**26.8 Wrap-Up**

In this chapter, we continued our presentation of JavaFX. We discussed JavaFX layout panes in more detail. You usedTitledPanes to organize RadioButtons and an AnchorPaneto display Circles.

You learned about the many mouse events supported by JavaFX nodes and we used the onMouseDragged event in a simple **Painter** app that displayed Circles as the user dragged the mouse across an AnchorPane. The **Painter** app allowed the user to choose the current color and pen size from groups of mutually exclusive RadioButtons. You usedToggleGroups to manage the relationship between theRadioButtons in each group. You also learned how to provide a so-called user data Object for a control. When aRadioButton was selected, you obtained it from theToggleGroup, then accessed the RadioButton’s user dataObject to determine the drawing color or pen size.

We discussed property binding and property listeners, then used them to implement a **Color Chooser** app. You bound aTextField’s text to a Slider’s value to automatically update the TextField when the user moved the Slider’s thumb. You also used a property listener to allow the app’s controller to update the color of a Rectangle when a Slider’s value changed.

In our **Cover Viewer** app, we showed how to bind aObservableList collection to a ListView control to populate it with the collection’s elements. By default, each object inthe collection was displayed as a String in the ListView. You configured a property listener to display an Image in anImageView when the user selected an item in the ListView. Finally, we modified the **Cover Viewer** app to use a customListView cell factory to specify the exact layout of aListView cell’s contents.

In the next chapter, we’ll demonstrate additional JavaFX features, including graphics, multimedia and customizing a GUI’s look-and-feel with JavaFX’s CSS (Cascading Style Sheets) capabilities.

#### Exercises

**26.1** ***(Painter*** ***App Modification)*** Incorporate the RGBA color chooser you created in the **Color Chooser** app ([Section 26.5](http://proquest.safaribooksonline.com/9780133813036/ch26lev1sec5_html#ch26lev1sec5)) into the **Painter**app ([Section 26.4](http://proquest.safaribooksonline.com/9780133813036/ch26lev1sec4_html#ch26lev1sec4)) so that the user can choose any drawing color. Changing a Slider’s value should update the color swatch displayed to the user and set the brushColor instance variable to the currentColor.

**26.2** ***(Contacts*** ***App)*** Create a **Contacts** app modeled after the **Cover Viewer** app ([Sections 26.6](http://proquest.safaribooksonline.com/9780133813036/ch26lev1sec6_html#ch26lev1sec6)–[26.7](http://proquest.safaribooksonline.com/9780133813036/ch26lev1sec7_html#ch26lev1sec7)). Store the contact information in anObservableList of Contact objects. A Contact should contain first name, last name, email and phone number properties (you can provide others). When the user selects a contact from the contacts list, its information should display in a Grid of TextFields. As the information is modified (a Contact’s data is updated, a new Contact is added or an existing Contact is deleted, the contacts ListView should display the updates. The ListView should display the Contact’s last names.

**26.3** ***(Contacts*** ***App Modification)*** Modify the **Contacts** app from the preceding exercises to include an image for each Contact. Provide a custom ListView cell factory that displays the Contact’s full name and picture with the names in sorted order by last name then first name.

**26.4** ***(Tip Calculator Modification)*** The **Tip Calculator** app from[Section 25.5](http://proquest.safaribooksonline.com/9780133813036/ch25lev1sec5_html#ch25lev1sec5) does not need a Button to perform its calculations. Reimplement this app to use property listeners to perform the calculations whenever the user modifies the bill amount or changes the custom tip percentage. Also use a property binding to update the Labelthat displays the tip percentage.

**26.5** ***(Advanced Project:*** ***Color Chooser*** ***App Modification)*** The property bindings we created in the **Color Chooser** app ([Section 26.5](http://proquest.safaribooksonline.com/9780133813036/ch26lev1sec5_html#ch26lev1sec5)) allowed a TextField’s text to update when a Slider’s value changed, but not vice versa. JavaFX also supports bi-directional property bindings. Research bi-directional property bindings online, then create bi-directional bindings between the Sliders and the TextFields such that modifying a TextField’s value updates the corresponding slider.