COP 2251 – Chapter 16 – JavaFX UI Controls & Multimedia

Introduction

* This chapter covers JavaFX controls in GUIs. Get the zipped examples from MyCourses.
* See Figure 16.1 on page 644 for the inheritance hierarchy of these controls.

Labeled and Label

* A Label is a control for short text, often used to show the purpose of other controls.
* A label can display a graphic and can be used for output, too.
* Label inherits from class **Labeled**. See the UML diagrams for both classes on page 645.

Examine and try **LabelWithGraphic.java** in the example zip. Be sure to enlarge the GUI window.

Button

* Classes **ButtonBase** and **Labeled** define properties common to the various button types.
* Class Labeled defines properties common to Label and the button types.
* A Button is like a Label except that it has an **onAction** property for setting a handler.

Examine and try **ButtonDemo.java** in the example zip.

Note that the scene is created by running method getPane() inside the start() method.

CheckBox

* CheckBox inherits the same properties as Button, but also has a **selected** property.
* Checking or unchecking fires an **ActionEvent**.
* The **isSelected()** method is used to determine if the CheckBox is in a checked state.

Examine and try **CheckBoxDemo.java** in the example zip. Note that this example extends class ButtonDemo and also the multiple options inside the handler.

RadioButton

* These buttons are used in GUIs where a user can select just one of a group of RadioButtons.
* This class inherits from class **ToggleButton**. See both classes’ UML on page 651
* RadioButtons are made mutually exclusive by putting them in a ToggleGroup. This is accomplished with the setToggleGroup() method. See lines 24-27 on page 653.
* Changing a RadioButton fires an **ActionEvent**. The status can be determined with **isSelected()**.

Examine and try **RadioButtonDemo.java** in the example zip. It extends CheckBoxDemo.

TextField

* This control is used to enter or display a string. It subclasses **TextInputControl**.
* For display purposes, it is best to **setEditable(false)** to prevent input.
* TextField has a subclass named **PasswordField** for um…passwords.
* See UML on page 654.

Examine and try **TextFieldDemo.java** in the example zip. It extends RadioButtondemo.

TextArea

* A TextArea is for entering multiple lines of text in a GUI. It also subclasses **TextInputControl**.
* A TextArea is often held by a **ScrollPane** to facilitate scrolling.
* See the code snippet on page 656 for how to put a TextArea into a ScrollPane.
* See UML on page 656.
* The text property’s value will be displayed in the TextArea. You can write to the TextArea with setText() and read the TextArea with getText().
* To write successive lines of code to a TextArea, you need to use:

taName.setText(taName.getText() + “your new string” + “\n”);

Examine **DescriptionPane.java** and try **TextAreaDemo.java** in the example zip.

ComboBox

* This is a generic class for displaying a list of objects. It subclasses class **ComboBoxBase**.
* The items to be displayed are stored in an **ObservableList**.
* Selecting an item from the list fires an ActionEvent.
* See UML on page 659.

Examine and try **ComboBoxDemo.java** in the example zip.

ListView

* This generic control is like a ComboBox except that single or multiple items can be selected.
* See UML on page 662.
* The **selectionModel** property has SelectionMode.SINGLE and SelectionMode.MULTIPLE constants to set the selection mode. See page 662 and Figure 16.19.

Examine and try **ListViewDemo.java** in the example zip. Try Ctrl + Mouse to select multiple countries from the list.

ScrollBar

* This control allows users to select from a range of values by scrolling.
* The orientation property can be set to Orientation.VERTICAL or Orientation.HORIZONTAL (the default). See Figure 16.21 on page 665.
* See UML diagram on page 665 for more properties.

Examine and try **ScrollBarDemo.java** in the example zip. Manipulate both scrollbars.

Slider

* A Slider is a like a ScrollBar but with more features.
* Both major and minor tick marks can be displayed to the user.
* A horizontal slider increases from left to right. A vertical slider **decreases** from top to bottom.
* See the UML on page 668.

Examine and try **SliderDemo.java** in the example zip. A slider’s value property can be used for input. Overwrite the slHorizontal lambda expression as follows…

slHorizontal.valueProperty().addListener(ov ->

System.***out***.println(slHorizontal.getValue()));

text.setX(slHorizontal.getValue() \* paneForText.getWidth() /

slHorizontal.getMax());

**BounceBallSlider.java** demonstrates this, too. Try it. It’s in the example zip.

Case Study: Tic-Tac-Toe Game

* Read secton 6.12.

Examine and try **TicTacToe.java** in the example zip.

Video and Audio

* Read pages 676-677 and note the UML for classes Media, MediaPlayer, and MediaView.

Examine and try **MediaDemo.java** in the example zip.

Case Study: National Flags and Anthems

* Read page 679.

Examine and try **FlagAnthem.java** in the example zip. The audio NO LONGER works!

Note how both the image filename and mp3 filename are constructed with concatenation.