Chapter 15 Lab
Creating GUI Applications with JavaFX and Scene Builder

Lab Objectives

- Be able to create a GUI with Scene Builder and save it to an FXML file
- Be able to edit the JavaFX application's main application class
- Be able to edit the logic in the JavaFX application's controller class
- Be able to run and test a standalone JavaFX application

Introduction

JavaFX is a standard Java library for developing rich applications that employ graphics. You can use it to create GUI applications, as well as applications that display 2D and 3D graphics. You can use JavaFX to create standalone graphics applications that run on your local computer, applications that run from a remote server, or applications that are embedded in a Web page.

You use Scene Builder to construct a GUI by dragging and dropping the components that you need onto a blank window. You visually arrange the components on the window, and set various component properties to create the appearance that you want for the GUI. Then, you save the GUI to an FXML file.

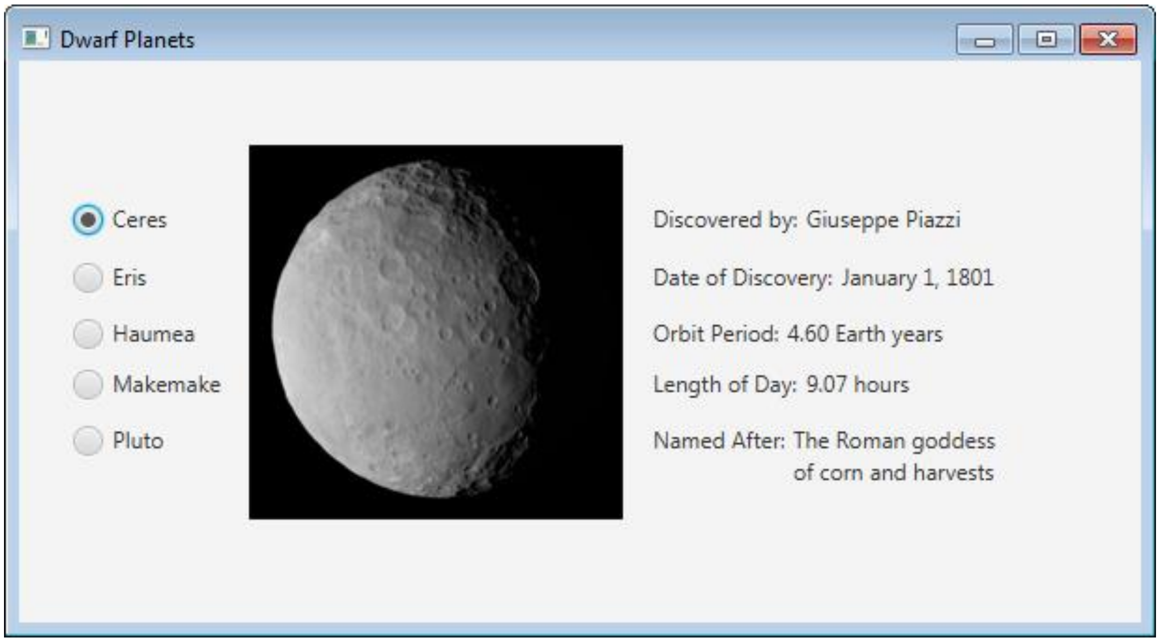
Once you save a GUI's scene graph to an FXML file, the next step is to write Java code that reads the FXML file and displays the GUI on the screen, and handles any events that occur while the application is running. This code is divided into two classes:

1. The main application class is responsible for building the scene graph and displaying the GUI. The main application class performs the following:
    - Loads the FXML file
    - Builds the scene graph in memory
    - Displays the GUI

2. 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 an `fx:id` in the scene graph
    - An optional `initialize` method that is automatically called after the FXML file is loaded
    - Event listener methods

In this lab we will create a standalone GUI application using JavaFX and Scene Builder. The application will display information about the first five dwarf planets that were discovered in our solar system. The application will display an image of a dwarf planet along with several facts. The image and text will change depending on which radio button is selected.

Our Final GUI should look like the following:



Discovered by: Giuseppe Piazzi

Date of Discovery: January 1, 1801

Orbit Period: 4.60 Earth years

Length of Day: 9.07 hours

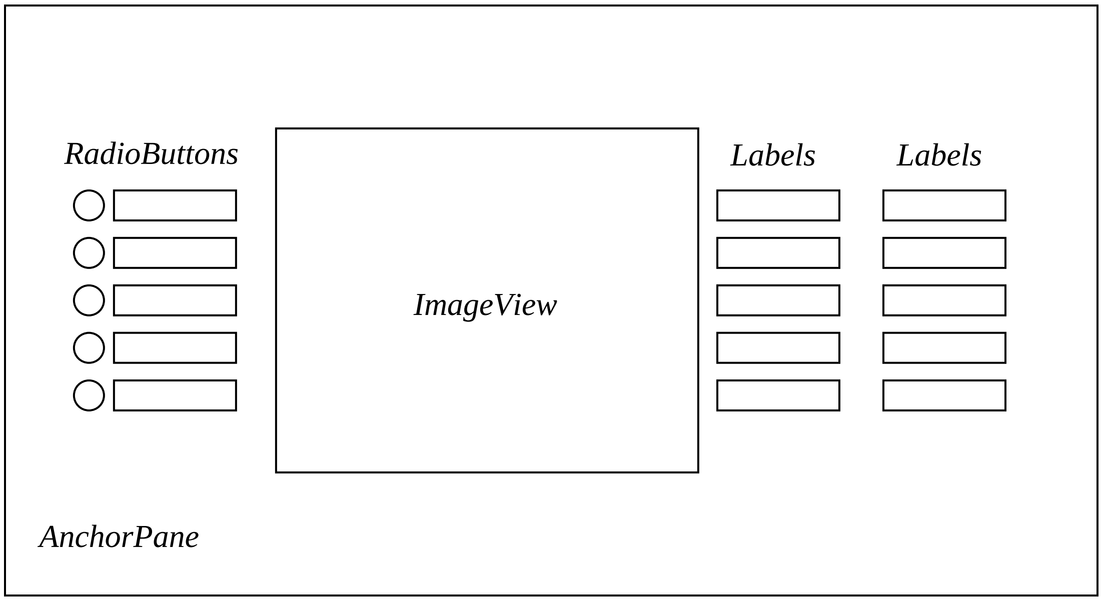
Named After: The Roman goddess of corn and harvests

Task #1 Creating the GUI's Scene Graph with Scene Builder

1. Copy the following files from the Student CD or as directed by your instructor.
    - *DwarfPlanets.java* (see Code Listing 15.1)
    - *DwarfPlanetsController.java* (see Code Listing 15.2)
    - *Ceres.png*
    - *Eris.png*
    - *Haumea.png*
    - *Makemake.png*
    - *Pluto.png*
2. Open Scene Builder.
3. Add an AnchorPane component to the scene. This will become the root node.
    a. Save the GUI as *DwarfPlanets.fxml*. Make sure this file is saved in the same location as the files you copied from the Student CD or as directed by your instructor.
    b. With the AnchorPane selected, set the Controller Class to the following:
        i. `DwarfPlanetsController`.

4. Add five RadioButton components, arranged in a single column, to the left side of the AnchorPane.
    a. Set the Text property of each RadioButton to the following:
        i. *Ceres*
        ii. *Eris*
        iii. *Haumea*
        iv. *Makemake*
        v. *Pluto*
    b. Set the `fx:id` field of each RadioButton to the following:
        i. `ceresRadioButton`
        ii. `erisRadioButton`
        iii. `haumeaRadioButton`
        iv. `makemakeRadioButton`
        v. `plutoRadioButton`
    c. To register the event listeners, set the On Action property of each RadioButton to the following:
        i. `ceresRadioButtonListener`
        ii. `erisRadioButtonListener`
        iii. `haumeaRadioButtonListener`
        iv. `makemakeRadioButtonListener`
        v. `plutoRadioButtonListener`
    d. Set the Toggle Group property of each RadioButton to the following:
        i. `dwarfPlanetsToggleGroup`
    e. Enable the Selected property of the `ceresRadioButton` RadioButton component.
5. Add an ImageView component to the right of the five RadioButton components.
    a. Resize the component to 200 pixels wide by 200 pixels high.
    b. Set the `fx:id` field to `dwarfPlanetsImageView`.
    c. Set the Image property to *Ceres.png*.
6. Add five Label components, arranged in a single column, to the right of the ImageView component.
    a. Set the Text property of each Label to the following:
        i. *Discovered By:*
        ii. *Date of Discovery:*
        iii. *Orbit Period:*
        iv. *Length of Day:*
        v. *Named After:*
7. Add five more Label components, arranged in a single column, to the right of the five Label components you added earlier.
    a. Set the Text property of each Label to the following:
        i. *Giuseppe Piazzi*
        ii. *January 1, 1801*
        iii. *4.60 Earth years*
        iv. *9.07 hours*
        v. *The Roman goddess of corn and harvests*

   b. You will notice that the last Label component is not large enough for all the text to fit. To allow the text to fit properly, set its properties to the following:
      i. Resize to 160 pixels wide by 40 pixels high
      ii. Set the Alignment property to TOP_LEFT
      iii. Enable the Wrap Text property.
   c. Set the `fx:id` field of each Label to the following:
      i. `discoveredByLabel`
      ii. `dateOfDiscoveryLabel`
      iii. `orbitPeriodLabel`
      iv. `lengthOfDayLabel`
      v. `namedAfterLabel`
8. Make sure the GUI components are arranged to resemble the diagram below.
9. Save the changes and close Scene Builder.



Task #2 Editing the Main Application Class

1. Open the `DwarfPlanets.java` file.
2. Edit the class so that it will load the `DwarfPlanets.fxml` file.
3. Edit the class so that it will display the text *Dwarf Planets* in the window title.
4. Save and compile the main application class, correcting any errors.
5. Run the main application class. Test the radio buttons. Notice that the radio buttons don't do anything. We will edit the logic in the next task.

Task #3 Editing the Controller Class

1. Open the `DwarfPlanetsController.java` file.
2. Edit the class so that it will load the images when the application starts.

    a. Import the JavaFX `Image` class.
    b. Declare private fields for each of the five images.
    c. Create the `initialize` method to load the images.
3. Edit the event listener method for each radio button so it will display the appropriate data when selected.
    a. Use the information provided in the table below.
4. Save and compile the controller class, correcting any errors.

Image	Discovered By	Discovery Date	Orbit Period	Length of Day	Named After
Ceres.png	Giuseppe Piazzi	January 1, 1801	4.60 Earth Years	9.07 Hours	The Roman Goddess of corn and harvests
Eris.png	Brown, Trujillo, and Rabinowitz	October 21, 2003	561.37 Earth Years	25.90 Hours	The ancient Greek goddess of discord and strife
Haumea.png	Sierra Nevada Observatory	March 7, 2003	281.93 Earth Years	3.91 hours	The Hawaiian goddess of childbirth and fertility
Makemake.png	Brown, Trujillo, and Rabinowitz	March 31, 2005	305.34 Earth years	22.48 hours	The Rapanui god of fertility
Pluto.png	Clyde Tombaugh	February, 18 1930	247.92 Earth years	153.29 hours	The ancient Greek god of the underworld

Task #4 Run and Test the JavaFX Application

1. Run the `DwarfPlanets` JavaFX application.
2. Test the application's radio buttons, and determine that the output is correct.
    a. Refer to the data in the table for testing.

Code Listing 15.1 (`DwarfPlanets.java`)

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

/**
   This is the main application class for the
   Dwarf Planets JavaFX application.
 */
```

```
public class DwarfPlanets extends Application
{
   public void start(Stage stage) throws Exception
   {
      // EDIT THE LINES FOR TASK #2 HERE
      // Load the DwarfPlanets.fxml file.
      Parent parent = FXMLLoader.load(
      getClass().getResource("FILENAME.fxml"));

      // Build the scene graph.
      Scene scene = new Scene(parent);

      // EDIT THE LINES FOR TASK #2 HERE
      // Set the title to display "Dwarf Planets".
      // Display our window, using the scene graph.
      stage.setTitle("TITLE");
      stage.setScene(scene);
      stage.show();
   }

   public static void main(String[] args)
   {
      // Launch the application.
      launch(args);
   }
}
```

Code Listing 15.2 (`DwarfPlanetsController.java`)

```
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
// ADD LINES FOR TASK #3 HERE
// Import the JavaFX Image class.
import javafx.scene.image.ImageView;

/**
   This is the controller class for the
   Dwarf Planets JavaFX application.
 */

public class DwarfPlanetsController
{
   @FXML
   private ImageView dwarfPlanetsImageView;
```

```
@FXML
private Label discoveredByLabel;

@FXML
private Label dateOfDiscoveryLabel;

@FXML
private Label orbitPeriodLabel;

@FXML
private Label lengthOfDayLabel;

@FXML
private Label namedAfterLabel;

@FXML
private ToggleGroup dwarfPlanetsToggleGroup;

@FXML
private RadioButton ceresRadioButton;

@FXML
private RadioButton erisRadioButton;

@FXML
private RadioButton haumeaRadioButton;

@FXML
private RadioButton makemakeRadioButton;

@FXML
private RadioButton plutoRadioButton;

// ADD LINES FOR TASK #3 HERE
// Declare private fields for the images
// Load the image files in the intialize method

// Event listener for the ceresRadioButton
public void ceresRadioButtonListener()
{
   // ADD THE LINES FOR TASK #3 HERE
   // If this radio button is selected,
   // display image and data for Ceres.
}

// Event listener for the erisRadioButton
```

```
   public void erisRadioButtonListener()
   {
      // ADD THE LINES FOR TASK #3 HERE
      // If this radio button is selected,
      // display image and data for Eris.
   }

   // Event listener for the haumeaRadioButton
   public void haumeaRadioButtonListener()
   {
      // ADD THE LINES FOR TASK #3 HERE
      // If this radio button is selected,
      // display image and data for Haumea.
   }

   // Event listener for the makemakeRadioButton
   public void makemakeRadioButtonListener()
   {
      // ADD THE LINES FOR TASK #3 HERE
      // If this radio button is selected,
      // display image and data for Makemake.
   }

   // Event listener for the plutoRadioButton
   public void plutoRadioButtonListener()
   {
      // ADD THE LINES FOR TASK #3 HERE
      // If this radio button is selected,
      // display image and data for Pluto.
   }
}
```