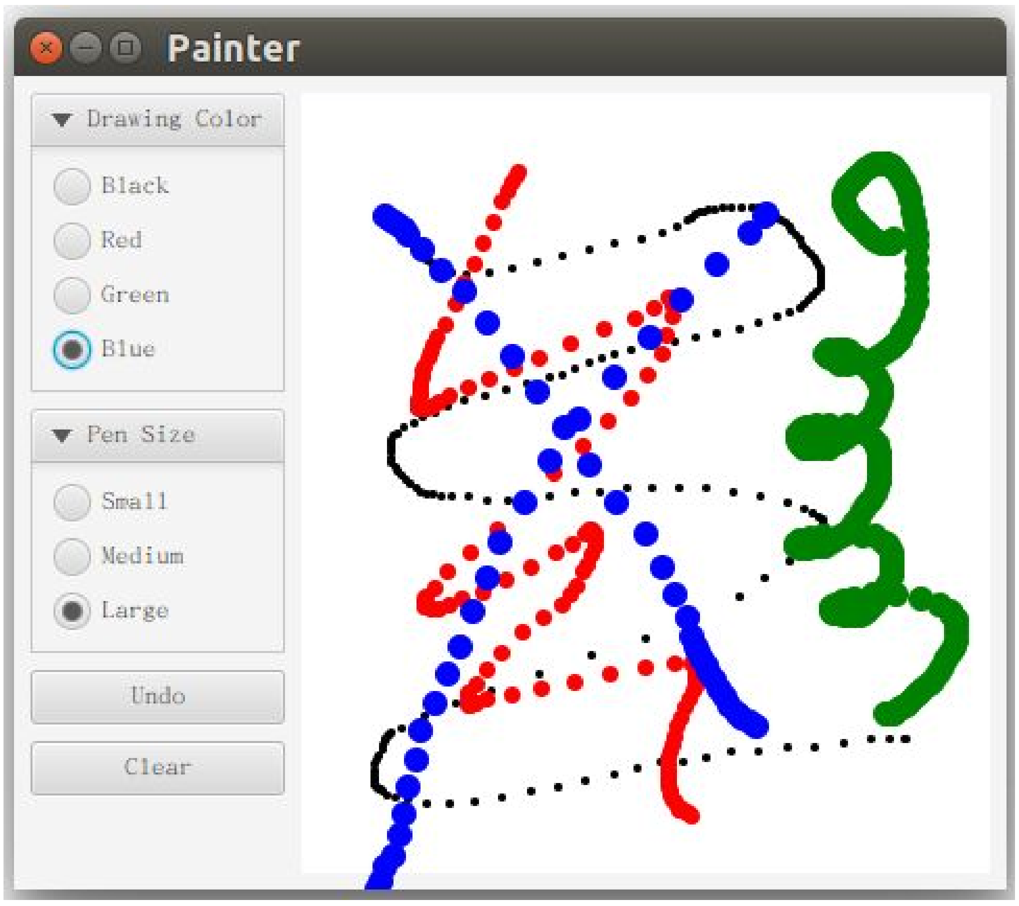
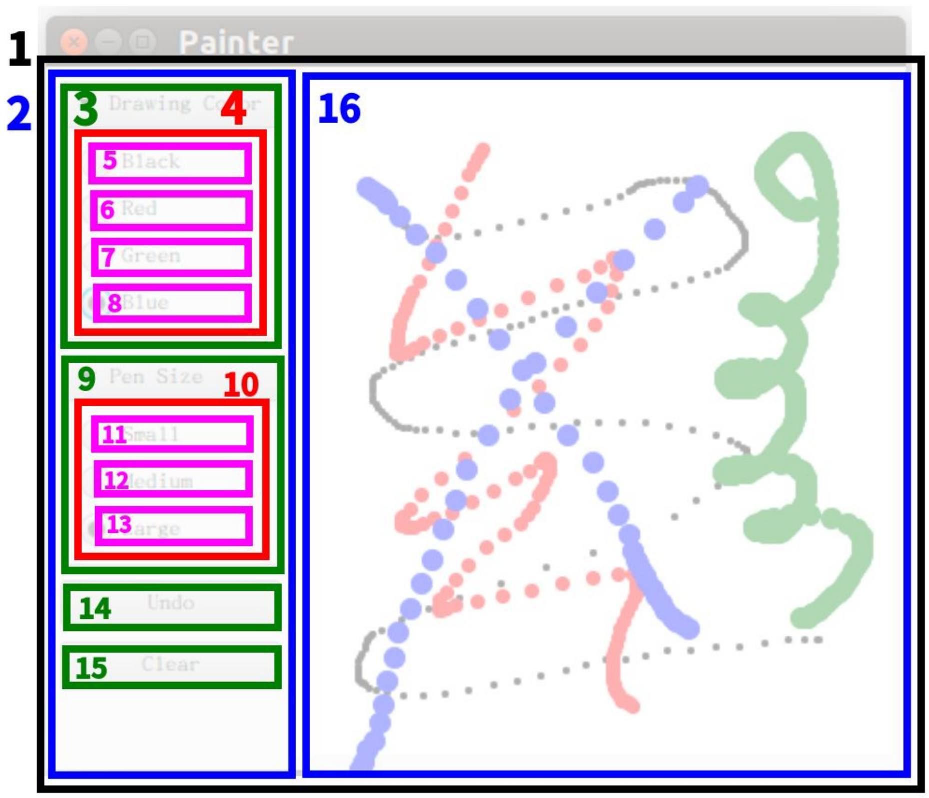
Homework 5

Please implement following GUI by Scene Builder and complete the application with given codes. Study the codes carefully and make sure you get a best understanding of what/how/when the programs do.

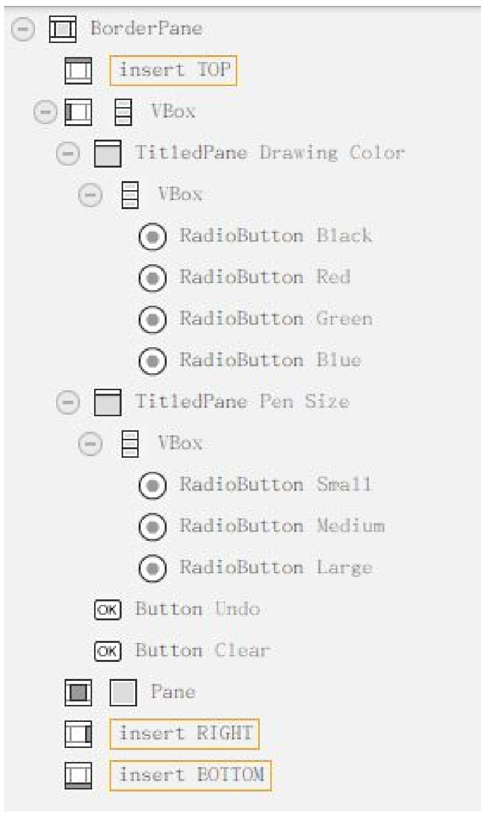
1. Painter



GUI Description:



Hierarchy:



```
BorderPane
    insert TOP
    VBox
        TitledPane Drawing Color
            VBox
                RadioButton Black
                RadioButton Red
                RadioButton Green
                RadioButton Blue
        TitledPane Pen Size
            VBox
                RadioButton Small
                RadioButton Medium
                RadioButton Large
        Button Undo
        Button Clear
    Pane
    insert RIGHT
    insert BOTTOM
```

0) File Name: Painter.fxml
Controller Class: PainterController

1) BorderPane

a) Pref Width Height: 640 x 480
b) Padding: 8 8 8 8 (TOP, RIGHT, BOTTOM, LEFT)

2) VBox *(BorderPane - LEFT)*

a) Max Height: MAX_VALUE
b) Spacing: 8

3) TitledPane

a) Text: "Drawing Color"

4) VBox

a) Spacing: 8

5~8) Radio Button x4

a) fx:id: “blackRadioButton” / “red...” / “green...” / “blue...”
b) On Action: “colorRadioButtonSelected”
c) Toggle Group: “colorToggleGroup”
d) Text: “Black” / “Red” / “Green” / “Blue”
e) Selected: True / False / False / False

9) TitledPane

a) Text: "Pen size"

10) VBox

a) Spacing: 8

11~13) Radio Button x3

a) fx:id: “smallRadioButton” / “medium...” / “large...”
b) On Action: “sizeRadioButtonSelected”
c) Toggle Group: “sizeToggleGroup”
d) Text: “Small” / “Medium” / “Large”
e) Selected: False / True / False

14) Button

a) fx:id: “undoButton”
b) On Action: “undoButtonPressed”
c) Max Width: MAX_VALUE
d) Text: "Undo"

15) Button

a) fx:id: “clearButton”
b) On Action: “clearButtonPressed”
c) Max Width: MAX_VALUE
d) Text: "Clear"

16) Pane *(BorderPane - RIGHT)*

a) fx:id: “drawingAreaPane”
b) On Mouse Dragged: “drawingAreaMouseDragged”
c) Pref Width Height: 200 x 200
d) Style: "-fx-background-color: white;"

Code:

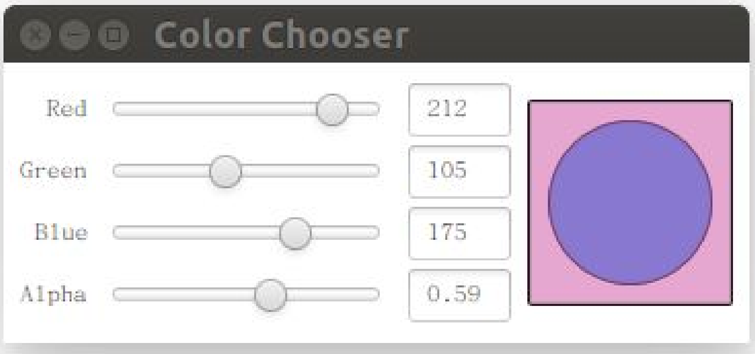
Painter.java

```java
// Main application class that loads and displays the Painter's GUI.
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Painter extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
         FXMLLoader.load(getClass().getResource("Painter.fxml"));

      Scene scene = new Scene(root);
      stage.setTitle("Painter"); // displayed in window's title bar
      stage.setScene(scene);
      stage.show();
   }

   public static void main(String[] args) {
      launch(args);
   }
}
```

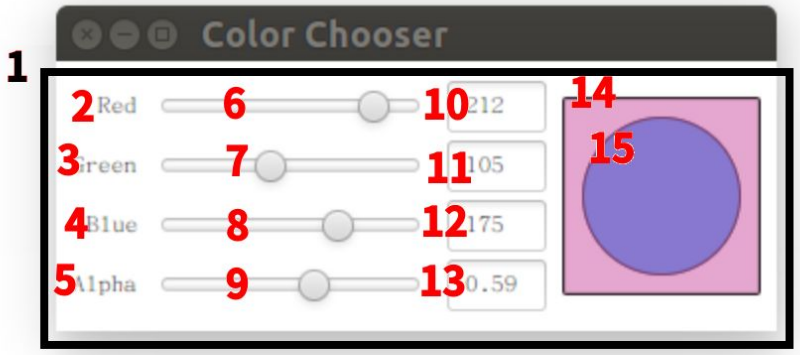
PainterController.java

```java
// Controller for the Painter app
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.paint.Paint;
import javafx.scene.shape.Circle;

public class PainterController {
   // enum representing pen sizes
   private enum PenSize {
      SMALL(2),
      MEDIUM(4),
      LARGE(6);

      private final int radius;

      PenSize(int radius) {this.radius = radius;}

      public int getRadius() {return radius;}
   };

   // instance variables that refer to GUI components
   @FXML private RadioButton blackRadioButton;
   @FXML private RadioButton redRadioButton;
   @FXML private RadioButton greenRadioButton;
   @FXML private RadioButton blueRadioButton;
   @FXML private RadioButton smallRadioButton;
   @FXML private RadioButton mediumRadioButton;
   @FXML private RadioButton largeRadioButton;
   @FXML private Pane drawingAreaPane;
   @FXML private ToggleGroup colorToggleGroup;
   @FXML private ToggleGroup sizeToggleGroup;

   // instance variables for managing Painter state
   private PenSize radius = PenSize.MEDIUM; // radius of circle
   private Paint brushColor = Color.BLACK; // drawing color

   // set user data for the RadioButtons
   public void initialize() {
      // user data on a control can be any Object
      blackRadioButton.setUserData(Color.BLACK);
      redRadioButton.setUserData(Color.RED);
      greenRadioButton.setUserData(Color.GREEN);
      blueRadioButton.setUserData(Color.BLUE);
      smallRadioButton.setUserData(PenSize.SMALL);
      mediumRadioButton.setUserData(PenSize.MEDIUM);
      largeRadioButton.setUserData(PenSize.LARGE);
   }

   // handles drawingArea's onMouseDragged MouseEvent
   @FXML
   private void drawingAreaMouseDragged(MouseEvent e) {
      Circle newCircle = new Circle(e.getX(), e.getY(),
         radius.getRadius(), brushColor);
      drawingAreaPane.getChildren().add(newCircle);
   }

   // handles color RadioButton's ActionEvents
   @FXML
   private void colorRadioButtonSelected(ActionEvent e) {
      // user data for each color RadioButton is the corresponding Color
      brushColor =
         (Color) colorToggleGroup.getSelectedToggle().getUserData();
   }

   // handles size RadioButton's ActionEvents
   @FXML
   private void sizeRadioButtonSelected(ActionEvent e) {
      // user data for each size RadioButton is the corresponding PenSize
      radius =
         (PenSize) sizeToggleGroup.getSelectedToggle().getUserData();
   }

   // handles Undo Button's ActionEvents
   @FXML
   private void undoButtonPressed(ActionEvent event) {
      int count = drawingAreaPane.getChildren().size();

      // if there are any shapes remove the last one added
      if (count > 0) {
         drawingAreaPane.getChildren().remove(count - 1);
      }
   }

   // handles Clear Button's ActionEvents
   @FXML
   private void clearButtonPressed(ActionEvent event) {
      drawingAreaPane.getChildren().clear(); // clear the canvas
   }
}
```

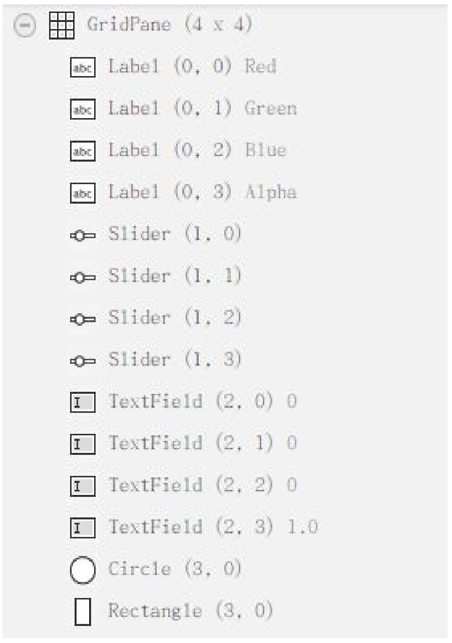
2. ColorChooser



GUI Description:



Hierarchy:



```
GridPane (4 x 4)
    Label (0, 0) Red
    Label (0, 1) Green
    Label (0, 2) Blue
    Label (0, 3) Alpha
    Slider (1, 0)
    Slider (1, 1)
    Slider (1, 2)
    Slider (1, 3)
    TextField (2, 0) 0
    TextField (2, 1) 0
    TextField (2, 2) 0
    TextField (2, 3) 1.0
    Circle (3, 0)
    Rectangle (3, 0)
```

0) File Name: ColorChooser.fxml
Controller Class: ColorChooserController

1) GridPane

a) 4 columns and 3 rows
b) Pref Width Height: Default
c) Column 0: Alignment: RIGHT ; Pref Width: Default
   Column 1: Alignment: CENTER ; Pref Width: Default
   Column 2: Alignment: CENTER ; Pref Width: Default
   Column 3: Alignment: CENTER ; Pref Width: 100 ; Min Width: 10
d) Row 0~2: Pref Width: 30 ; Min Width: 10
e) Padding: 8 8 8 8 (TOP, RIGHT, BOTTOM, LEFT)
f) Hgap: 8
g) Style: "-fx-background-color: white;"

2~5) Label

a) Text: “Red” / “Green” / “Blue” / “Alpha”
b) Row Col Index: 0,0 / 1,0 / 2,0 / 3,0

6~9) Slider

a) fx:id: “redSlider” / “green...” / “blue...” / “alpha...”
b) Row Col Index: 0,1 / 1,1 / 2,1 / 3,1
c) Max: 255 / 255 / 255 / 1.0
d) Value: 0 / 0 / 0 / 1.0
e) Block Increment: 10 / 10 / 10 / 0.1

10~13) TextField

a) fx:id: “redTextField” / “green...” / “blue...” / “alpha...”
b) Row Col Index: 0,2 / 1,2 / 2,2 / 3,2
c) Text: “0” / “0” / “0” / “1.0”
d) Pref Width: 50

14) Rectangle

a) fx:id: “colorRectangle”
b) Row Col Index: 0,3
c) Row Span: REMAINING
d) Width Height: 100 x 100
e) Arc Width Height: 5.0 x 5.0
f) Stroke: BLACK
g) Stroke Type: INSIDE

15) Circle

a) Row Col Index: 0,3
b) Row Span: REMAINING
c) Fill: DODGERBLUE
d) Radius: 40.0
e) Stroke: BLACK
f) Stroke Type: INSIDE

Code:

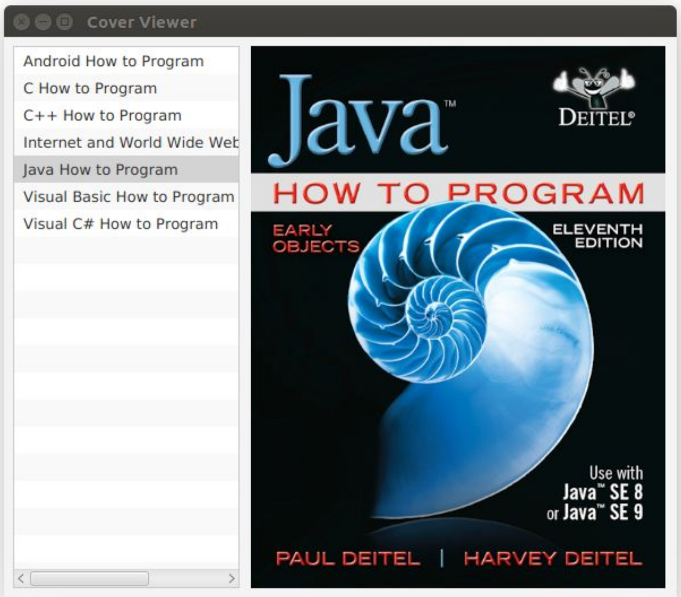
ColorChooser.java

```java
// Main application class that loads and displays the ColorChooser's GUI.
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class ColorChooser extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
         FXMLLoader.load(getClass().getResource("ColorChooser.fxml"));

      Scene scene = new Scene(root);
      stage.setTitle("Color Chooser");
      stage.setScene(scene);
      stage.show();
   }

   public static void main(String[] args) {
      launch(args);
   }
}
```

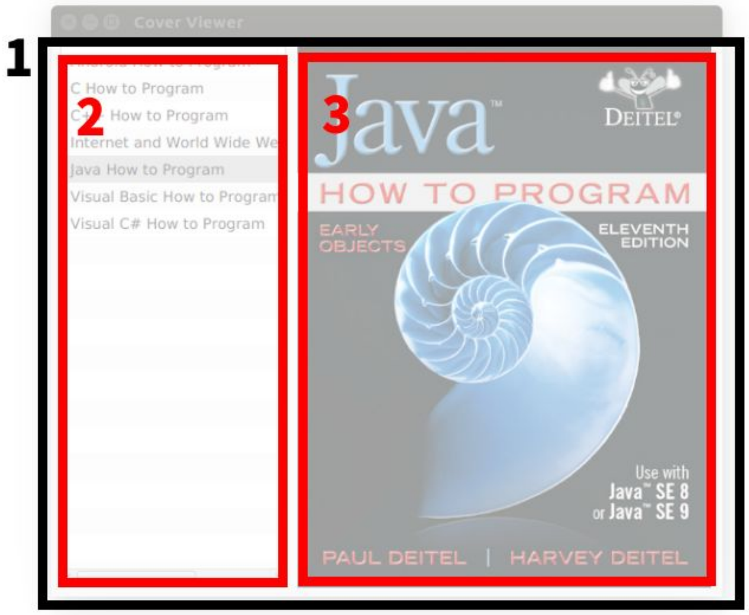
ColorChooserController.java

```java
// Controller for the ColorChooser app
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.fxml.FXML;
import javafx.scene.control.Slider;
import javafx.scene.control.TextField;
import javafx.scene.paint.Color;
import javafx.scene.shape.Rectangle;

public class ColorChooserController {
   // instance variables for interacting with GUI components
   @FXML private Slider redSlider;
   @FXML private Slider greenSlider;
   @FXML private Slider blueSlider;
   @FXML private Slider alphaSlider;
   @FXML private TextField redTextField;
   @FXML private TextField greenTextField;
   @FXML private TextField blueTextField;
   @FXML private TextField alphaTextField;
   @FXML private Rectangle colorRectangle;

   // instance variables for managing
   private int red = 0;
   private int green = 0;
   private int blue = 0;
   private double alpha = 1.0;

   public void initialize() {
      // bind TextField values to corresponding Slider values
      redTextField.textProperty().bind(
         redSlider.valueProperty().asString("%.0f"));
      greenTextField.textProperty().bind(
         greenSlider.valueProperty().asString("%.0f"));
      blueTextField.textProperty().bind(
         blueSlider.valueProperty().asString("%.0f"));
      alphaTextField.textProperty().bind(
         alphaSlider.valueProperty().asString("%.2f"));

      // listeners that set Rectangle's fill based on Slider changes
      redSlider.valueProperty().addListener(
         new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
               Number oldValue, Number newValue) {
               red = newValue.intValue();
               colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
            }
         }
      );
      greenSlider.valueProperty().addListener(
         new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
               Number oldValue, Number newValue) {
               green = newValue.intValue();
               colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
            }
         }
      );
      blueSlider.valueProperty().addListener(
         new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
               Number oldValue, Number newValue) {
               blue = newValue.intValue();
               colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
            }
         }
      );
      alphaSlider.valueProperty().addListener(
         new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov,
               Number oldValue, Number newValue) {
               alpha = newValue.doubleValue();
               colorRectangle.setFill(Color.rgb(red, green, blue, alpha));
            }
         }
      );
   }
}
```

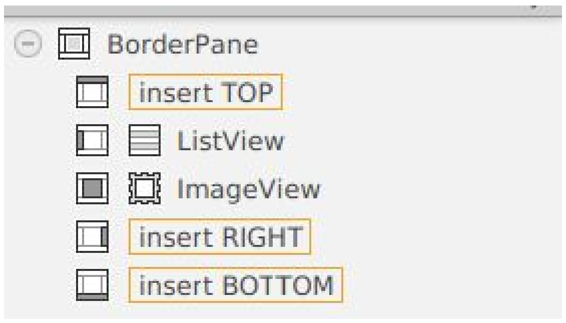
3. CoverViewer



GUI Description:



Hierarchy:



```
BorderPane
    insert TOP
    ListView
    ImageView
    insert RIGHT
    insert BOTTOM
```

0) File Name: CoverViewer.fxml
Controller Class: CoverViewerController

1) BorderPane

a) Pref Width Height: Default
b) Padding: 8 8 8 8 (TOP, RIGHT, BOTTOM, LEFT)

3) ListView *(BorderPane - LEFT)*

a) fx:id: "booksListView"
b) Pref Width: 200
c) Max Height: MAX_VALUE
d) BorderPane Alignment: CENTER
e) BorderPane Margin: 0 8 0 0 (TOP, RIGHT, BOTTOM, LEFT)

2) ImageView *(BorderPane - CENTER)*

a) fx:id: "coverImageView"
b) Fit Width Height: 370 x 480
c) Pick On Bounds: True
d) Preserve Ratio: True

Code:

Book.java

```java
// Book.java
public class Book {
   private String title; // book title
   private String thumbImage; // source of book cover's thumbnail image
   private String largeImage; // source of book cover's full-size image

   public Book(String title, String thumbImage, String largeImage) {
      this.title = title;
      this.thumbImage = thumbImage;
      this.largeImage = largeImage;
   }

   public String getTitle() {return title;}
   public void setTitle(String title) {this.title = title;}
   public String getThumbImage() {return thumbImage;}
   public void setThumbImage(String thumbImage) {this.thumbImage = thumbImage;}
   public String getLargeImage() {return largeImage;}
   public void setLargeImage(String largeImage) {this.largeImage = largeImage;}

   @Override
   public String toString() {return getTitle();}
}
```

CoverViewer.java

```java
// Main application class that loads and displays the CoverViewer's GUI.
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class CoverViewer extends Application {
   @Override
   public void start(Stage stage) throws Exception {
      Parent root =
         FXMLLoader.load(getClass().getResource("CoverViewer.fxml"));

      Scene scene = new Scene(root);
      stage.setTitle("Cover Viewer");
      stage.setScene(scene);
      stage.show();
   }

   public static void main(String[] args) {
      launch(args);
   }
}
```

CoverViewerController.java

```java
// Controller for Cover Viewer application
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.ListView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class CoverViewerController {
   // instance variables for interacting with GUI
   @FXML private ListView<Book> booksListView;
   @FXML private ImageView coverImageView;

   // stores the list of Book Objects
   private final ObservableList<Book> books =
      FXCollections.observableArrayList();

   // initialize controller
   public void initialize() {
      // populate the ObservableList<Book>
      books.add(new Book("Android How to Program",
         "/images/small/androidhtp.jpg", "/images/large/androidhtp.jpg"));
      books.add(new Book("C How to Program",
         "/images/small/chtp.jpg", "/images/large/chtp.jpg"));
      books.add(new Book("C++ How to Program",
         "/images/small/cpphtp.jpg", "/images/large/cpphtp.jpg"));
      books.add(new Book("Internet and World Wide Web How to Program",
         "/images/small/iw3htp.jpg", "/images/large/iw3htp.jpg"));
      books.add(new Book("Java How to Program",
         "/images/small/jhtp.jpg", "/images/large/jhtp.jpg"));
      books.add(new Book("Visual Basic How to Program",
         "/images/small/vbhtp.jpg", "/images/large/vbhtp.jpg"));
      books.add(new Book("Visual C# How to Program",
         "/images/small/vcshtp.jpg", "/images/large/vcshtp.jpg"));
      booksListView.setItems(books); // bind booksListView to books

      // when ListView selection changes, show large cover in ImageView
      booksListView.getSelectionModel().selectedItemProperty().
         addListener(
            new ChangeListener<Book>() {
               @Override
               public void changed(ObservableValue<? extends Book> ov,
                  Book oldValue, Book newValue) {
                  coverImageView.setImage(
                     new Image(newValue.getLargeImage()));
               }
            }
         );
   }
}
```

Local JPG images

```
/images
        /large
                androidhtp.jpg
                chtp.jpg
                cpphtp.jpg
                iw3htp.jpg
                jhtp.jpg
                vbhtp.jpg
                vcshtp.jpg
        /small
                androidhtp.jpg
                chtp.jpg
                cpphtp.jpg
                iw3htp.jpg
                jhtp.jpg
                vbhtp.jpg
                vcshtp.jpg
```