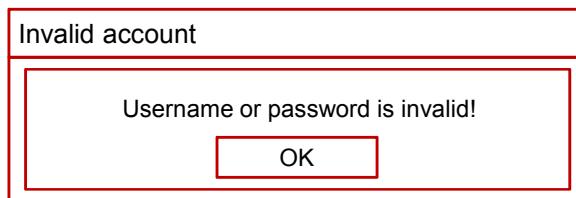
- interface ISignInForm: quy định các phương thức cần được triển khai trên SignInForm :

- String getUsernameOnSignInForm(): trả về user name người dùng đã nhập
- String getPasswordOnSignInForm(): trả về giá trị mật khẩu người dùng đã nhập
- void setSignInButtonActionListener(ActionListener)
- void closeForm(): đóng form

130

View (tiếp)

- InvalidDialog: Hộp thoại thông báo đăng nhập thất bại:



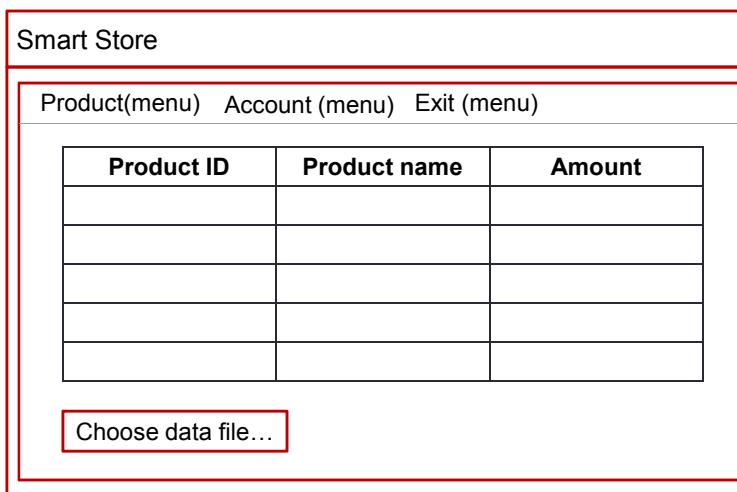
- Sự kiện:

- Nút OK được nhấn: Quay trở lại cửa sổ đăng nhập
- Cửa sổ bị tắt: Quay trở lại cửa sổ đăng nhập

131

View(tiếp)

- MainWindow



132

View(tiếp)

- Nếu người dùng đăng nhập thành công, cửa sổ chính xuất hiện với hệ thống menu gồm:
- Product Menu: hiển thị thông tin hàng hóa dưới dạng bảng. Nội dung menu này là mặc định. Trên menu này có nút bấm “Choose data file...” để người dùng có thể chọn file văn bản chứa thông tin hàng hóa.
- Account Menu: có 2 mục
 - Sign out: Người dùng chọn mục này, màn hình Sign in xuất hiện
 - Create new account: Mục này chỉ xuất hiện khi người đăng nhập có quyền admin (Permission là 1). Người dùng chọn mục này, cửa sổ mới xuất hiện để thêm tài khoản mới
- Exit Menu: người dùng chọn mục này, chương trình kết thúc

133

View(tiếp)

- interface `IMainWindow` quy định các phương thức cần triển khai trên `MainWindow`:
 - `void setSignOutActionListener(ActionListener)`
 - `void setCreateAccountActionListener(ActionListener)`
 - `Void addCreateAccountMenu()`

134

View (tiếp)

- CreateNewAccountForm

The diagram shows a rectangular form with a red border. Inside, there are three input fields: 'User name:' with a text box, 'Password:' with a text box, and 'Administrator account:' with a text box. Below these is a radio button group with two options: 'Yes' and 'No', where 'No' is selected. At the bottom are two buttons: 'Create' and 'Cancel'.

135

View (tiếp)

- interface `ICreateNewAccountForm`: quy định các phương thức cần được triển khai trên `CreateNewAccountForm`:
 - `String getNewUserName ()`: trả về user name người dùng mới
 - `String getNewPassword ()`: trả về giá trị mật khẩu người dùng mới
 - `void setActionListenerOnCreateButton(ActionListener)`

136

Controller

- SignInController: kiểm tra thông tin tài khoản khi nhân viên đăng nhập
- DisplayProductController: hiển thị thông tin trên Product Menu
- SignoutController
- CreateAccountController: xử lý khi tạo tài khoản mới

137