Unit VII

Working with Graphics User Interface (AWT Controls)

Tutor: Pema Galey

Learning Outcomes

In this session, you will learn about:

- Window Fundamentals
- Work with Abstract Window Toolkit (AWT) control components
- Create Swing components
- Use layout managers

AWT Control Fundamentals

- The AWT supports the following types of control:
  1. Labels
  2. Push buttons
  3. Check boxes
  4. Choice lists
  5. Lists
  6. Scroll bars
  7. Text Editing

All AWT controls are subclasses of **Component**.

AWT Control Components

Using various AWT components:

- An AWT control is a component that enables end user to interact with applications created in Java.
- All AWT controls in Java are subclasses of the Component class.
- The various classes that you can use for creating AWT controls are:
  - `Label`
  - `Button`
  - `List`
  - `CheckBox`
  - `CheckboxGroup`
  - `Choice`
  - `TextField`
  - `TextArea`
  - `Scrollbar`

AWT Control: Labels

The `Label` class:

- Labels are used for displaying a single line of text in a container.
- Java provides the `Label` class to create a label.
- The following list describes the various constructors that you can use to create an instance of the `Label` class:
  - `public Label()`
  - `public Label(String labelText)`
  - `public Label(String labelText, int align)`

AWT Control: Buttons

The `Button` class:

- Buttons are AWT controls that are used for handling events. Java provides the `Button` class to create AWT button components.
- The following list describes the various constructors that you can use to create an instance of the `Button` class:
  - `public Button()`
  - `public Button(String label)`
- The following list describes some of the methods of the Button class:
  - `public String getLabel()`
  - `public void setLabel(String label)`

AWT Control: Check Boxes

The `CheckBox` class:

- Check boxes are the components that exist in dual state, checked and unchecked.
- Java provides the `CheckBox` class to create a check box.
- The following list describes some of the constructors that you can use to create an instance of the `CheckBox` class:
  - `public CheckBox()`
  - `public CheckBox(String labelText)`
  - `public CheckBox(String labelText, boolean state)`
  - `public CheckBox(String labelText, CheckBoxGroup group, boolean state)`

AWT Control: Check Boxes

The following table describes some of the methods of the `CheckBox` class.

Method	***Description***
`public String getLabel()`	*Returns the label of the check box.*
`public void setLabel(String label)`	*Sets the specified label for a check box.*
`public boolean getState()`	*Retrieves the state of the check box.*
`public void setState(boolean state)`	*Sets the specified state for a check box.*
`public CheckBoxGroup getCheckBoxGroup()`	*Retrieves the check box group for the check box.*
`public void setCheckBoxGroup(CheckBoxGroup g)`	*Sets the specified check box group for a check box.*

AWT Control: CheckboxGroup

The `CheckboxGroup` class:

- It is possible to create a set mutually exclusive check boxes in which one and only one check box in the group can be checked at one time.
- These check boxes are often called *radio buttons*.

AWT Control: Choice

- The `Choice` class:
  - The Choice class is used to create a pop-up list of items from which the user may choose. Thus, a Choice control is a form of menu.
  - The following list describes some of the constructors that you can use to create an instance of the `Choice` class:

```
public Choice()
```

AWT Control: Choice

- The `Choice` class: The methods of the `Choice` class.

Method	***Description***
`public int getItemCount()`	*Returns the total number of items of the choice menu.*
`public String getItem(int index)`	*Returns the `String` at the specified index of the choice menu.*
`public void insert(String item, int index)`	*Inserts an item into a choice menu at the specified position.*
`public void remove(String item)`	*Removes the first occurrence of an item from a choice menu.*
`public void remove(int position)`	*Removes an item at the specified position from a choice menu.*
`public void removeAll()`	*Removes all the items from a choice menu.*
`public String getSelectedItem()`	*Returns the current selected item of a choice menu in the form of a string.*
`public int getSelectedIndex()`	*Returns the index of the currently selected item of a choice menu.*
`public void select(int pos)`	*Selects the text specified at the position by the `pos` argument.*

AWT Control: Lists

The `List` class:

- The list control is a scrollable list of text items that enables you to select either one item or multiple items.
- Java provides the `List` class to create the list control.
- The following list describes some of the constructors that you can use to create an instance of the `List` class:
  - `public List()`
  - `public List(int rows, boolean multipleselection)`

AWT Control: Lists

The following table describes some of the methods of the `List` class.

Method	***Description***
`public void add(String item)`	*Adds the specified item at the end of a list.*
`public void add(String item, int index)`	*Adds the specified item at the defined position in a list.*
`public boolean removeall (Collection c)`	*Removes all the items of a list that are contained in the specified collection.*
`public int getItemCount()`	*Retrieves the number of items from a list.*
`public boolean remove(Object o)`	*Removes the first occurrence of the specified item that is passed as an argument.*
`public Object remove (int position)`	*Removes the specified item from this scrolling list.*
`public int getMinimumSize(int rows)`	*Retrieves the minimum dimensions for a list with the specified number of rows.*
`public int getSelectedIndex()`	*Retrieves the index of the selected item from a list.*
`public int getSelectedItem()`	*Retrieves the selected item from a list.*
`public String getItem(int index)`	*Retrieves an item of the scrolling list at the specified index position of a list.*

AWT Control: Lists

The following table describes some of the methods of the `List` class.

`public void replaceItem(String newstring, int index)`	*Replaces an item at the specified index position of a list.*
`public void setMultipleMode(boolean b)`	*Sets the flag that allows multiple selections in a scrolling list.*

AWT Control: Text Editing

The `TextField` class:

- Text fields are user interface components that accept text input from an end user.
- A text field enables you to type text in a single line. An instance of the `TextField` class is used to create a text field.
- The following list describes the various constructors that you can use to create an instance of the `TextField` class:
  - `public TextField()`
  - `public TextField(int cols)`
  - `public TextField(String text)`
  - `public TextField(String text, int cols)`

AWT Control: Text Editing

The following lists represents some of the methods of the `TextField` class:

- `public int getColumns()`
- `public String getText()`
- `public void setText(String text)`

AWT Control: Text Editing

The `TextArea` class:

- Text areas are used to accept multiple lines of text input from an end user.
- An instance of the `TextArea` class is used to create a text area.

- The following list describes various constructors that you can use to create an instance of the `TextArea` class.
  - `public TextArea()`
  - `public TextArea(int rows, int cols)`
  - `public TextArea(String text)`
  - `public TextArea(String text, int rows, int cols, int scrollbar)`

AWT Control: Text Editing

- The following table describes some of the methods of the **`TextArea`** class.

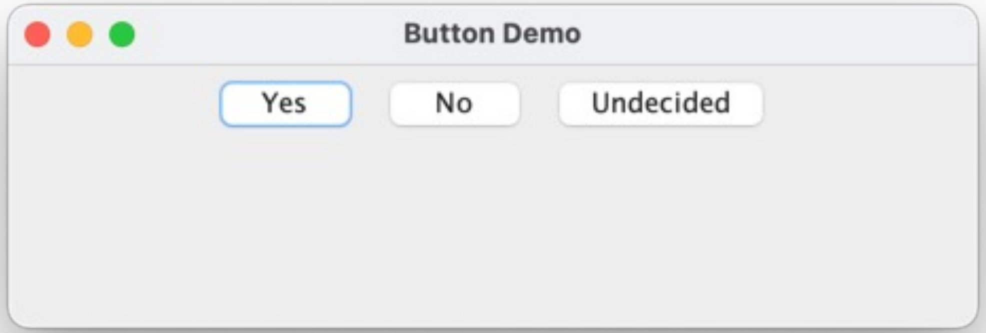
Method	***Description***
`public int getColumns()`	*Returns the number of columns of the text area.*
`public String getText()`	*Returns the text contained in the text area.*
`public void setText(String text)`	*Sets the required text to the text area.*
`public int getRows()`	*Retrieves the total number of rows of the text area.*
`public void insert(String text, in pos)`	*Inserts the required text at the specified position in the text area.*
`public void append(String text)`	*Appends the required text to the current text in the text area.*

Example

```
import java.awt.*;
import java.awt.event.*;
public class ButtonDemo extends Frame{
    Button yes, no, maybe;
    public ButtonDemo(){
        // Use a flow layout
        setLayout(new FlowLayout());
        yes=new Button("Yes");
        no=new Button("No");
        maybe=new Button("Undecided");
        // Add Buttons to frame
        add(yes);
        add(no);
        add(maybe);

        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                System.exit(0);
            }
        });
    }
```

```
    public static void main(String arg[]){
        ButtonDemo ob=new ButtonDemo();
        ob.setSize(new Dimension(250,150));
        ob.setTitle("Button Demo");
        ob.setVisible(true);
    }
}
```



Lab Work

- Design a registration form by including following information:
  - Name (TextField)
  - Student_ID (TextField)
  - Qualification (ChecBox)
  - Programme (CheckboxGroup)
  - Semester (List)
  - Address (TextArea)
  - Submit Button

- Note: Use label for each items except for Button

Thank you!