NCP2103: Object-Oriented Programming (Java Programming)

Introduction to `Swing` Components

Module 10

Errol John M. Antonio
Assistant Professor
Computer Engineering Department
University of the East – Manila Campus

Objectives

- Understand `Swing` components
- Use the `JFrame` class
- Use the `JLabel` class
- Use a layout manager
- Extend the `JFrame` class

Objectives (cont'd.)

- Add `JTextFields`, `JButtons`, and tool tips to a `JFrame`
- Learn about event-driven programming
- Understand `Swing` event listeners
- Use the `JCheckBox`, `ButtonGroup`, and `JComboBox` classes

Understanding `Swing` Components

- UI components
  - Buttons, text fields, and other components with which the user can interact
- `Swing` component
  - Descendant of `JComponent`
  - Inherits from `java.awt.Container` class
- Insert import statement:

```
import javax.swing.*;
```

Understanding Swing Components (cont'd.)

- Container
  - Type of component that holds other components
  - Can treat group as single entity
  - Defined in `Container` class
  - Often takes form of window
    - Drag
    - Resize
    - Minimize
    - Restore
    - Close

Understanding Swing Components (cont'd.)

- `Window` class
  - Child of `Container`
  - Does not have title bars or borders
  - Rarely used
  - Instead use subclasses
    - `Frame`
    - `JFrame`

Using the `JFrame` Class

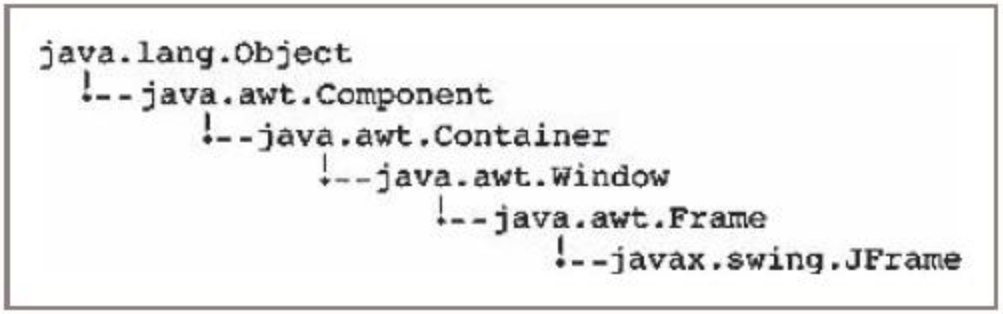

Figure 14-1 Relationship of the `JFrame` class to its ancestors

Using the `JFrame` Class (cont'd.)

- Create `JFrame`
  - Place other objects within it for display
- Constructors

```
JFrame()
JFrame(String title)
JFrame(GraphicsConfiguration gc)
JFrame(String title,
GraphicsConfiguration gc)
```

Method	Purpose
`void setTitle(String)`	Sets a `JFrame`'s title using the `String` argument
`void setSize(int, int)`	Sets a `JFrame`'s size in pixels with the width and height as arguments
`void setSize(Dimension)`	Sets a `JFrame`'s size using a `Dimension` class object; the `Dimension(int, int)` constructor creates an object that represents both a width and a height
`String getTitle()`	Returns a `JFrame`'s title
`void setResizable(boolean)`	Sets the `JFrame` to be resizable by passing `true` to the method, or sets the `JFrame` not to be resizable by passing `false` to the method
`boolean isResizable()`	Returns `true` or `false` to indicate whether the `JFrame` is resizable
`void setVisible(boolean)`	Sets a `JFrame` to be visible using the `boolean` argument `true` and invisible using the `boolean` argument `false`
`void setBounds(int, int, int, int)`	Overrides the default behavior for the `JFrame` to be positioned in the upper-left corner of the computer screen's desktop; the first two arguments are the horizontal and vertical positions of the `JFrame`'s upper-left corner on the desktop, and the final two arguments set the width and height

Table 14-1 Useful methods inherited by the `JFrame` class

Using the `JFrame` Class (cont'd.)

- Create `JFrame`

```
JFrame firstFrame = new
  JFrame("Hello");
```

- Set size and title

```
firstFrame.setSize(200, 100);
firstFrame.setTitle("My frame");
```

Using the `JFrame` Class (cont'd.)

```
import javax.swing.*;
public class JFrame1
{
   public static void main(String[] args)
   {
      JFrame aFrame = new JFrame("First frame");
      aFrame.setSize(250, 100);
      aFrame.setVisible(true);
   }
}
```

Figure 14-2 The JFrame1 application

Using the `JFrame` Class (cont'd.)

- Close `JFrame`
  - Click Close button
  - `JFrame` becomes hidden and application keeps running
    - Default behavior
  - To change this behavior
    - Use `setDefaultCloseOperation()` method

Customizing a `JFrame`'s Appearance

- Window decorations
  - Icon and buttons
- Look and feel
  - Default appearance and behavior of user interface
  - `setDefaultLookAndFeelDecorated()` method
    - Set `JFrame`'s look and feel

Customizing a JFrame's Appearance (cont'd.)

```
import javax.swing.*;
public class JFrame2
{
   public static void main(String[] args)
   {
      JFrame.setDefaultLookAndFeelDecorated(true);
      JFrame aFrame = new JFrame("Second frame");
      aFrame.setSize(250, 100);
      aFrame.setVisible(true);
   }
}
```

Figure 14-4 The JFrame2 class

Using a `JLabel`

- `JLabel`
  - Holds text you can display
  - Available constructors
- Methods
  - `add()`
  - `remove()`
  - `setText()`
  - `getText()`

Changing a `JLabel`'s Font

- `Font` class
  - Creates an object that holds typeface and size information
  - To construct a `Font` object
    - Arguments: typeface, style, and point size
- `setFont()` method
  - Requires a `Font` object argument

Changing a JLabel's Font (cont'd.)

```
import javax.swing.*;
import java.awt.*;
public class JFrame4
{
   public static void main(String[] args)
   {
      final int FRAME_WIDTH = 250;
      final int FRAME_HEIGHT = 100;
      Font headlineFont = new Font("Arial", Font.BOLD, 36);
      JFrame aFrame = new JFrame("Fourth frame");
      aFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
      aFrame.setVisible(true);
      aFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JLabel greeting = new JLabel("Good day");
      greeting.setFont(headlineFont);
      aFrame.add(greeting);
   }
}
```

Figure 14-9 The JFrame4 program

Using a Layout Manager

- Layout manager
  - Class that controls component positioning
- `BorderLayout`
  - Normal (default) behavior of a `JFrame`
  - Divides a container into regions
- `FlowLayout`
  - Places components in a row

Extending the `JFrame` Class

- Create class that descends from `JFrame` class
- Advantage
  - Set `JFrame`'s properties within object's constructor
  - When `JFrame` child object created
    - Automatically endowed with specified features
- Create child class using keyword `extends`
- Call parent class's constructor method
  - Using keyword `super`

Extending the `JFrame` Class (cont'd.)

```
import javax.swing.*;
public class JMyFrame extends JFrame
{
   final int WIDTH = 200;
   final int HEIGHT = 120;
   public JMyFrame()
   {
      super("My frame");
      setSize(WIDTH, HEIGHT);
      setVisible(true);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

Figure 14-15 The JMyFrame class

Adding `JTextFields`, `JButtons`, and Tool Tips to a `JFrame`

- In addition to including `JLabel` objects
  - `JFrame`s often contain other window features, such as `JTextField`s, `JButton`s, and tool tips

Adding `JTextFields`

- `JTextField`
  - Component into which a user can type a single line of text data
  - Several constructors
  - Methods
    - `setText()`
    - `getText()`
    - `setEditable()`

Adding `JButtons`

- `JButton`
  - Click with a mouse to make a selection
  - Five constructors
  - Methods
    - `setText()`
    - `getText()`
- `add()` method
  - Adds a `JButton` to a `JFrame`

Adding `JButtons` (cont'd.)

- When clicked
  - No resulting actions occur
  - Code has not yet been written to handle user-initiated events

Using Tool Tips

- Tool tips
  - Popup windows
  - Help a user understand the purpose of components in an application
  - Appear when a user hovers the mouse pointer over the component
- `setToolTipText()` method
  - Set tool tip for a `Component`

Learning about Event-Driven Programming

- Event
  - Occurs when a user takes action on a component, such as clicking the mouse on a `JButton` object
- Event-driven program
  - User might initiate any number of events in any order
- Source
  - Component on which an event is generated
- Listener
  - Object that is interested in an event

Learning about Event-Driven Programming (cont'd.)

- Respond to user events within any class you create
  - Prepare your class to accept event messages
  - Tell your class to expect events to happen
  - Tell your class how to respond to events

Preparing Your Class to Accept Event Messages

- Import the `java.awt.event` package
- Add the phrase `implements ActionListener` to the class header
- `ActionListener`
  - Standard event method specifications that allow your listener to work with `ActionEvents`

Telling Your Class to Expect Events to Happen

- `addActionListener()` method
- `aButton.addActionListener(this);`
  - Causes any `ActionEvent` messages (button clicks) that come from `aButton` to be sent to "this current object"

Telling Your Class How to Respond to Events

- `ActionListener` interface
  - `actionPerformed(ActionEvent e)` method specification
    - Body contains any statements that you want to execute when the action occurs
- When more than one component is added and registered to a `JFrame`
  - Necessary to determine which component was used
  - Find source of the event using `getSource();`

Using the `setEnabled()` Method

- `setEnabled()` method
  - Make a component unavailable
  - Then make it available again in turn
- Use after a specific series of actions has taken place

Understanding `Swing` Event Listeners

- Classes that respond to user-initiated events
  - Must implement interface that deals with events
  - Called event listeners
- Many types of listeners exist in Java
  - Each can handle specific event type
- Class can implement as many event listeners as it needs
- Event occurs every time user types character or clicks mouse button

Understanding `Swing` Event Listeners (cont'd.)

Listener	Type of Events	Example
`ActionListener`	Action events	Button clicks
`AdjustmentListener`	Adjustment events	Scroll bar moves
`ChangeListener`	Change events	Slider is repositioned
`FocusListener`	Keyboard focus events	Text field gains or loses focus
`ItemListener`	Item events	Check box changes status
`KeyListener`	Keyboard events	Text is entered
`MouseListener`	Mouse events	Mouse clicks
`MouseMotionListener`	Mouse movement events	Mouse rolls
`WindowListener`	Window events	Window closes

Table 14-2 Alphabetical list of some event listeners

Understanding `Swing` Event Listeners (cont'd.)

- Create relationships between `Swing` components and classes that react to users' manipulations of them
- `JCheckBox` responds to user's clicks
  - `addItemListener()` method
  - Register `JCheckBox` as type of object that can create `ItemEvent`
  - Format:

```
theSourceOfTheEvent.addListenerMethod
(theClassThatShouldRespond);
```

Understanding `Swing` Event Listeners (cont'd.)

Component(s)	Associated Listener-Registering Method(s)
`JButton`, `JCheckBox`, `JComboBox`, `JTextField`, and `JRadioButton`	`addActionListener()`
`JScrollBar`	`addAdjustmentListener()`
All `Swing` components	`addFocusListener()`, `addKeyListener()`, `addMouseListener()`, and `addMouseMotionListener()`
`JButton`, `JCheckBox`, `JComboBox`, and `JRadioButton`	`addItemListener()`
All `JWindow` and `JFrame` components	`addWindowListener()`
`JSlider` and `JCheckBox`	`addChangeListener()`

Table 14-3 Some `Swing` components and their associated listener-registering methods

Understanding `Swing` Event Listeners (cont'd.)

- Class of object that responds to event
  - Contains method that accepts event object created by user's action
  - Specific methods react to specific event types
- Declare class that handles event
  - Create class to either:
    - Implement listener interface
    - Extend class that implements listener interface
- Declare a class that extends `MyFrame`
  - Need not include `implements ItemListener` in header

Understanding `Swing` Event Listeners (cont'd.)

- Register instance of event handler class as listener for one or more components

Using the `JCheckBox`, `ButtonGroup`, and `JComboBox` Classes

- Besides `JButton`s and `JTextField`s
  - Several other Java components allow a user to make selections in a UI environment

The `JCheckBox` Class

- `JCheckBox`
  - Consists of a label positioned beside a square
  - Click square to display or remove check mark
  - Use to allow user to turn option on or off
- Constructors

```
JCheckBox()
JCheckBox("Check here")
JCheckBox("Check here", false)
```

The JCheckBox Class (cont'd.)

Figure 14-31 Output of the CheckBoxDemonstration class

The JCheckBox Class (cont'd.)

Method	Purpose
`void setText(String)`	Sets the text for the `JCheckBox`
`String getText()`	Returns the `JCheckBox` text
`void setSelected(boolean)`	Sets the state of the `JCheckBox` to `true` for selected or `false` for unselected
`boolean isSelected()`	Gets the current state (checked or unchecked) of the `JCheckBox`

Table 14-5 Frequently used `JCheckBox` methods

The `JCheckBox` Class (cont'd.)

- Methods
  - `setText`
  - `setSelected`
  - `isSelected`
- Status of `JCheckBox` changes from unchecked to checked
  - `ItemEvent` generated
  - `itemStateChanged()` method executes

The `ButtonGroup` Class

- `ButtonGroup`
  - Group several components so that user can select only one at a time
- Group `JCheckBox` objects
  - All of other `JCheckBox`es automatically turned off when user selects any one check box

Using the `ButtonGroup` Class (cont'd.)

- Create `ButtonGroup` and then add `JCheckBox`
  - Create `ButtonGroup`

```
ButtonGroup aGroup = new
ButtonGroup();
```

  - Create `JCheckBox`

```
JCheckBox aBox = new JCheckBox();
```

  - Add `aBox` to `aGroup`

```
aGroup.add(aBox);
```

Using the `JComboBox` Class

- `JComboBox`
  - Component that combines two features
    - Display area showing one option
    - List box containing additional options
  - When user clicks `JComboBox`, list of alternative items drops down
    - User selects one to replace box's displayed item

Using the JComboBox Class (cont'd.)

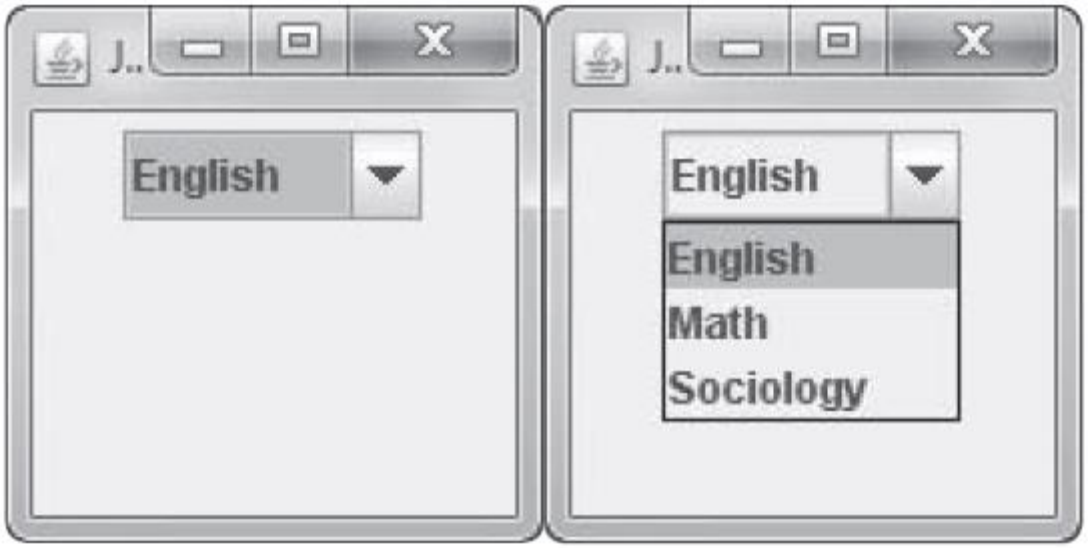

Figure 14-35 A JComboBox before and after the user clicks it

Using the JComboBox Class (cont'd.)

- Build JComboBox
  - Use constructor with no arguments
  - Add items with addItem() method
  - Alternatively, construct by using array of Objects as constructor argument

```
String[] majorArray = {"English",
 "Math", "Sociology"};
JComboBox majorChoice = new
 JComboBox(majorArray);
```

Method	Purpose
`void addItem(Object)`	Adds an item to the list
`void removeItem(Object)`	Removes an item from the list
`void removeAllItems()`	Removes all items from the list
`Object getItemAt(int)`	Returns the list item at the index position specified by the integer argument
`int getItemCount()`	Returns the number of items in the list
`int getMaximumRowCount()`	Returns the maximum number of items the combo box can display without a scroll bar
`int getSelectedIndex()`	Returns the position of the currently selected item
`Object getSelectedItem()`	Returns the currently selected item
`Object[] getSelectedObjects()`	Returns an array containing selected `Objects`
`void setEditable(boolean)`	Sets the field to be editable or not editable
`void setMaximumRowCount(int)`	Sets the number of rows in the combo box that can be displayed at one time
`void setSelectedIndex(int)`	Sets the index at the position indicated by the argument
`void setSelectedItem(Object)`	Sets the selected item in the combo box display area to be the `Object` argument

Table 14-6 Some `JComboBox` class methods

Using the `JComboBox` Class (cont'd.)

- `setSelectedItem()` or `setSelectedIndex()` method
  - Choose one item in `JComboBox` to be selected item
- `getSelectedItem()` or `getSelectedIndex()` method
  - Discover which item currently selected
- Treat list of items in `JComboBox` object as array
  - First item at position 0
  - Second at position 1

You Do It

- Creating a `JFrame`
- Ending an application when a `JFrame` closes
- Adding components to a `JFrame`
- Adding functionality to a `JButton` and a `JTextField`
- Distinguishing event sources
- Including `JCheckBox`es in an application

Don't Do It

- Don't forget the "x" in `javax` when you import `Swing` components into an application
- Don't forget to use a `JFrame`'s `setVisible()` method if you want the `JFrame` to be visible
- Don't forget to use `setLayout()` when you add multiple components to a `JFrame`

Don’t Do It (cont'd.)

- Don’t forget to call `validate()` and `repaint()` after you add or remove a component from a container that has been made visible
- Don’t forget that the `ButtonGroup` class does not begin with a “J”

Summary

- `JFrame`
  - `Swing` container that resembles a `Window`
  - Has title bar and borders and ability to be resized, minimized, restored, and closed
- Many types of listeners exist in Java
  - Each can handle specific event type
  - Register listener with event source
  - Handle event in event-handling method

Summary (cont'd.)

- `JCheckBox`
  - Consists of a label positioned beside a square
- `ButtonGroup`
  - Group several components so user can select only one at a time
- `JComboBox`
  - Display area showing an option combined with list box containing additional options

End of Module.

REFERENCE:

Farrell, J. (2016). ***Java Programming***. 8th Edition. Course Technology, Cengage Learning.