JavaFX

Building Graphical User Interfaces and Drawing Graphics

JavaFX – Basic Elements

- Basic JavaFX classes for creating the layout and organization of a graphical application:
  - The “Application” class – javafx.application.Application
  - The “Scene” class – javafx.scene.Scene
  - The “Stage” class – javafx.stage.Stage
  - The “Pane” class – javafx.scene.layout.Pane
  - The “StackPane” class – javafx.scene.layout.StackPane
  - The “FlowPane” class – javafx.scene.layout.FlowPane
  - The “GridPane” class – javafx.scene.layout.GridPane
  - The “BorderPane” class – javafx.scene.layout.BorderPane
  - The “HBox” class – javafx.scene.layout.Hbox
  - The “VBox” class – javafx.scene.layout.Vbox

JavaFX – Basic Elements

- Basic JavaFX classes for modifying the layout and appearance of a graphical application:
  - The “Color” class – javafx.scene.paint.Color
  - The “Insets” class – javafx.geometry.Insets
  - The “Pos” class – javafx.geometry.Pos
  - The “Font” class – javaFX.scene.text.Font
  - The “FontWeight” – javafx.scene.text.FontWeight

JavaFX – Basic Elements

- Basic JavaFX classes for modifying the layout and appearance of a graphical application:
  - The “Color” class – javafx.scene.paint.Color
  - The “Insets” class – javafx.geometry.Insets
  - The “Pos” class – javafx.geometry.Pos

JavaFX – Basic Elements

- Basic JavaFX controls for creating elements with which users interact:
  - The “Text” class – javafx.scene.text.Text
  - The “Label” class – javafx.scene.control.Label
  - The “TextField” class – javafx.scene.control.TextField
  - The “PasswordField” class – javafx.scene.control.PasswordField
  - The “Button” class – javafx.scene.control.Button
  - The “ActionEvent” class – javafx.event.ActionEvent
  - The “EventHandler” class – javafx.event.EventHandler

JavaFX – Basic Elements

- Basic JavaFX classes for detecting user actions like clicking buttons
  - The “ActionEvent” class – javafx.event.ActionEvent
  - The “EventHandler” class – javafx.event.EventHandler

JavaFX – Basic Elements

- Basic JavaFX classes – the “Application” class
  - All classes that implement JavaFX must extend the Application class.
  - Such classes do not use a “main” method.
  - They use a “start” method to launch themselves.
  - Actions performed by the “start” method include:
    - Creating an instance of one of the “Pane” classes to organize the application’s layout
    - Creating an instance of the Scene class and passing the Pane object to it
    - Invoking various methods of the primaryStage object to setup the scene and show it

JavaFX – Basic Elements

- Basic JavaFX classes – the “Scene” class
  - Think of a theater analogy: you view the stage on which a scene is presented.
  - Represents a graphical construct which contains other elements.
  - We create a scene object and place it on a stage object.
  - Uses an object of one of the “Pane” classes to organize the layout.

JavaFX – Basic Elements

- Basic JavaFX classes – the “Stage” class
  - Think of a theater analogy: you view the stage on which a scene is presented.
  - Represents a graphical construct which contains the scene object.
  - We do NOT explicitly create an object of the Stage class; rather, Java “magically” creates it and passes it into the start method of the Application class.
  - The setTitle method of the Stage class puts a title on the application’s window.
  - The setScene method of the Stage class associates the Scene object with the Stage, for example:

```
stageObject.setScene( sceneObject );
```

JavaFX – Basic Elements

- Basic JavaFX classes – the various “Pane” classes
  - Panes provide a means for arranging the graphical elements in a scene.
  - The various Pane classes provide different ways of arranging them.
  - Pane class – very little organization.
  - StackPane class – allows you to stack graphical elements on top of each other; elements can overlay other elements, beyond this, little organization.
  - FlowPane class – Place elements on the scene from top left to bottom right in the order in which they are added, very little organization.
  - GridPane class – Very organized, it creates a grid and you can specify which sub-pane into which an element will be placed.
  - BorderPane class – Also, very organized, has pre-arranged sub-panes – center, north, south, east and west – into which you can place elements. You do need not to use all of the possible sub-panes.
  - You can add pane objects to other pane objects to manage layouts.

JavaFX – Basic Elements

- Basic JavaFX classes – the “HBox” and “VBox” classes
  - These are container classes into which you place other elements.
  - HBox provides a horizontally oriented way of arranging them.
  - VBox provides a vertically oriented way of arranging them.
  - You use HBox and VBox objects to organize other graphical objects then you add the HBox and VBox objects to a pane.

JavaFX – Basic Elements

- Basic JavaFX classes – the Color class:
  - The “Color” class creates custom colors using four numeric values that represent the amounts of red, green and blue in a color and its opacity.
  - The values of these numbers range from 0.0 for the darkest hue of a color to 1.0 for the lightest hue.
  - Opacity ranges from 0.0 for transparent to 1.0 for opaque, for example:

```
Color colorObject = new Color(0.25, 0.50, 0.25, 1.0);
```

JavaFX – Basic Elements

- Basic JavaFX classes – the Insets class:
  - The “Insets” class creates padding between the edges of a container object and the objects within it.

```
// The numeric value represent, in order, the padding, in pixels,
// on the Top Right Bottom Left edges. I use TRouBLe to remember it.
grid.setPadding(new Insets(25, 25, 25, 25));
```

JavaFX – Basic Elements

- Basic JavaFX classes – the Pos class:
  - The “Pos” class defines static constants that refer to positions such as left, right and center.
  - Pos.CENTER represents Center aligned vertically and horizontally.
  - Pos.CENTER_LEFT represents Center aligned vertically and Left aligned horizontally.
  - Pos.CENTER_RIGHTT represents Center aligned vertically and Rightt aligned horizontally.
  - Pos.LEFT represents Left aligned vertically and horizontally.
  - Pos.RIGHT represents Right aligned vertically and horizontally.
  - Pos.TOP_LEFT represents vertical alignment Top and horizontal Left.
  - Pos.TOP_CENTER represents vertical alignment Top and horizontal Center.
  - Pos.TOP_RIGHT represents vertical alignment Top and horizontal Right.
  - Pos.BOTTOM_LEFT represents vertical alignment Bottom and horizontal Left.
  - Pos.BOTTOM_CENTER represents vertical alignment Bottom and horizontal Center.
  - Pos.BOTTOM_RIGHT represents vertical alignment Bottom and horizontal Right.

```
//For example, to set the alignment of objects within an HBox named hbBtn:
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
```

JavaFX – Basic Elements

- Basic JavaFX classes – the Font class:
  - The “Font” class determines what font to use in rendering text.
  - It uses a Font-Family like “Times New Roman” or “Arial” and a Font-size in points, for instance:

```
Font tnr20 = new Font(“Times New Roman”, 20 );
```

We could use variables or named constants instead of the literal values above:

```
Font tnr20 = new Font( FONT_FAMILY_NAME, FONT_SIZE );
```

We can also set the FontWeight (NORMAL, BOLD, BLACK, LIGHT, MEDIUM, etc.) and FontPosture (ITALIC or REGULAR):

```
Font tnr20BoldItalic = new Font(“Times New Roman”, 20, FontWeight.BOLD, FontPosture.ITALIC );
```

JavaFX – Basic Elements

- Basic JavaFX classes – the Font class:
  - We use objects of the "Font" class by applying them to objects that have a text property, for instance:

```
Label lastNameLabel =  new Label("Last name: ");

lastNameLabel.setFont(tnr20);
```

JavaFX – Basic Elements

- Basic JavaFX controls – the Text class:
  - Represents text that appears in a container, for instance:

```
// Create and initialize a Text object.
Text scenetitle = new Text("Welcome");

// Set the font for the Text object.
scenetitle.setFont(Font.font("Georgia", FontWeight.NORMAL, 20));

// Add the Text object to a GridPane object at a specified location.
grid.add(scenetitle, 0, 0, 2, 1);
```

JavaFX – Basic Elements

- Basic JavaFX controls – the Label class:
  - Also represents text that appears in a container.
  - Users cannot directly change the value of a Label object, but a program can.

  For instance:

```
// Create and initialize a Label object.
Label userName = new Label("User Name:");

// Set the font for the Text object. We used a named constant for the FONT_NAME.
userName.setFont(Font.font(FONT_NAME, FontWeight.NORMAL, 20));

// Add the Label object to a GridPane object at a specified location.
grid.add(userName, 0, 1); // column 0, row 1
```

JavaFX – Basic Elements

- Basic JavaFX controls – the TextField class:
  - Represents an input control in which users can type a value.

  For instance:

```
// Create and initialize a TextField object.
TextField userTextField = new TextField();

// Set the font for the TextField object. Notice that we used named constants.
userTextField.setFont(Font.font(FONT_NAME, FontWeight.NORMAL, 20));

// Add the TextFieldobject to a GridPane object at a specified location.
grid.add(userTextField, 1, 1); // column 1, row 1
```

JavaFX – Basic Elements

- Basic JavaFX controls – the PasswordField class:
  - Represents an input control in which users can type a value.
  - Like a TextField except that it conceals the characters being typed.

  For instance:

```
PasswordField pwField = new PasswordField();
pwField.setFont(Font.font(FONT_NAME, FontWeight.NORMAL, 20));
grid.add(pwField, 1, 2); // column 1, row 2
```

JavaFX – Basic Elements

- Basic JavaFX controls – the Button class:
  - Represents a input control which users can click to perform an action.

  For instance:

```
// Create a Button object labeled “Sign in”.
Button btn = new Button("Sign in");

// Set the Font of the Button’s label.
btn.setFont(Font.font(FONT_NAME, FontWeight.NORMAL, 20));

// Add the Button object to an HBox object. More on getChildren later.
hbBtn.getChildren().add(btn);

// Associate an action with the Button object, i. e., what will happen when it’s clicked.
btn.setOnAction(new EventHandler<ActionEvent>() { // Code for EventHandler…}
```

JavaFX – Basic Elements

- Basic JavaFX controls – the ActionEvent and EventHandler classes:
  - These classes work together to define what happens when a user triggers an action by, for instance, clicking a Button or selecting an item from a list. See the sample program LoginFormGridPane.

The code below creates an anonymous EventHandler object by invoking the EventHandler constructor, btn.setOnAction(new EventHandler<ActionEvent>() , and providing the code for its handle method within the braces that follow it.

```
btn.setOnAction(new EventHandler<ActionEvent>()
{
        public void handle(ActionEvent e)
        {
                // Code to execute
        }
});
```

JavaFX – Basic Elements

- Basic JavaFX controls – the ActionEvent and EventHandler classes:
  - These classes work together to define what happens when a user triggers an action by, for instance, clicking a Button or selecting an item from a list. See the sample program LoginFormGridPaneV2.

The code below creates a named EventHandler object by invoking the EventHandler constructor, btn.setOnAction(new EventHandler<ActionEvent>() , and providing the code for its handle method within the braces that follow it.

```
EventHandler signonBtnHandler = new EventHandler<ActionEvent>()
{
    public void handle(ActionEvent e)
    {
        // Code to execute
    }
};
btn.setOnAction(signonBtnHandler); // Associate btn with the named EventHandler.
```

JavaFX – Basic Elements

- Basic JavaFX controls – the ActionEvent and EventHandler classes:

  The code below creates a separate, user-defined class that implements EventHandler. To perform its work, the handle method of this class invokes a public method within the form class. See the sample program LoginFormGridPaneV3.

```
class ClickSignonButton implements EventHandler<ActionEvent>
{
	LoginFormGridPaneV3 formObj = null; // Declare a data member to represent an object of the form class.

	public ClickSignonButton(LoginFormGridPaneV3 formObj) // This constructor receives an object of the form class.
	{
		this.formObj = formObj;
	}

	public void handle(ActionEvent e)
	{
		formObj.setNotifications(); // formObj.setNotifications() is a public method in LoginFormGridPaneV3
	}
}
// Back in the form class LoginFormGridPaneV3:
ClickSignonButton signonBtnHandler = new ClickSignonButton( this ); // Instantiate event handler and pass the form itself into it.
btn.setOnAction(signonBtnHandler); // Associate Button btn with the named EventHandler signonBtnHandler.
```

JavaFX – Basic Elements

See the sample programs in the JavaFX content module, especially LoginFormGridPane.java, LoginFormGridPaneV2.java and LoginFormGridPaneV3.java

The module includes other programs that illustrate slightly more advanced topics like using CSS to apply style formats to JavaFX objects.

JavaFX – Basic Elements

A more streamlined approach to associating ActionListeners with graphical controls, note **->**:

```
btnGetStudents.setOnAction((ActionEvent e) ->
{
        String selectedMajor = majorDataCombo.getValue();

        majorCodeParm = selectedMajor.substring(0, 3);

        getStudentData();

});
```

JavaFX – Basic Elements

Adding objects to containers like the various Panes, HBoxes and VBoxes

Objects placed within these containers are considered “children” of the containers. The containers maintain a list of their children. To add an object to a container, you must first access the list of its children using the getChildren method. Then you can invoke the add method to add the object, for instance:

For more info see Oracle's JavaFX Documentation

https://docs.oracle.com/javase/8/javafx/api/overview-summary.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/BorderPane.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/HBox.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/VBox.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/text/FontWeight.html

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/text/FontPosture.html

And more.
Notice how the folders after “api” use the words in the package name.