

LESSON XIII. GUI and Event Programming (2/2)

Trịnh Tuấn Đạt

1

Objectives

- After this lesson, students (learners) can:
 - Create menus inside an AWT application
 - Process action when choosing a menu item
 - Create shortcuts for menu items
 - Create a popup menu when right-clicking on any AWT components
 - Understand Swing's advanced features compared to AWT's
 - Write Swing application

2

Content

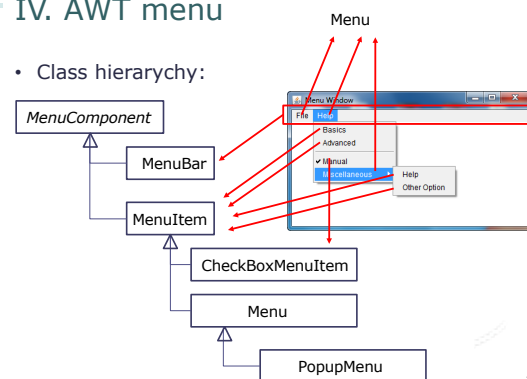
IV. AWT Menu

V. Programming GUI with Swing

3

IV. AWT menu

- Class hierarchy:



4

4.1. Steps to add menus to a Frame

- 1. Create a MenuBar

```
MenuBar mb = new MenuBar();
```
- 2. Create a Menu

```
Menu m = new Menu("File");
```
- 3. Add MenuItem to the menu

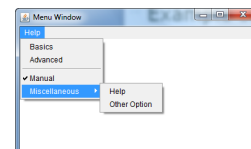
```
m.add(new MenuItem("Open"));
m.add(new CheckboxMenuItem("Type here"));
```
- 4. Add the menu to the MenuBar

```
mb.add(m);
```
- 5. add the MenuBar to the Frame by calling the `setMenuBar()` method

5

Example of a menu-description

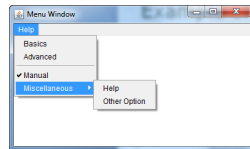
- Application:
 - Create a MenuBar which has
 - A Menu: **Help** which has
 - 2 MenuItem: **Basics, Advanced**
 - A CheckboxMenuItem: **Manual**
 - A Menu: **Miscellaneous** which has
 - 2 MenuItem: **Help, Other Option**
 - Event Handling: if we click on menu item Basics and Help, application prints something to the screen



6

Example of a menu – our Frame class

```
public class MainWindow extends Frame {
    public MainWindow() {
        super("Menu Window");
        setSize(400, 400);
        HelpMenu helpMenu = new HelpMenu();
        MenuBar mb = new MenuBar();
        mb.add(helpMenu);
        setMenuBar(mb);
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                setVisible(false);
                dispose();
                System.exit(0);
            }
        });
    }
    public static void main(String args[]) {
        MainWindow w = new MainWindow();
        w.setVisible(true);
    }
}
```



7

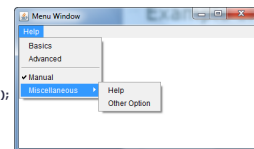
Example of a menu – our Menu class

```
public class HelpMenu extends Menu implements ActionListener {

    public HelpMenu() {
        super("Help");
        MenuItem mi;
        add(mi = new MenuItem("Basics"));
        mi.addActionListener(this);
        add(mi = new MenuItem("Advanced"));
        mi.addActionListener(this);
        addSeparator();
        add(mi = new CheckboxMenuItem("Manual"));
        mi.addActionListener(this);

        Menu subMenu = new Menu("Miscellaneous");
        subMenu.add(mi = new MenuItem("Help"));
        mi.addActionListener(this);
        subMenu.add(mi = new MenuItem("Other Option"));
        mi.addActionListener(this);
        add(subMenu);
    }

    public void actionPerformed(ActionEvent e) {
        String item = e.getActionCommand();
        if (item.equals("Basics"))
            System.out.println("Basics");
        else if (item.equals("Help"))
            System.out.println("Help");
    }
}
```



8

4.2. Menu Shortcuts

- How to quickly invoke a MenuItem?
 - Using Keyboard Shortcut
- When you create a MenuItem, using this constructor to associate it with a keyboard shortcut
`MenuItem(String label, MenuShortcut s)`
- MenuShortcut constructors:
 - `/*Constructs a new MenuShortcut for the specified key*/`
`public MenuShortcut(int key)`
 - `/*Constructs a new MenuShortcut for the specified key*/`
`public MenuShortcut(int key, boolean useShiftModifier)`
 - key: raw key code (each key has one)
 - useShiftModifier: whether this MenuShortcut is invoked with the SHIFT key down (Otherwise, CTRL only)

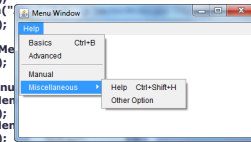
9

Example of Menu shortcuts

- Modify the previous example so that we can access **Basics** menu item with CTRL+B and **Help** menu item with CTRL+SHIFT+H

```
public HelpMenu() {
    super("Help");
    MenuItem mi;
    add(mi = new MenuItem("Basics", new MenuShortcut(KeyEvent.VK_B)));
    mi.addActionListener(this);
    add(mi = new MenuItem("Help", new MenuShortcut(KeyEvent.VK_H, true)));
    mi.addActionListener(this);
    addSeparator();
    add(mi = new MenuItem("Manual", new MenuShortcut(KeyEvent.VK_M)));
    mi.addActionListener(this);

    Menu subMenu = new Menu("Miscellaneous");
    subMenu.add(mi = new MenuItem("Other Option", new MenuShortcut(KeyEvent.VK_O)));
    subMenu.add(mi = new MenuItem("Other Option", new MenuShortcut(KeyEvent.VK_P)));
    subMenu.add(mi = new MenuItem("Other Option", new MenuShortcut(KeyEvent.VK_Q)));
    add(subMenu);
}
```



10

4.3. PopupMenu

- PopupMenu:
 - extends Menu
 - can be add to any Component, using `add (aPopupMenu)`
 - Can be deinstalled from Component, using `remove (aPopupMenu)`
 - is activated when the user holds the right mouse button
- Constructors:
 - `public PopupMenu()`
 - creates an untitled PopupMenu.
 - `public PopupMenu(String label)`
 - creates a PopupMenu with a title of label
 - Once created, the menu can be populated with menu items like any other menu

11

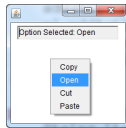
4.3. PopupMenu

- Method to display the PopupMenu
 - `public void show(Component origin, int x, int y)`
 - x, y: location at which the pop-up menu should appear; origin specifies the Component whose coordinate system is used to locate x and y
- How to check whether the popup was triggered by right mouse click?
 - use `isPopupTrigger()` method of `MouseEvent` class.
 - Note: Popup menus are triggered differently on different systems
 - Therefore, `isPopupTrigger` should be checked in both `mousePressed` and `mouseReleased`

12

4.3. Popup menu Example - Description

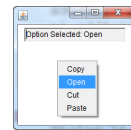
- Application:
 - Has a Popup menu and a textfield
 - When Popup menu is triggered, the selection will be displayed on the textfield



13

4.3. Popup menu Example

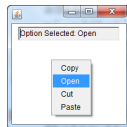
```
public class PopupMenuDemo extends Frame {
    TextField msg; PopupAppMenu m;
    public PopupMenuDemo() {
        setLayout(new FlowLayout());
        msg = new TextField(20);
        msg.setEditable(false); add(msg);
        m = new PopupAppMenu(this); add(m);
        addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
                if (e.isPopupTrigger()) m.show(e.getComponent(), e.getX(), e.getY());
            }
            public void mouseReleased(MouseEvent e) {
                if (e.isPopupTrigger()) m.show(e.getComponent(), e.getX(), e.getY());
            }
        });
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                setVisible(false); dispose(); System.exit(0);
            }
        });
        setSize(200, 200); setVisible(true);
    }
    public static void main(String[] args) {
        PopupMenuDemo app = new PopupMenuDemo();
    }
}
```



14

4.3. Popup menu Example

```
class PopupAppMenu extends PopupMenu implements ActionListener {
    PopupMenuDemo ref;
    public PopupAppMenu(PopupMenuDemo ref) {
        super("File");
        this.ref = ref;
        MenuItem mi;
        add(mi = new MenuItem("Copy"));
        mi.addActionListener(this);
        add(mi = new MenuItem("Open"));
        mi.addActionListener(this);
        add(mi = new MenuItem("Cut"));
        mi.addActionListener(this);
        add(mi = new MenuItem("Paste"));
        mi.addActionListener(this);
    }
    public void actionPerformed(ActionEvent e) {
        String item = e.getActionCommand();
        ref.msg.setText("Option Selected: " + item);
    }
}
```



15

Content

IV. AWT Menu

V. Programming GUI with Swing

16

V. Swing

- 5.1. Introduction
- 5.2. Swing features
- 5.3. Swing API
- 5.4. Sample Swing Application

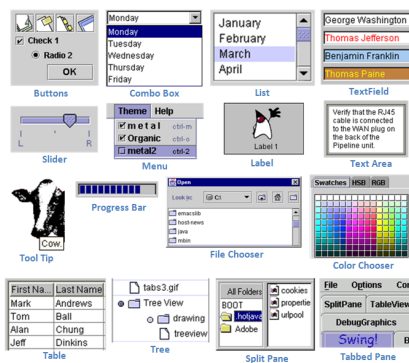
17

5.1. Introduction

- Swing is part of the so-called "Java Foundation Classes (JFC)"
- JFC consists of:
 - Swing API
 - Accessibility API
 - Java 2D API
 - Pluggable look and feel supports.
 - Drag-and-drop support between Java and native applications
- Swing appeared after JDK 1.1
- Swing is a rich set of easy-to-use, easy-to-understand GUI components

18

5.2. Swing features



- Huge:
 - 18 packages

19

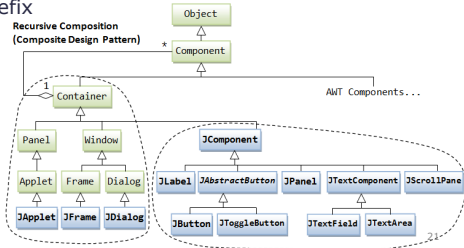
5.2. Swing features

- Written in pure java
- Swing components are lightweight
- Swing components support pluggable look-and-feel
- Swing supports *mouse-less operation*
- Swing components support "tool-tips".
- Swing components are *JavaBeans*
- Swing application uses AWT event-handling classes
- Swing application uses AWT's layout manager
- Swing implements *double-buffering* and automatic repaint batching
- Swing supports floating toolbars (in JToolBar), splitter control, "undo"

20

5.3. Swing API

- Switching AWT programming (container/component, event-handling, layout manager) to Swing is straight-forward
- "J" Prefix



a. Swing's Top-Level and Secondary Containers

- Three top-level containers in Swing:
 - JFrame: used for the application's main window (with an icon, a title, minimize/maximize/close buttons, an optional menu-bar, and a content-pane).
 - JDialog: used for secondary pop-up window (with a title, a close button, and a content-pane).
 - JApplet: used for the applet's display-area (content-pane) inside a browser's window.
- Secondary containers (JPanel)
 - Used to group and layout components

22

b. The Content-Pane of Swing's Top-Level Container

- JComponents shall not be added onto the top-level container (e.g., JFrame, JApplet) directly.
 - JComponents must be added onto the so-called *content-pane* of the top-level container
 - Content-pane: a java.awt.Container, can be used to group and layout components
- Two ways to add JComponent to top-level container:
 - get the content-pane via getContentPane() from a top-level container, and add components onto it
 - set the content-pane to a JPanel (the main panel created in your application which holds all your GUI components) via JFrame's setContentPane()
- Note:** If a component is added directly into a JFrame, it is added into the content-pane of JFrame instead. Inside a JFrame

```
add(new JLabel("add to JFrame directly"));
is executed as
getContentPane().add(new JLabel("add to JFrame directly"));
```

23

Using getContentPane()

```
public class TestGetContentPane extends JFrame {
    public TestGetContentPane() {
        Container cp = this.getContentPane();
        cp.setLayout(new FlowLayout());
        cp.add(new JLabel("Hello, world!"));
        cp.add(new JButton("Button"));
        .....
    }
    .....
}
```

24

Using setContentPane()

```
public class TestSetContentPane extends JFrame {
    public TestSetContentPane() {
        JPanel mainPanel = new JPanel(new FlowLayout());
        mainPanel.add(new JLabel("Hello, world!"));
        mainPanel.add(new JButton("Button"));

        this.setContentPane(mainPanel);
        .....
    }
    .....
}
```

25

c. How to write swing application

- Similar to write awt application
 - Remember prefix "J"
 - Use the Swing components with prefix "J" in package javax.swing
 - Add JComponents to content-pane of the top-level container
 - Event-handling:
 - uses the AWT event-handling classes
 - Swing introduces a few new event-handling classes (in package javax.swing.event) but they are not frequently used.

26

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

public class Template extends JFrame {
    // private variables

    public Template() {
        Container cp = this.getContentPane();
        // cp.setLayout(new ...Layout());
        // adds components

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Exit the program when the close-window button clicked
        setTitle("Some title"); // "this" JFrame sets title
        setSize(300, 150); // "this" JFrame sets initial size (or pack())
        setVisible(true); // show it
    }

    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 Template(); // Let the constructor do the job
            }
        });
    }
}
```

27

d. Swing program template

e. Special notes working with Swing

- JFrame's setDefaultCloseOperation(int operation)
 - to process the "close-window" button without writing a WindowEvent listener, use setDefaultCloseOperation()
 - Operation can be:
 - DO_NOTHING_ON_CLOSE; don't do anything
 - HIDE_ON_CLOSE: Automatically hide the frame
 - DISPOSE_ON_CLOSE: Automatically hide and dispose the frame
 - EXIT_ON_CLOSE: Exit the application using the System.exit() method
 - we choose the option JFrame.EXIT_ON_CLOSE, which terminates the application via a System.exit():
 - setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

28

e. Special notes working with Swing

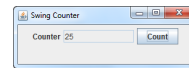
- Running the GUI Construction Codes on the Event-Dispatching Thread
 - We can invoke the constructor directly in the main() method → it is executed in the so-called "Main-Program" thread, causing multi-threading issues (e.g., unresponsive user-interface & deadlock)
 - Recommendation:
 - execute the GUI setup codes in the so-called "Event-Dispatching" thread, for thread-safe operations. To do so, invoke static method `SwingUtilities.invokeLater()`

```
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 Template(); // Let the constructor do the job
        }
    });
}
```

29

5.4. Sample Swing application

- The application includes 3 JComponents:
 - A JLabel
 - A JTextField
 - A JButton
- Whenever users click the count button, a number representing times of clicks is updated in the JTextField



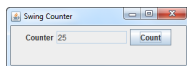
30

5.4. Sample Swing application

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

public class SwingCounter extends JFrame {
    private JTextField tfCount;
    private int count = 0;

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



31

```
public SwingCounter () {
    Container cp = getContentPane();
    cp.setLayout(new FlowLayout());

    cp.add(new JLabel("Counter"));
    tfCount = new JTextField("0", 10);
    tfCount.setEditable(false);
    cp.add(tfCount);

    JButton btnCount = new JButton("Count");
    cp.add(btnCount);

    btnCount.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            count++;
            tfCount.setText(count + "");
        }
    });

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setTitle("Swing Counter");
    setSize(300, 100);
    setVisible(true);
    //end of constructor
} //end of class
```

Quick quiz (1/2)

- Out of all these following classes, which one is root class?
 - MenuItem
 - MenuComponent
 - MenuBar
 - CheckboxMenuItem
 - Menu
 - PopupMenu
- Which command should be used to add MenuBar mb to a Frame fr?
 - fr.add(mb);
 - fr.addMenuBar(mb);
 - fr.setMenuBar(mb);

32

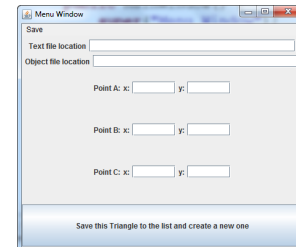
Quick quiz (2/2)

- 3. Which class we can get key raw code from?
 - a. Key
 - b. KeyEvent
 - c. Container
 - d. Component
- 4. Why is `PopupTrigger` should be checked in both `mousePressed` and `mouseReleased`
- 5. What are the top-level containers in Swing?
- 6. Can we add components directly into a `JFrame`?

33

Quiz

- Transform your AWT application in previous lesson in to an Swing application
 - The interface now looks better?



34

Review

- 5 important classes to work with AWT menu: `MenuComponent`, `MenuBar`, `MenuItem`, `CheckboxMenuItem`, `Menu`, `PopupMenu`
- There are four steps to add Menus to a frame: (1) create a `MenuBar`, (2) create a `Menu`, (3) Add `MenuItem` to the `Menu` and (4) add the `MenuBar` to the `Frame`
- Use `MenuShortcut` to associate a `MenuItem` with a keyboard shortcut
- `PopupMenu` can be added to any `Component`

35

Review

- Swing is part of JFC. It is huge with 18 packages
- Switching AWT programming (container/component, event-handling, layout manager) to Swing is straight-forward
- Three top-level containers in Swing are `JFrame`, `JDialog`, `JApplet`
- `JComponents` must be added onto the so-called *content-pane* of the top-level container.
- It is recommended to execute the GUI setup codes in the so-called "Event-Dispatching" thread, instead of "Main-Program" thread, for thread-safe operations.

36