UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

NGÔN NGỮ LẬP TRÌNH JAVA

BUỔI 11
LẬP TRÌNH GIAO DIỆN VỚI AWT, SWING (TT)

GVGD: ThS. Lê Thanh Trọng

NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

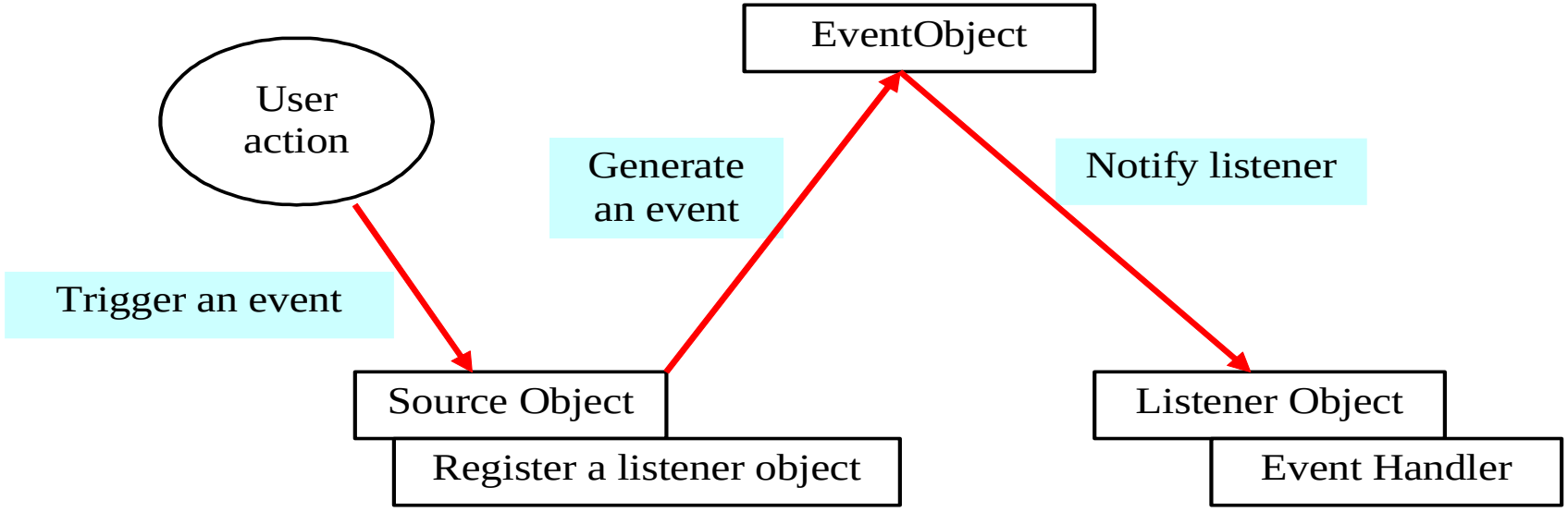
1. **Mô hình xử lý sự kiện**
2. **Xử lý sự kiện chuột**
3. **Xử lý sự kiện bàn phím**
4. **Một số component thông dụng**
   - Jlabel
   - Jbutton
   - JCheckBox và JRadioButton
   - Menu
   - Dialog
   - JFileChooser
   - JColorChooser

NỘI DUNG

1. **Mô hình xử lý sự kiện**
2. Xử lý sự kiện chuột
3. Xử lý sự kiện bàn phím
4. Một số component thông dụng

Mô hình xử lý sự kiện

- 3 yếu tố quan trọng trong mô hình xử lý sự kiện:
  - Nguồn phát sinh sự kiện (event source)
  - Sự kiện (event object)
  - Bộ lắng nghe sự kiện (event listener)



Mô hình xử lý sự kiện

- Khai báo lớp xử lý sự kiện

  public class MyClass implements <Event>Listener

- Cài đặt các phương thức trong listener interface.
  - Ví dụ: ActionListener

```
public void actionPerformed(ActionEvent e) {
    ...//
}
```

- Gắn bộ xử lý vào component

  someComponent.add<Event>Listener(instanceOfMyClass);

Mô hình xử lý sự kiện

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

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

NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

1. Mô hình xử lý sự kiện
2. **Xử lý sự kiện chuột**
3. Xử lý sự kiện bàn phím
4. Một số component thông dụng

Xử lý sự kiện chuột

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

- Event-listener của mouse events
  - *MouseListener:* bắt các sự kiện mà chuột click vào các component
  - *MouseMotionListener:* bắt các sự kiện khi chuột di chuyển trên các component

MouseListener and MouseMotionListener interface methods	
Methods of interface **MouseListener**	
`public void mousePressed( MouseEvent event )`	Called when a mouse button is pressed with the mouse cursor on a component.
`public void mouseClicked( MouseEvent event )`	Called when a mouse button is pressed and released on a component without moving the mouse cursor.
`public void mouseReleased( MouseEvent event )`	Called when a mouse button is released after being pressed. This event is always preceded by a `mousePressed` event.
`public void mouseEntered( MouseEvent event )`	Called when the mouse cursor enters the bounds of a component.
`public void mouseExited( MouseEvent event )`	Called when the mouse cursor leaves the bounds of a component.
Methods of interface **MouseMotionListener**	
`public void mouseDragged( MouseEvent event )`	Called when the mouse button is pressed with the mouse cursor on a component and the mouse is moved. This event is always preceded by a call to `mousePressed`.
`public void mouseMoved( MouseEvent event )`	Called when the mouse is moved with the mouse cursor on a component.

Ví dụ

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
1  // Fig. 12.17: MouseTracker.java
2  // Demonstrating mouse events.
3
4  // Java core packages
5  import java.awt.*;
6  import java.awt.event.*;
7
8  // Java extension packages
9  import javax.swing.*;
10
11 public class MouseTracker extends JFrame
12    implements MouseListener,
MouseMotionListener {
13
14    private JLabel statusBar;
15
16    // set up GUI and register mouse event
handlers
17    public MouseTracker()
18    {
19       super("Demonstrating Mouse Events" );
20
21       statusBar = new JLabel();
22       getContentPane().add(statusBar,
BorderLayout.SOUTH);
23
24       // application listens to its own
mouse events
25       addMouseListener(this);
26       addMouseMotionListener(this);
27
28       setSize(275, 100);
29       setVisible(true);
30    }
31
32    // MouseListener event handlers
33
34    // handle event when mouse released
immediately after press
```

Ví dụ

```
35     public void mouseClicked(MouseEvent event )
36     {
37        statusBar.setText( "Clicked at [" +
event.getX() +
38           ", " + event.getY() + "]" );
39     }
40
41     // handle event when mouse pressed
42     public void mousePressed(MouseEvent event )
43     {
44        statusBar.setText( "Pressed at [" +
event.getX() +
45           ", " + event.getY() + "]" );
46     }
47
48     // handle event when mouse released after
dragging
49     public void mouseReleased(MouseEvent event
)
50     {
51        statusBar.setText("Released at [" +
event.getX() +
52           ", " + event.getY() + "]" );
53     }
54
55     // handle event when mouse enters area
56     public void mouseEntered(MouseEvent event)
57     {
58        JOptionPane.showMessageDialog(null,
"Mouse in window");
59     }
60
61     // handle event when mouse exits area
62     public void mouseExited(MouseEvent event )
63     {
64        statusBar.setText("Mouse outside window"
);
65     }
66
67     // MouseMotionListener event handlers
68
69     // handle event when user drags mouse with
button pressed
70     public void mouseDragged(MouseEvent event )
```

Ví dụ

```
71      {
72         statusBar.setText("Dragged at [" + event.getX() +
73            ", " + event.getY() + "]" );
74      }
75
76      // handle event when user moves mouse
77      public void mouseMoved(MouseEvent event )
78      {
79         statusBar.setText("Moved at [" + event.getX() +
80            ", " + event.getY() + "]" );
81      }
82
83      // execute application
84      public static void main(String args[] )
85      {
86         MouseTracker application = new MouseTracker();
87
88         application.setDefaultCloseOperation(
89            JFrame.EXIT_ON_CLOSE);
90      }
91
92   }  // end class MouseTracker
```

Ví dụ


Demonstrating Mouse Events
Mouse outside window


Message
Mouse in window
OK


Demonstrating Mouse Events
Moved at [68, 34]


Demonstrating Mouse Events
Pressed at [135, 49]


Demonstrating Mouse Events
Dragged at [45, 47]


Demonstrating Mouse Events
Dragged at [45, 47]


Demonstrating Mouse Events
Clicked at [195, 56]

NỘI DUNG

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

1. Mô hình xử lý sự kiện
2. Xử lý sự kiện chuột
3. **Xử lý sự kiện bàn phím**
4. Một số component thông dụng

Xử lý sự kiện bàn phím

- Interface KeyListener
- Dùng để xử lý key events
- Phát sinh khi 1 phím được nhấn và thả ra
- ***KeyEvent***: Chứa virtual key code đại diện cho các phím

Ví dụ

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
1  // Fig. 12.22: KeyDemo.java
2  // Demonstrating keystroke events.
3
4  // Java core packages
5  import java.awt.*;
6  import java.awt.event.*;
7
8  // Java extension packages
9  import javax.swing.*;
10
11 public class KeyDemo extends JFrame
implements KeyListener{
12    private String line1 = "", line2 =
"";
13    private String line3 = "";
14    private JTextArea textArea;
15
16    // set up GUI
17    public KeyDemo()
18    {
19       super("Demonstrating Keystroke
Events" );
20
21       // set up JTextArea
22       textArea = new JTextArea(10, 15);
23       textArea.setText( "Press any key
on the keyboard..." );
24       textArea.setEnabled(false );
25       getContentPane().add(textArea);
26
27       // allow frame to process Key
events
28       addKeyListener(this);
29
30       setSize(350, 100);
31       setVisible(true);
32    }
33
34    // handle press of any key
35    public void keyPressed(KeyEvent event
```

Ví dụ

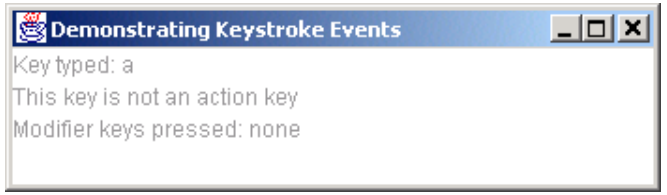
```
36     {
37        line1 = "Key pressed: " +
38           event.getKeyText(
event.getKeyCode() );
39        setLines2and3(event);
40     }
41
42     // handle release of any key
43     public void keyReleased(KeyEvent event)
44     {
45        line1 = "Key released: " +
46           event.getKeyText(
event.getKeyCode());
47        setLines2and3(event);
48     }
49
50     // handle press of an action key
51     public void keyTyped(KeyEvent event)
52     {
53        line1 = "Key typed: " +
event.getKeyChar();
54        setLines2and3(event);
55     }
56
57     // set second and third lines of output
58     private void setLines2and3(KeyEvent
event)
59     {
60        line2 = "This key is " +
61           (event.isActionKey() ? "" : "not
" ) +
62           "an action key";
63
64        String temp =
65           event.getKeyModifiersText(
event.getModifiers());
66
67        line3 = "Modifier keys pressed: " +
68           (temp.equals("") ? "none" : temp
);
69
```

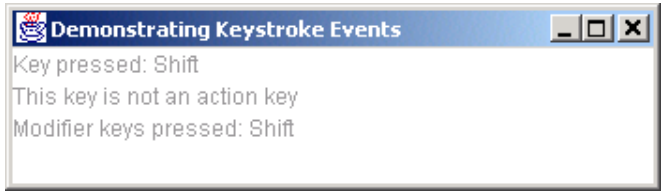
Ví dụ

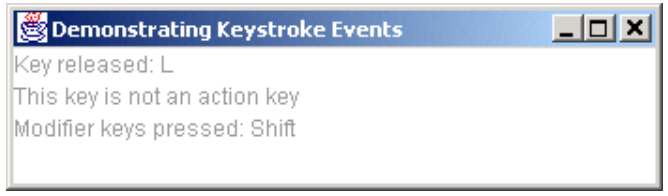
UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

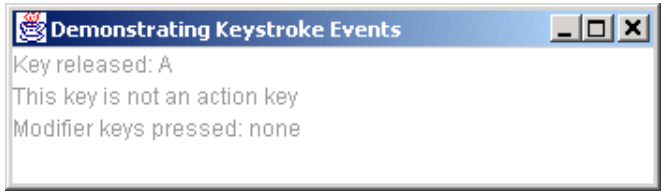
```
70          textArea.setText(
71             line1 + "\n" + line2 + "\n" + line3 + "\n" );
72       }
73
74       // execute application
75       public static void main(String args[])
76       {
77          KeyDemo application = new KeyDemo();
78
79          application.setDefaultCloseOperation(
80             JFrame.EXIT_ON_CLOSE );
81       }
82
83    }  // end class KeyDemo
```

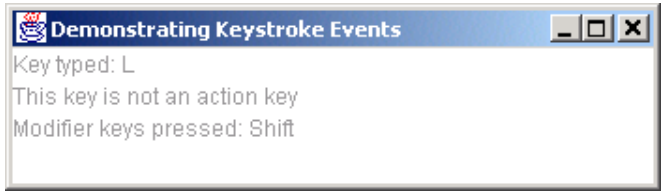
Ví dụ

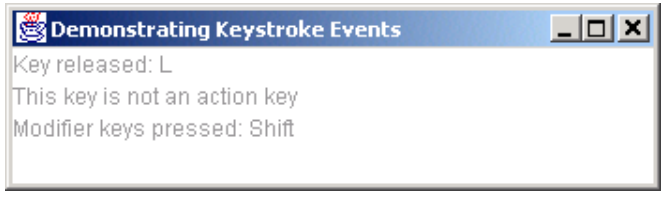

Demonstrating Keystroke Events
Key typed: a
This key is not an action key
Modifier keys pressed: none

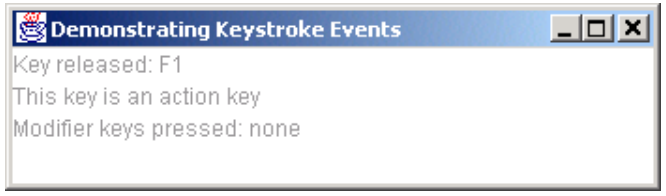

Demonstrating Keystroke Events
Key pressed: Shift
This key is not an action key
Modifier keys pressed: Shift


Demonstrating Keystroke Events
Key released: L
This key is not an action key
Modifier keys pressed: Shift


Demonstrating Keystroke Events
Key released: A
This key is not an action key
Modifier keys pressed: none


Demonstrating Keystroke Events
Key typed: L
This key is not an action key
Modifier keys pressed: Shift


Demonstrating Keystroke Events
Key released: L
This key is not an action key
Modifier keys pressed: Shift


Demonstrating Keystroke Events
Key released: F1
This key is an action key
Modifier keys pressed: none

NỘI DUNG

1. Mô hình xử lý sự kiện
2. Xử lý sự kiện chuột
3. Xử lý sự kiện bàn phím
4. **Một số component thông dụng**
   - Jlabel
   - Jbutton
   - JCheckBox và JRadioButton
   - Menu
   - Dialog
   - JFileChooser
   - JColorChooser

JLabel

- Label dùng để hiển thị một chuỗi văn bản thông thường nhằm mô tả thêm thông tin cho các đối tượng khác
- Các constructor*JLabel()*

  JLabel(String text)

  JLabel(String text,int hAlignment)

  JLabel(Icon icon)

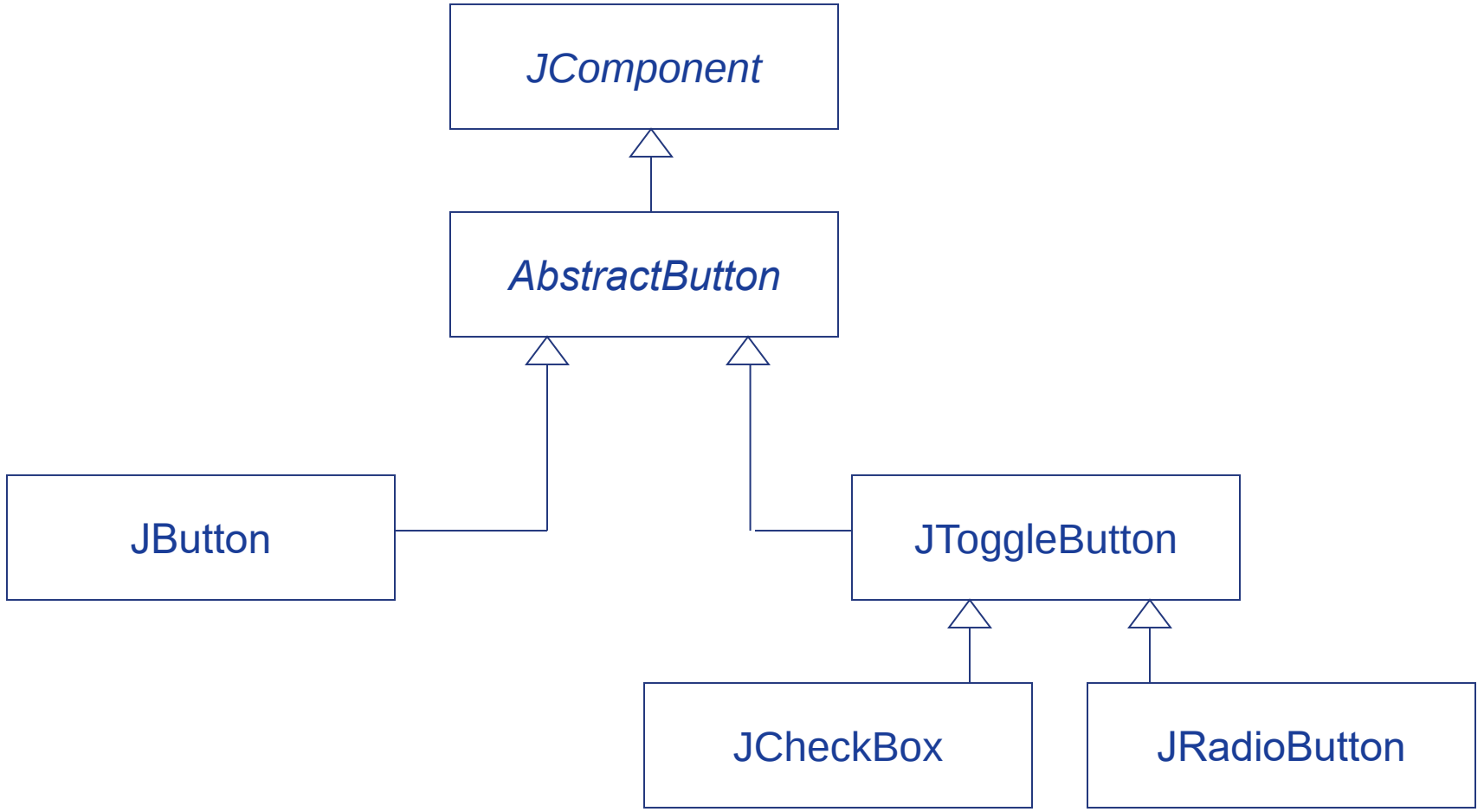
  JLabel(Icon icon, int hAlignment)

  JLabel(String text,Icon icon,int hAlignment)

Các thuộc tính JLabel

- text
- icon
- horizontalAlignment
- verticalAlignment

Swing Button Classes



JButton

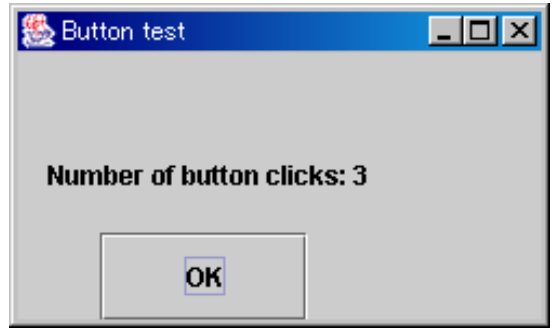
- Button là một thành phần gây ra một sự kiện hành động khi được kích chuột
- Constructor
  - *JButton()*
  - *JButton(String text)*
  - *JButton(String text, Icon icon)*
  - *JButton(Icon icon)*

Các thuộc tính JButton

- text
- icon
- mnemonic
- horizontalAlignment
- verticalAlignment
- horizontalTextPosition
- verticalTextPosition

Đáp ứng các sự kiện JButton

```
public void actionPerformed(ActionEvent e)
{
  // Get the button label
  String actionCommand = e.getActionCommand();
  // Make sure the event source is Left button
  if (e.getSource() instanceof JButton)
    // Make sure it is the right button
    if ("Left".equals(actionCommand))
       System.out.println ("Button pressed!");
}
```

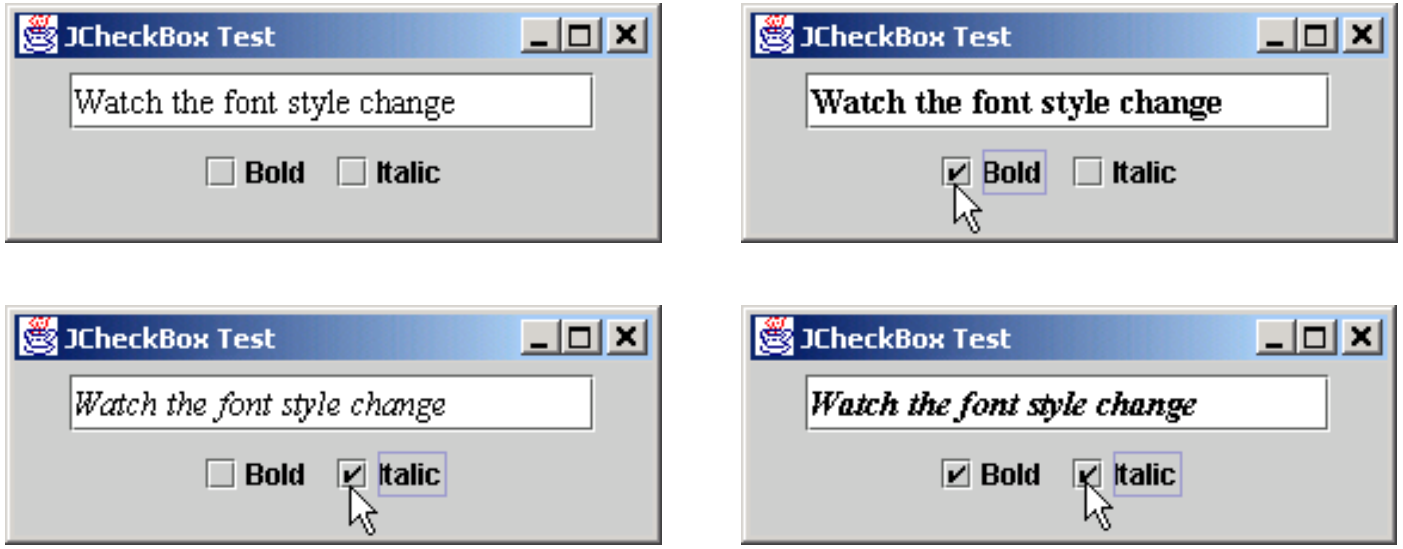


JCheckBox và JRadioButton

- Các nút lệnh thay đổi trạng thái
  - Nhận các giá trị on/off hoặc true/false
  - Swing hỗ trợ các kiểu:
    - JCheckBox
    - JRadioButton
- Item Event
  - Được tạo ra khi người dùng chọn các mục khác nhau trên JCheckBox, JRadioButton,..
  - Các phương thức
    - Object *getItem*(): trả về mục được chọn
    - int *getStateChange*(): trả về trạng thái trạng thái của mục chọn (DESELECTED/SELECTED)

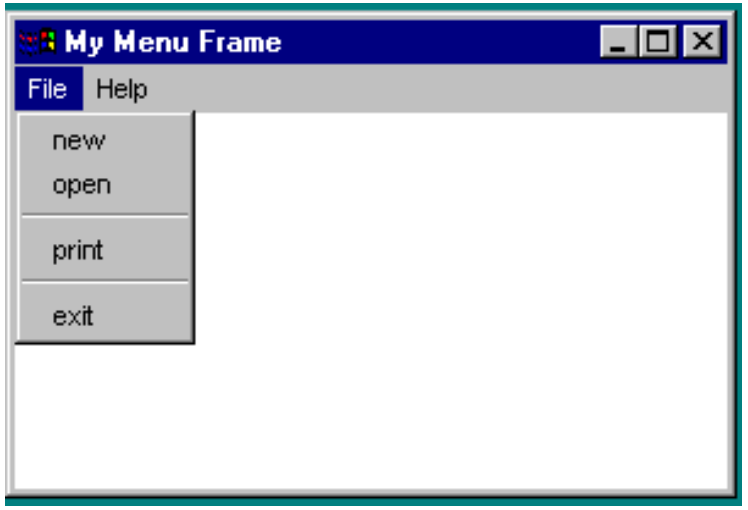
JCheckBox và JRadioButton

- *void itemStateChanged(ItemEvent e):* được gọi thi hành khi người dùng chọn hoặc bỏ chọn 1 mục



Menus

- Java cung cấp một số lớp - *JMenuBar, JMenu, JMenuItem, JCheckBoxMenuItem, và JRadioButtonMenuItem* - để thực thi menu trong một frame
- Một JFrame hoặc JApplet có thể chứa một menu bar trên đó có gắn các pull-down menu
- Các menu chứa các menu item để người dùng lựa chọn (hoặc bật/tắt)



Lớp JMenuBar

- Menu bar chứa các menu; menu bar chỉ có thể được thêm vào 1 frame. Đoạn code sau tạo và thêm một JMenuBar vào 1 frame:

```
JFrame f = new JFrame();
f.setSize(300, 200);
f.setVisible(true);
JMenuBar mb = new JMenuBar();
f.setJMenuBar(mb);
```

Lớp Menu

- Gắn các menu vào một JMenuBar
- Ví dụ, tạo 2 menu *File* và *Help*, và thêm vào *JMenuBar* **mb**:

```
JMenu fileMenu = new JMenu("File", false);
JMenu helpMenu = new JMenu("Help", true);
mb.add(fileMenu);
mb.add(helpMenu);
```

Lớp JMenuItem

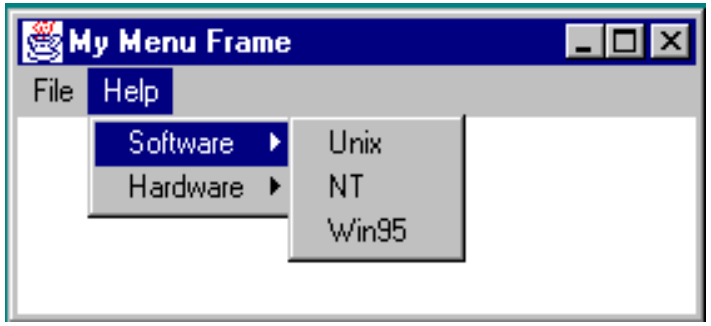
- Đoạn code sau thêm các mục chọn (menu item) và các separator trong menu fileMenu:

```
fileMenu.add(new JMenuItem("New"));
fileMenu.add(new JMenuItem("Open"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Print"));
fileMenu.addSeparator();
fileMenu.add(new JMenuItem("Exit"));
```

Submenus

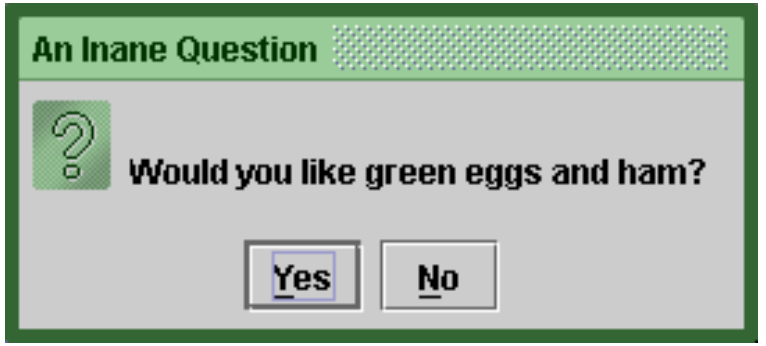
- Có thể thêm các *submenus* vào các *menuitem*
- Ví dụ, thêm các submenu "Unix", "NT", và "Win95" vào trong mục chọn "Software"

```
JMenu softwareHelpSubMenu = new JMenu("Software");
JMenu hardwareHelpSubMenu = new JMenu("Hardware");
helpMenu.add(softwareHelpSubMenu);
helpMenu.add(hardwareHelpSubMenu);
softwareHelpSubMenu.add(new JMenuItem("Unix"));
softwareHelpSubMenu.add(new JMenuItem("NT"));
softwareHelpSubMenu.add(new JMenuItem("Win95"));
```



Dialog Boxes

- Nhận thông tin từ người sử dụng
  - số liệu,…
  - danh mục tập tin,…
- Hiển thị kết quả
  - hiển thị thông tin cảnh báo
  - in kết quả lên màn hình,…



JOptionPane

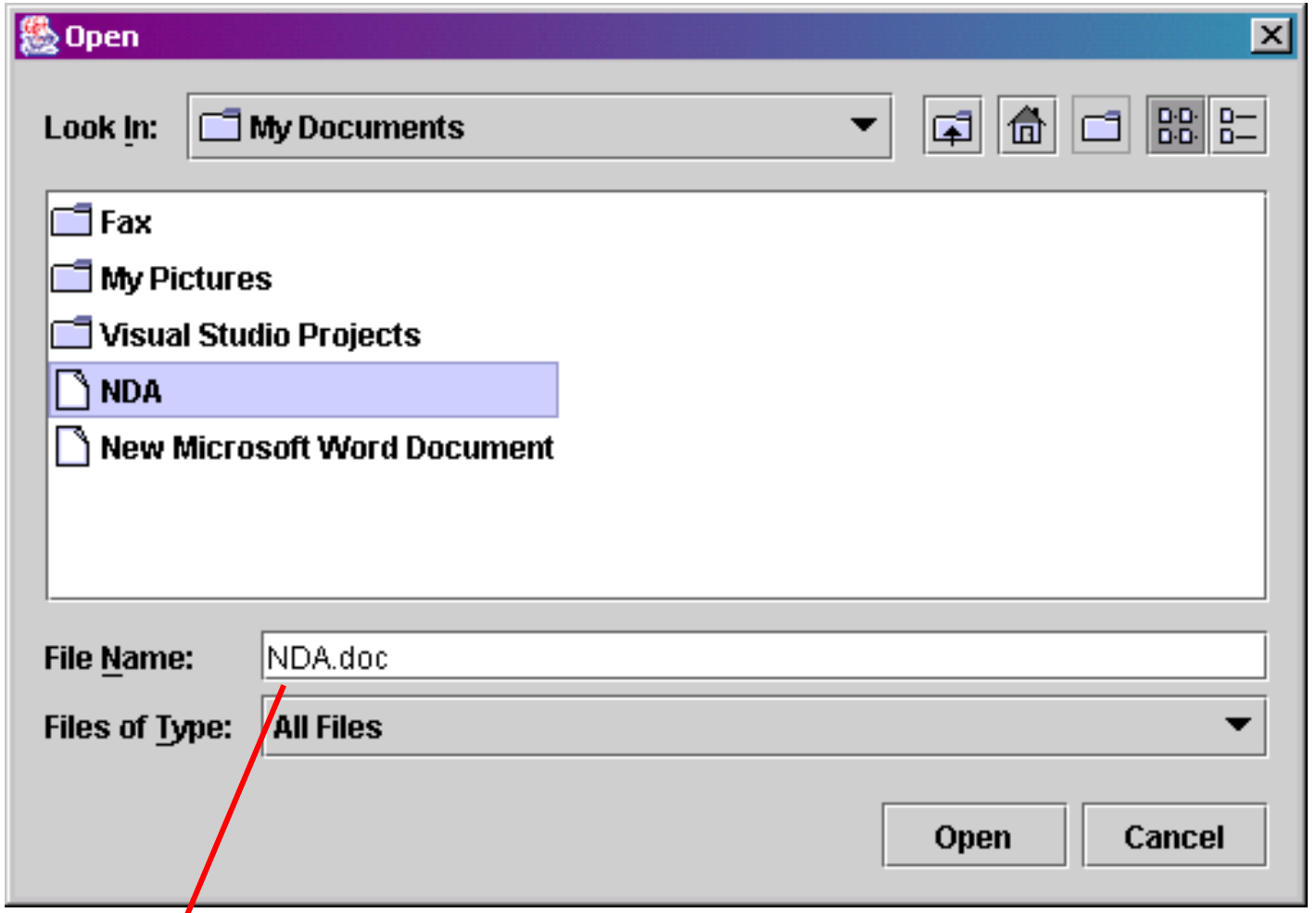
- *showConfirmDialog:* Yêu cầu xác nhận *yes/no/cancel*
- *showInputDialog:* Yêu cầu nhập liệu
- *showMessageDialog:* Thông báo
- *showOptionDialog:* Kết hợp các tính năng trên

JOptionPane.showInputDialog("Enter your home directory");

JFileChooser

- JFileChooser cho phép hiển thị một hộp thoại cho người dùng để chọn một file hay thư mục
- Các phương thức
  - *int showOpenDialog(Component parent):* mở hộp thoại đọc tập tin
  - *int showSaveDialog(Component parent):* mở hộp thoại lưu tập tin
  - *JFileChooser():* tạo đối tượng dùng để mở hộp thoại ghi/đọc tập tin
  - *File getSelectedFile():* lấy thông tin về tập tin/thư mục được chọn
  - String *getPath*()
  - String *getName*()

JFileChooser




```
if (returnValue == JFileChooser.APPROVE_OPTION)
```

JFileChooser

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

showOpenDialog

```
public int showOpenDialog(Component parent)
                   throws HeadlessException
```

Pops up an "Open File" file chooser dialog. Note that the text that appears in the approve button is determined by the L&F.

Parameters:

`parent` - the parent component of the dialog, can be `null`; see `showDialog` for details

Returns:

the return state of the file chooser on popdown:

- JFileChooser.CANCEL_OPTION
- JFileChooser.APPROVE_OPTION
- JFileCHooser.ERROR_OPTION if an error occurs or the dialog is dismissed

Throws:

`HeadlessException` - if GraphicsEnvironment.isHeadless() returns true.

See Also:

`GraphicsEnvironment.isHeadless()`, `showDialog(java.awt.Component, java.lang.String)`

---

showSaveDialog

```
public int showSaveDialog(Component parent)
                   throws HeadlessException
```

Pops up a "Save File" file chooser dialog. Note that the text that appears in the approve button is determined by the L&F.

Ví dụ JFileChooser

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

```
import java.io.*;       // import JAVA file and stream handling classes
import javax.swing.*;   // import JAVA SWING graphical interface libraries

// This class demonstrates use of JFileChooser
public class SimpleDialog
{
    String fileName;
    File theFile;
    public int fileChooserDialog()
    {
        // Make an instance, call the showOpenDialog method and return result
        JFileChooser fc = new JFileChooser();
        int retval = fc.showOpenDialog(null);
        if(retval==JFileChooser.APPROVE_OPTION){
            theFile=fc.getSelectedFile();
            fileName=fc.getName(theFile);
        }
        return retval;
    }
}
```

Ví dụ JFileChooser

```
import java.io.*;       // import JAVA file and stream handling classes
import javax.swing.*;   // import JAVA SWING graphical interface libraries

// This class demonstrates use of JFileChooser
public class SimpleDialog
{
    String fileName;
    File theFile;
    public int fileChooserDialog()
    {
        // Make an instance, call the showOpenDialog method and return result
        JFileChooser fc = new JFileChooser();
        int retval = fc.showOpenDialog(null);
        if(retval==JFileChooser.APPROVE_OPTION){
            theFile=fc.getSelectedFile();
            fileName=fc.getName(theFile);
        }
        return retval;
    }
}
```

Ví dụ JFileChooser

```
import java.io.*;       // import JAVA file and stream handling classes
import javax.swing.*;   // import JAVA SWING graphical interface libraries

// This class demonstrates use of JFileChooser
public class SimpleDialog
{
    String fileName;
    File theFile;
    public int fileChooserDialog()
    {
        // Make an instance, call the showOpenDialog method and return result
        JFileChooser fc = new JFileChooser();
        int retval = fc.showOpenDialog(null);
        if(retval==JFileChooser.APPROVE_OPTION){
            theFile=fc.getSelectedFile();
            fileName=fc.getName(theFile);
        }
        return retval;
    }
}
```

a `File` object

a `String`

Lọc Tập Tin Hiển Thị

- FileNameExtensionFilter(String dispc, String filter)

  FileNameExtensionFilter filter = new FileNameExtensionFilter("JPG & GIF Images", "jpg", "gif");

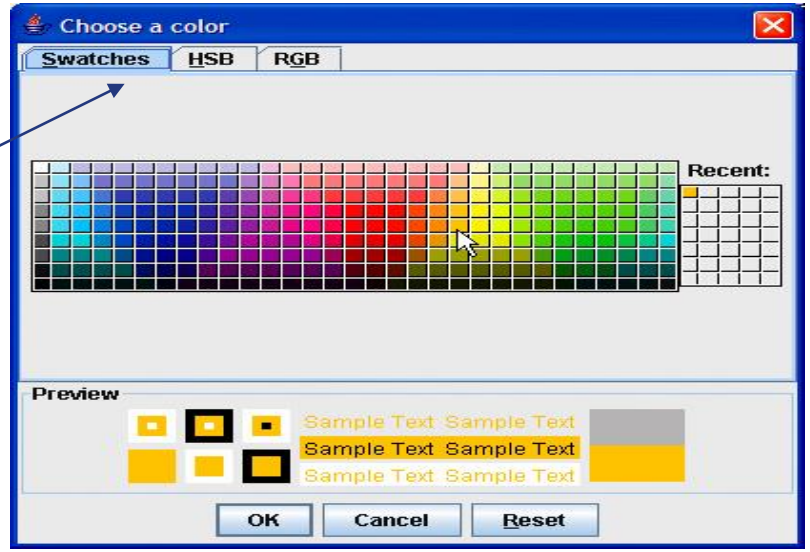
- JFileChooser:

  setFileFilter(FileNameExtensionFilter filter)

JColorChooser

- *static Color showDialog*(Component parent, String title, Color initColor)
  - Mở hộp thoại chọn màu
  - Trả về đối tượng màu đã chọn

Select a color from one of the color swatches.



Tóm tắt bài học

- 3 yếu tố quan trọng trong mô hình xử lý sự kiện: Nguồn phát sinh sự kiện (event source), sự kiện (event object), bộ lắng nghe sự kiện (event listener)
- Các bước xử lý sự kiện
  - Khai báo lớp xử lý sự kiện
  - Cài đặt các phương thức trong listener interface
  - Gắn bộ xử lý vào component
- Event-listener của mouse events gồm: MouseListener và MouseMotionListener
- Xử lý sự kiện bàn phím có interface KeyListener, dùng để xử lý key events (phát sinh khi 1 phím được nhấn và thả ra). KeyEvent chứa virtual key code đại diện cho các phím
- Label dùng để hiển thị một chuỗi văn bản thông thường nhằm mô tả thêm thông tin

Tóm tắt bài học

- Button là một thành phần gây ra một sự kiện hành động khi được kích chuột
- JCheckBox cho phép chọn nhiều lựa chọn, JRadioButton cho phép chọn 1 trong nhiều lựa chọn
- Java cung cấp một số lớp - JMenuBar, JMenu, JMenuItem, JCheckBoxMenuItem, và JRadioButtonMenuItem - để thực thi menu trong một frame
- Dialog Boxes giúp nhận thông tin từ người dùng hoặc hiển thị kết quả/thông báo
- JFileChooser cho phép hiển thị một hộp thoại cho người dùng để chọn một file hay thư mục
- JColorChooser cho phép mở hộp thoại chọn màu, chọn màu và trả về đối tượng màu đã chọn