GUI Application (2)

Han-fen Hu

UNLV

Outline

- Introduction to FXML
- Using Scene Builder to Create UI
- Writing the Application Code
- Some More Controls in JavaFX
  - RadioButton
  - CheckBox
  - ComboBox
  - TextArea

FXML

- FXML is a scriptable, XML-based markup language for constructing Java object graphs.
  - It is ideally suited to defining the user interface of a JavaFX application
- FXML describes the components in a JavaFX scene graph
  - FXML uses tags to organize data, in a manner similar to the way that HTML uses tags to format text in a Web browser.

source: https://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html

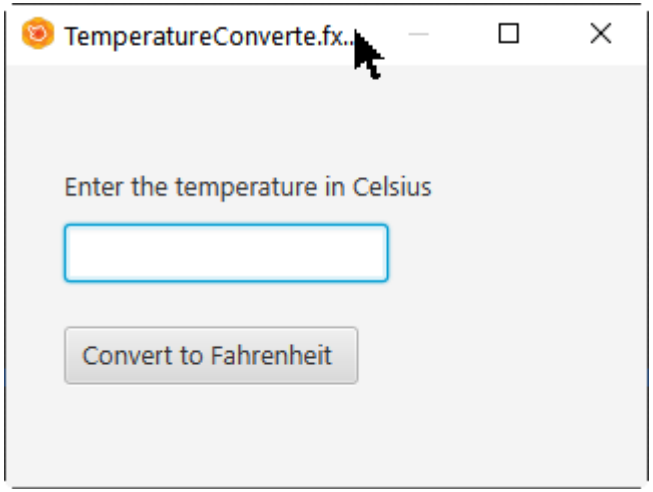
Using Scene Builder to Create UI

- Free , open-source design tool
- Generates the FXML source code as you define the user interface for your application
  - Dragging and dropping the components that you need onto a blank window
  - Visually arrange the components on the window and set various component properties to create the appearance that you want for the GUI
  - Later the UI components can be linked to the application logic

UNLV

Lab (1)

- TemperatureConverter.fxml



Writing the Application Code

- Once you have saved an application's GUI to an FXML file, you can write the Java code that runs the application.
- A simple JavaFX application uses:
  - a main application class
  - a controller class

The Main Application Class (1)

- Once you have created a GUI with Scene Builder, and saved it to an FXML file, you need to write a Java class that performs the following:
  - Loads the FXML file
  - Builds the scene graph in memory
  - Displays the GUI

The Main Application Class (2)

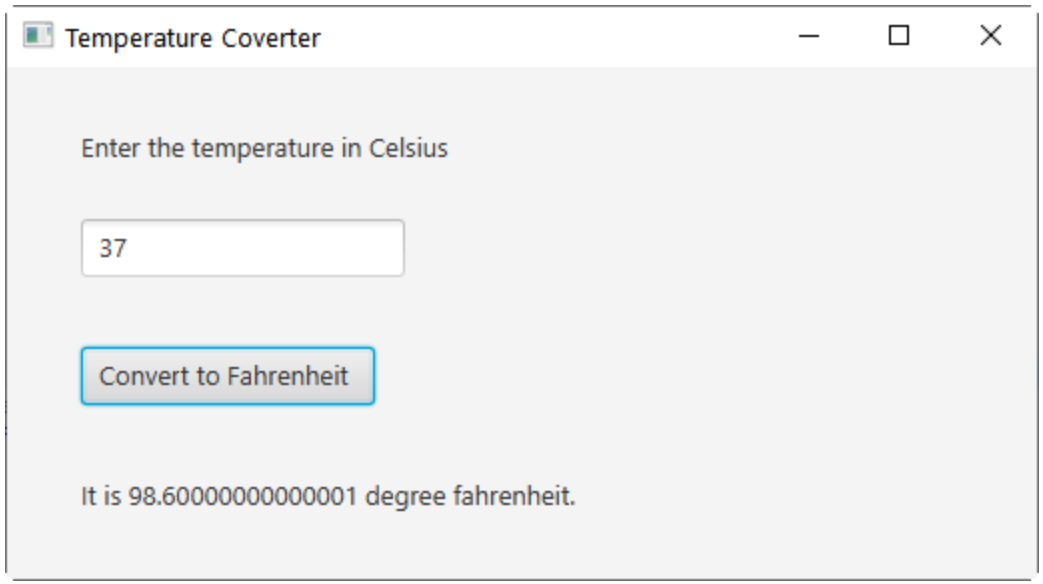
- The main class extends from `Application` and contains two methods.
  - `start()` method is automatically called when the application is launched from within the `main` method
  - `start()`receives a `Stage` as parameter

The Controller Class

- The *controller class* is responsible for handling events that occur while the application is running.
- The controller class contains the following items:
  - The necessary import statements
  - Private variables to reference the components that have a `fx:id` in the scene graph
  - An optional initialize method that is automatically called after the FXML file is loaded
  - Event listener methods

Lab (2)

- TemperatureCoverter.java
- TemperatureCoverterController.java



Why Use FXML

- Easy for a development team to create and maintain a testable user interface
  - FXML is not a compiled language; you do not need to recompile the code to see the changes
- The content of an FXML file can be localized as the file is read
- You can use JavaScript or other scripting languages in FXML

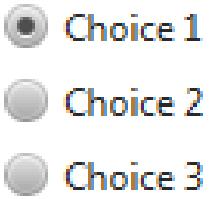
UNLV

Steps of Creating a JavaFX Application with FXML

1. Use Scene Builder to design the GUI. Save the GUI as an FXML file.
   - Be sure to give an `fx:id` to all of the components that you plan to access in your Java code.
2. Write the code for the main application class
   - Loads the FXML file and launches the application.
3. Write the code for the controller class
   - Event handler methods for the GUI.
4. In Scene Builder, register the controller class, then register an event handler method for each component that needs to respond to events.

UNLV

RadioButton (1)



- Behavior
  - Allow the user to select one of several possible options
  - Have a Text property that determines the text they display.
- Normally are in a toggle group.
  - Only one of the `RadioButton` controls in a toggle group may be selected at any time
  - Clicking on a `RadioButton` selects it and automatically deselects any other `RadioButton` in the same toggle group
  - You usually want one of the `RadioButtons` in a group to be initially selected.

RadioButton (2)

- If action needs to take place at the time the user clicks a `RadioButton`.
  - Write event listener methods in the controller class
  - Register the event listener in Scene Builder
- In the controller class, you can use the `RadioButton's isSelected` method to determine whether the `RadioButton` is selected or not.

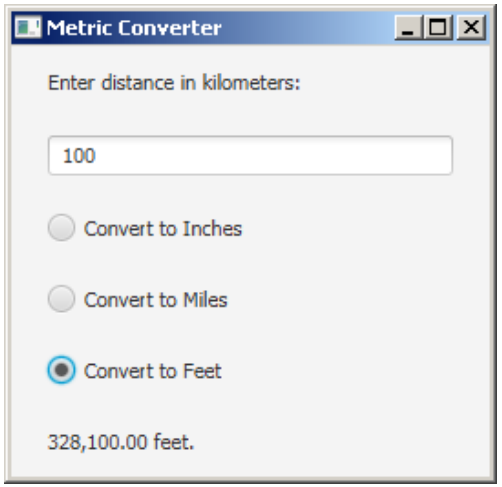
RadioButton (3)

- The `isSelected` method returns a boolean value.
  - If the `RadioButton` is selected, the method returns true . Otherwise, it returns false .

```
if (radio.isSelected()){
    // Code here executes if the radio
    // button is selected.
}
```

Lab (3)

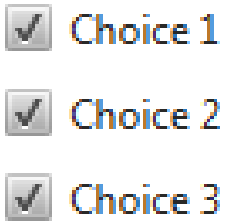
- MetricConverter.fxml
- MetricConverter.java
- MetricConverterController.java



Console Output for GUI Applications

- Be cautious about using console output for GUI Applications
  - End user would not see the console when they run a GUI application
  - Users expect to see results and enter values in GUI only
- Use console output for debugging only
  - Print variable values for debugging
  - Print exception messages for debugging

CheckBox (1)



- Behavior
  - Allow the user to make yes/no or on/off selections
  - Text property that determines the text they display
- Sometimes you want an action to take place at the time the user clicks a CheckBox.
  - you must write an event listener method in the controller class for the CheckBox
  - and then select the method as the event listener in Scene Builder.

CheckBox (2)

- In the controller class, you can use the CheckBox's `isSelected` method to determine whether the CheckBox is selected or not.
- The `isSelected` method returns a boolean value.
  - If the CheckBox is selected, the method returns `true` . Otherwise, it returns `false`

```
if (checkbox.isSelected()) {
    // Code here executes if the
    // CheckBox is selected.
}
```

CheckBox (3)

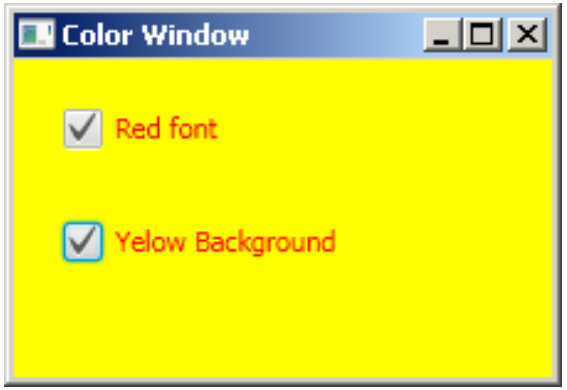
- You can select a `CheckBox` in code with the `CheckBox` class's `setSelected` method:

```
check1.setSelected(true);
```

- If you want an action to take place immediately when the user clicks a `CheckBox`, register an event handler with the `CheckBox` control.

Lab (4)

- ColorWindow.fxml
- ColorWindow.java
- ColorWindowController.java

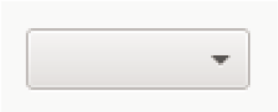


ComboBox (1)

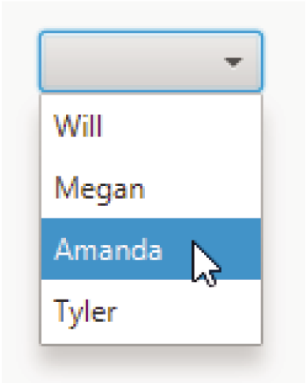
- A ComboBox presents a drop-down list of items that the user may select from.

```
ComboBox<String> nameComboBox = new ComboBox<>();
nameComboBox.getItems().addAll("Will",
"Megan", "Amanda", "Tyler");
```

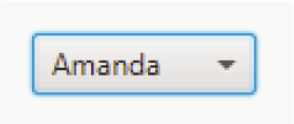
The ComboBox's initial appearance



When the user clicks the ComboBox, a list of items drops down.



The selected item is displayed by the ComboBox.



ComboBox (2)

- You can use the `ComboBox` class's `getValue()` method get the item that is currently selected:

```
String selected = comboBox.getValue();
```

- If no item is selected in the `ComboBox`, the method will return null.
- By default, `ComboBox` controls are uneditable
  - Use the `setEditable` method to make a `ComboBox` editable:

ComboBox (3)

- In FXML design, the items of a CmoboBox need to be added at the Controller class, **`initialized`** method

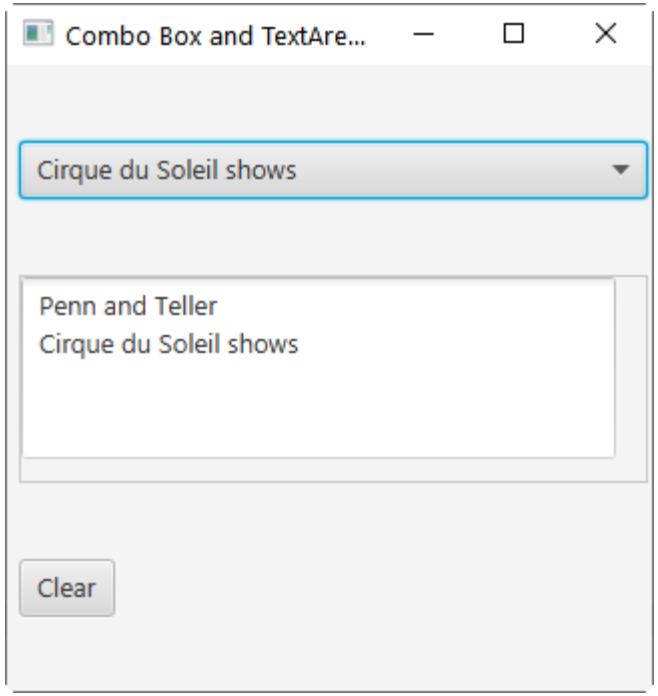
TextArea

- A `TextArea` is a multiline `TextField` that can accept several lines of text input, or display several lines of text
- By default, `TextArea` controls do not perform text wrapping
  - To enable text wrapping, use the `setWrapText` method.

```
textInput.setWrapText(true);
```

Lab (5)

- ShowSelection.fxml
- ShowSelection.java
- ShowSelectionController.java



Using Model Class in GUI Application

- Model classes can be instantiated in the controller class
  - Methods can be used to perform tasks

Exercise: Employee Data (1)

- Please create a JavaFX application to allow entry of employee name, pay rate, and working hours. Then add the employee details into a text area showing the details with total pay
  - Payroll (included in the lab files) is a model class with three fields and some methods.
  - At the action listener of the button, please instantiate Payroll objects and use its methods to complete the program

Exercise: Employee Data (2)