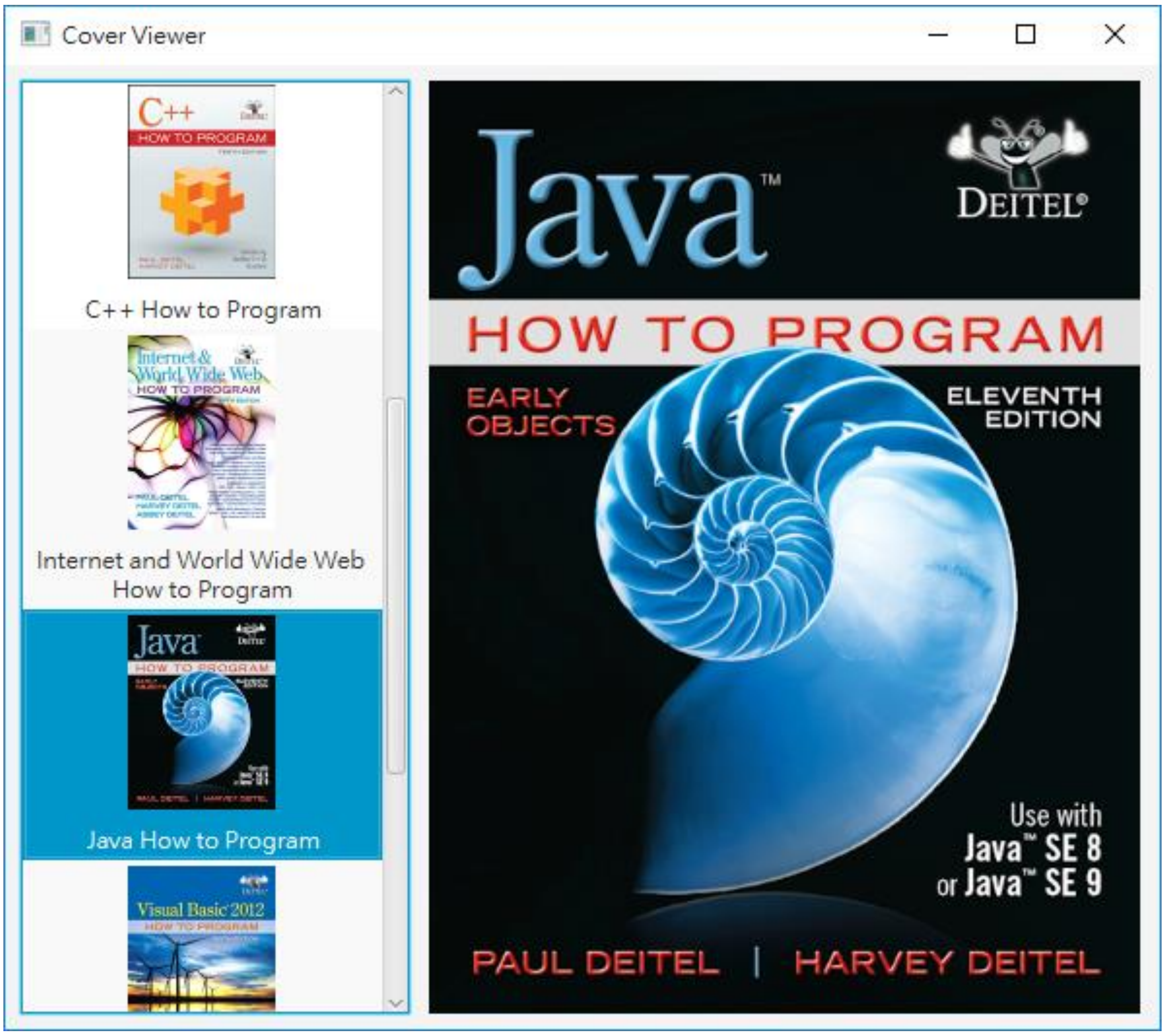
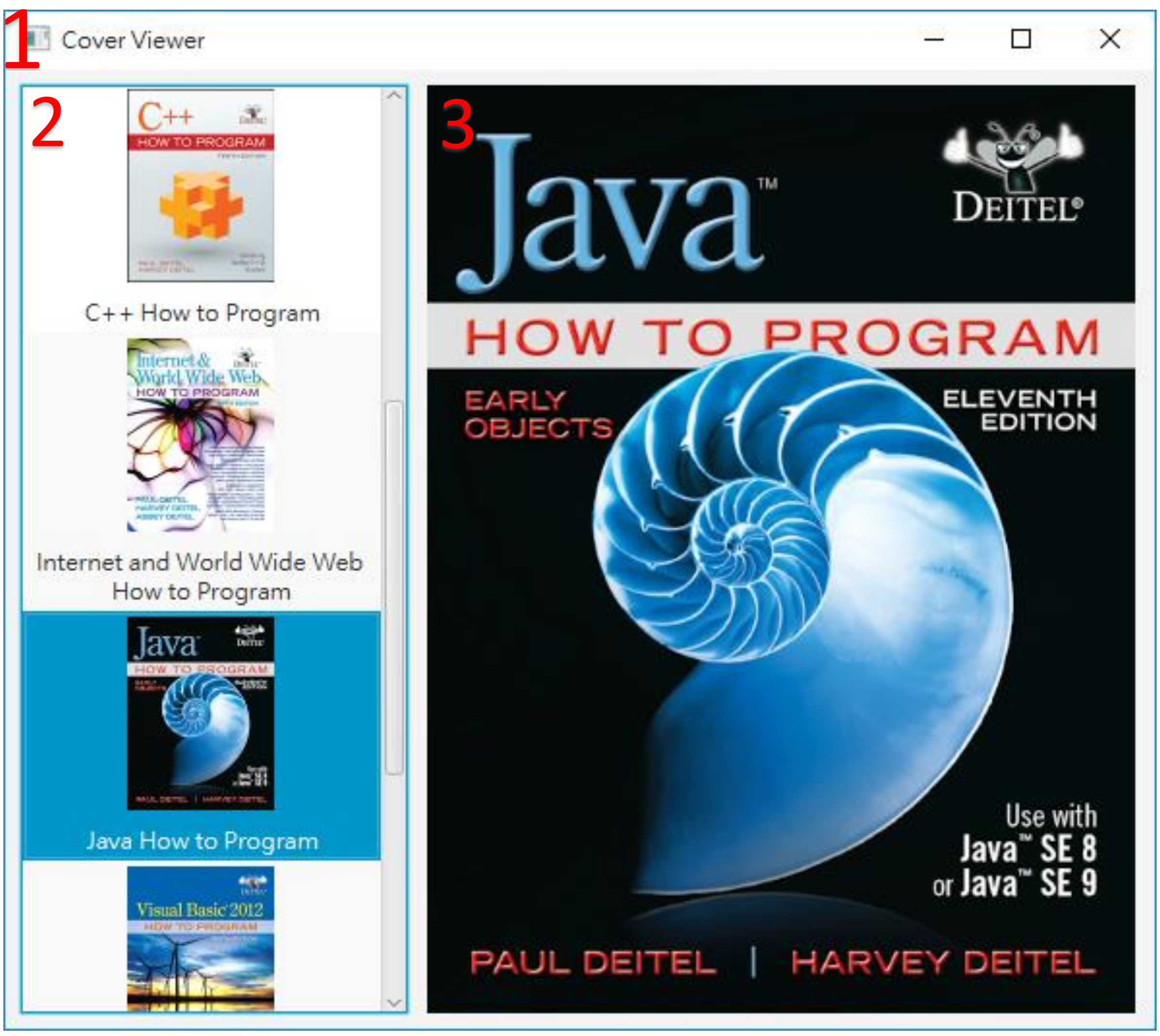
Homework 6

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.

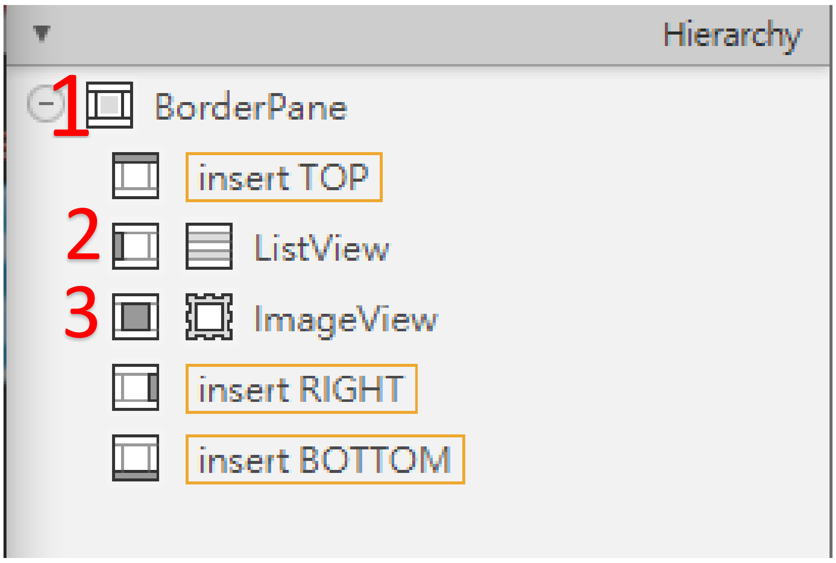
1. Cover Viewer Custom List View



GUI Description:



Hierarchy:



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

1) **BorderPane**
   a) Padding: 8 8 8 8 (TOP, RIGHT, BOTTOM, LEFT)
   b) Min Width: USE_COMPUTED_SIZE
   c) Min Height: USE_COMPUTED_SIZE
   d) Pref Width: USE_COMPUTED_SIZE
   e) Pref Height: USE_COMPUTED_SIZE
   f) Max Width: USE_COMPUTED_SIZE
   g) Max Height: USE_COMPUTED_SIZE

2) **ListView**
   a) Pref Height: USE_COMPUTED_SIZE
   b) Max Height: MAX_VALUE
   c) fx:id: booksListView

3) **ImageView**
   a) Fit Width: 370
   b) Fit Height: 480
   c) Fx:id: coverImageView

CoverViewer.java

```
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

```
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.ListCell;
import javafx.scene.control.ListView;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.util.Callback;

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();

   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()));
               }
            }
         );

      // set custom ListView cell factory
      booksListView.setCellFactory(
         new Callback<ListView<Book>, ListCell<Book>>() {
            @Override
            public ListCell<Book> call(ListView<Book> listView) {
               return new ImageTextCell();
            }
         }
      );
   }
}
```

Book.java

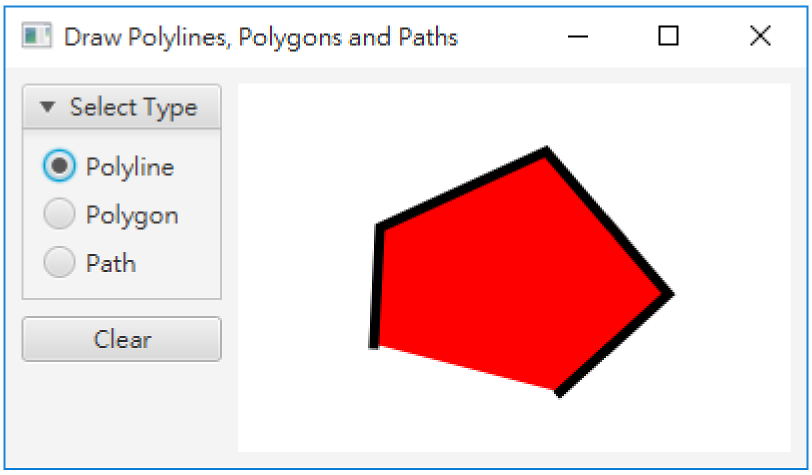
```
public class Book {
   private String title; // book title
   private String thumbImage; // source of book cover's thumbnail image
   private String largeImage; // source of book cover's full-size image
```

```java

   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();}
}
```

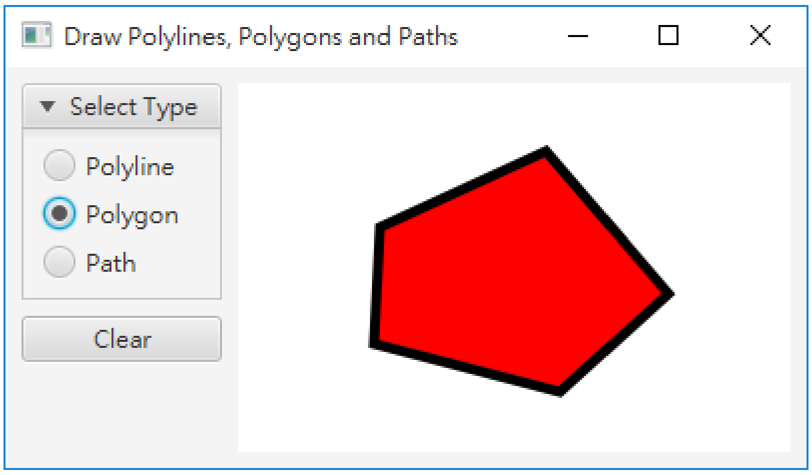
ImageTextCell.java

```java
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.control.ListCell;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.VBox;
import javafx.scene.text.TextAlignment;

public class ImageTextCell extends ListCell<Book> {
   private VBox vbox = new VBox(8.0); // 8 points of gap between controls
   private ImageView thumbImageView = new ImageView(); // initially empty
   private Label label = new Label();

   // constructor configures VBox, ImageView and Label
   public ImageTextCell() {
      vbox.setAlignment(Pos.CENTER); // center VBox contents horizontally

      thumbImageView.setPreserveRatio(true);
      thumbImageView.setFitHeight(100.0); // thumbnail 100 points tall
      vbox.getChildren().add(thumbImageView); // attach to Vbox

      label.setWrapText(true); // wrap if text too wide to fit in label
      label.setTextAlignment(TextAlignment.CENTER); // center text
      vbox.getChildren().add(label); // attach to VBox

      setPrefWidth(USE_PREF_SIZE); // use preferred size for cell width
   }

   // called to configure each custom ListView cell
   @Override
   protected void updateItem(Book item, boolean empty) {
      // required to ensure that cell displays properly
      super.updateItem(item, empty);

      if (empty || item == null) {
         setGraphic(null); // don't display anything
      }
      else {
         // set ImageView's thumbnail image
         thumbImageView.setImage(new Image(item.getThumbImage()));
         label.setText(item.getTitle()); // configure Label's text
         setGraphic(vbox); // attach custom layout to ListView cell
      }
   }
}
```

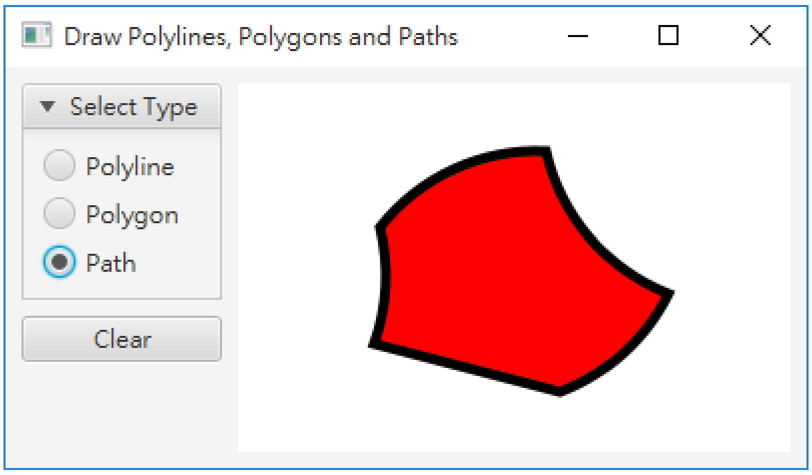
2. Poly Shapes



Select Type: [✓ Polyline] Polygon Path

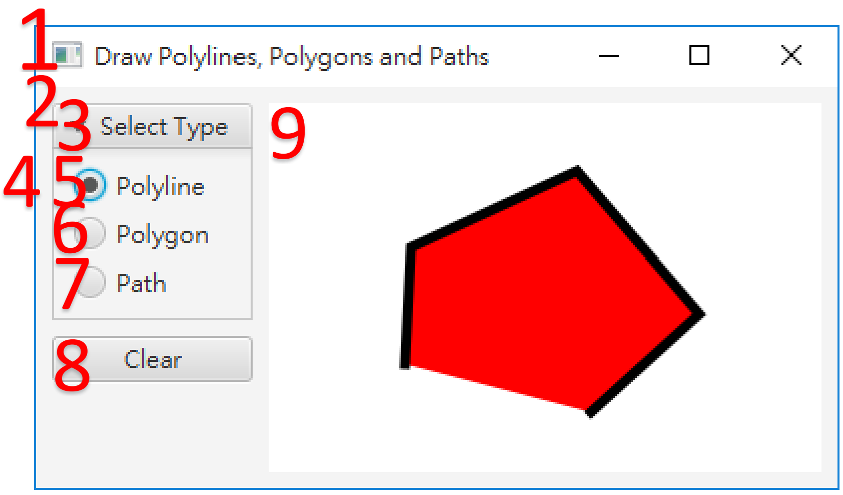


Select Type: Polyline [✓ Polygon] Path

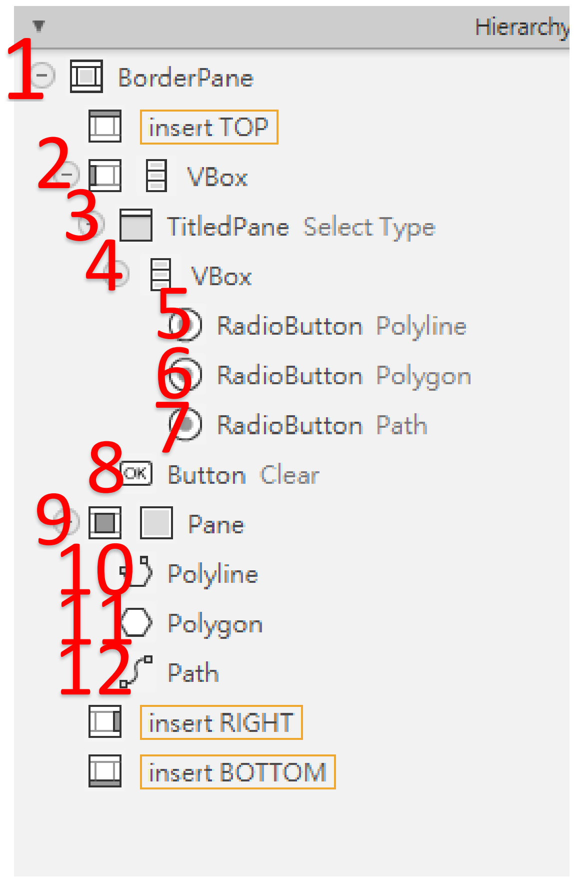


Select Type: Polyline Polygon [✓ Path]

GUI Description:



Hierarchy:



0) **File Name: PolyShapes.fxml**
   Controller Class: PolyShapesController

1) **BorderPane**
   a) Stylesheets: PolyShapes.css

   b) Padding: 8 8 8 8 (TOP, RIGHT, BOTTOM, LEFT)
   c) Min Width: USE_COMPUTED_SIZE
   d) Min Height: USE_COMPUTED_SIZE
   e) Pref Width: 400
   f) Pref Height: 200
   g) Max Width: USE_COMPUTED_SIZE
   h) Max Height: USE_COMPUTED_SIZE

2) **VBox**
   a) Spacing: 8

3) **TitledPane**
   a) Min Width: USE_COMPUTED_SIZE
   b) Min Height: USE_COMPUTED_SIZE
   c) Pref Width: USE_COMPUTED_SIZE
   d) Pref Height: USE_COMPUTED_SIZE
   e) Max Width: USE_COMPUTED_SIZE
   f) Max Height: USE_COMPUTED_SIZE

4) **VBox**
   a) Min Width: USE_COMPUTED_SIZE
   b) Min Height: USE_COMPUTED_SIZE
   c) Pref Width: USE_COMPUTED_SIZE
   d) Pref Height: USE_COMPUTED_SIZE
   e) Max Width: USE_COMPUTED_SIZE
   f) Max Height: USE_COMPUTED_SIZE

5) **RadioButton**
   a) Text: Polyline
   b) Selected: click
   c) Toggle Group: toggleGroup
   d) fx:id: polylineRadioButton
   e) On Action: shapeRadioButtonSelected

6) **RadioButton**
   a) Text: Polygon
   b) Toggle Group: toggleGroup
   c) fx:id: polygonRadioButton
   d) On Action: shapeRadioButtonSelected

7) **RadioButton**
   a) Text: Path
   b) Toggle Group: toggleGroup
   c) fx:id: pathRadioButton
   d) On Action: shapeRadioButtonSelected

8) **Button**
   a) Text: Clear
   b) Max Width: MAX_VALUE
   c) fx:id: clearButton
   d) On Action: clearButtonPressed

9) **Pane**
   a) Style: -fx-background-color white
   b) Margin: 0 0 0 8 (TOP, RIGHT, BOTTOM, LEFT)
   c) On Mouse Clicked: drawingAreaMouseClicked

10) **Polyline**
   a) Visible: unclick
   b) fx:id: polyline

11) **Polygon**
   a) Visible: unclick
   b) fx:id: polygon

12) **Path**
   a) Visible: unclick
   b) fx:id: polygon

PolyShapes.java

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

public class PolyShapes extends Application {
   @Override
   public void start(Stage stage) throws Exception {
```

```java
        Parent root =
            FXMLLoader.load(getClass().getResource("PolyShapes.fxml"));

        Scene scene = new Scene(root);
        stage.setTitle("Draw Polylines, Polygons and Paths");
        stage.setScene(scene);
        stage.show();
    }

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

PolyShapesController.java

```java
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.shape.ArcTo;
import javafx.scene.shape.ClosePath;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;
import javafx.scene.shape.Polygon;
import javafx.scene.shape.Polyline;

public class PolyShapesController {
    // enum representing shape types
    private enum ShapeType {POLYLINE, POLYGON, PATH};

    // instance variables that refer to GUI components
    @FXML private RadioButton polylineRadioButton;
    @FXML private RadioButton polygonRadioButton;
    @FXML private RadioButton pathRadioButton;
    @FXML private ToggleGroup toggleGroup;
    @FXML private Polyline polyline;
    @FXML private Polygon polygon;
```

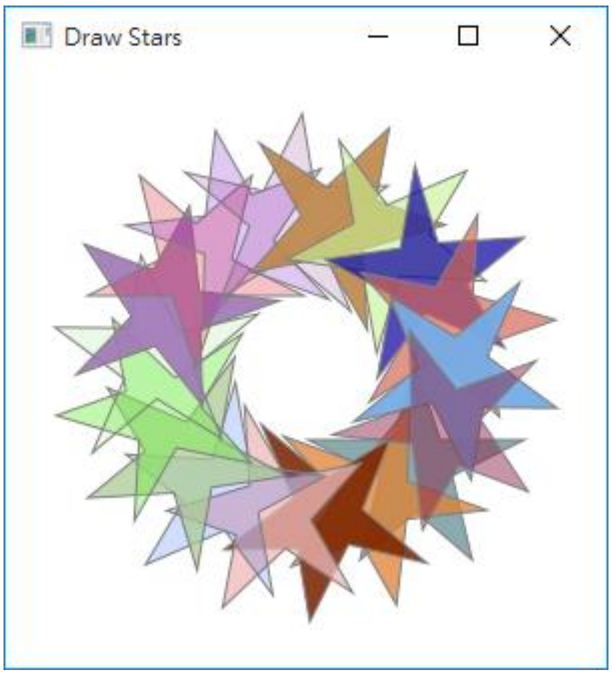
```
@FXML private Path path;

// instance variables for managing state
private ShapeType shapeType = ShapeType.POLYLINE;
private boolean sweepFlag = true; // used with arcs in a Path

// set user data for the RadioButtons and display polyline object
public void initialize() {
   // user data on a control can be any Object
   polylineRadioButton.setUserData(ShapeType.POLYLINE);
   polygonRadioButton.setUserData(ShapeType.POLYGON);
   pathRadioButton.setUserData(ShapeType.PATH);

   displayShape(); // sets polyline's visibility to true when app loads
   clearButtonPressed(null);
}

// handles drawingArea's onMouseClicked event
@FXML
private void drawingAreaMouseClicked(MouseEvent e) {
   polyline.getPoints().addAll(e.getX(), e.getY());
   polygon.getPoints().addAll(e.getX(), e.getY());

   // if path is empty, move to first click position and close path
   if (path.getElements().isEmpty()) {
      path.getElements().add(new MoveTo(e.getX(), e.getY()));
      path.getElements().add(new ClosePath());
   }
   else { // insert a new path segment before the ClosePath element
      // create an arc segment and insert it in the path
      ArcTo arcTo = new ArcTo();
      arcTo.setX(e.getX());
      arcTo.setY(e.getY());
      arcTo.setRadiusX(100.0);
      arcTo.setRadiusY(100.0);
      arcTo.setSweepFlag(sweepFlag);
      sweepFlag = !sweepFlag;
      path.getElements().add(path.getElements().size() - 1, arcTo);
   }
}
```

```java
   // handles color RadioButton's ActionEvents
   @FXML
   private void shapeRadioButtonSelected(ActionEvent e) {
      // user data for each color RadioButton is a ShapeType constant
      shapeType =
         (ShapeType) toggleGroup.getSelectedToggle().getUserData();
      displayShape(); // display the currently selected shape
   }

   // displays currently selected shape
   private void displayShape() {
      polyline.setVisible(shapeType == ShapeType.POLYLINE);
      polygon.setVisible(shapeType == ShapeType.POLYGON);
      path.setVisible(shapeType == ShapeType.PATH);
   }

   // resets each shape
   @FXML
   private void clearButtonPressed(ActionEvent event) {
      polyline.getPoints().clear();
      polygon.getPoints().clear();
      path.getElements().clear();
   }
}
```

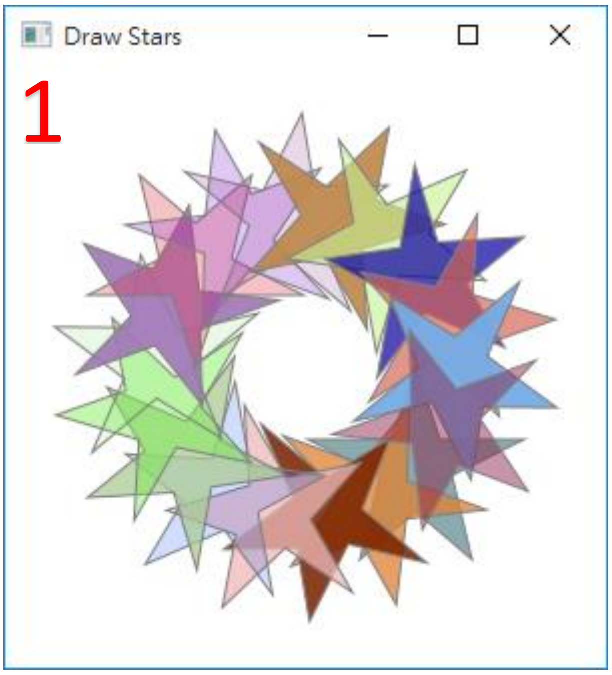
PolyShapes.css

```css
Polyline, Polygon, Path {
    -fx-stroke: black;
    -fx-stroke-width: 5;
    -fx-fill: red;
}
```

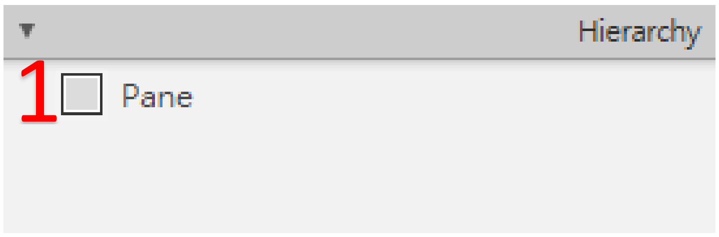
3. Cover Viewer Custom List View



GUI Description:



Hierarchy:



0) **File Name: DrawStars.fxml**
   Controller Class: DrawStarsController

1) **BorderPane**
   a) Min Width: USE_COMPUTED_SIZE
   b) Min Height: USE_COMPUTED_SIZE
   c) Pref Width: 300
   d) Pref Height: 300
   e) Max Width: USE_COMPUTED_SIZE
   f) Max Height: USE_COMPUTED_SIZE
   g) fx:id: pane

DrawStars.java

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

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

      Scene scene = new Scene(root);
      stage.setTitle("Draw Stars");
      stage.setScene(scene);
      stage.show();
   }
```

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

DrawStarsController.java

```java
import java.security.SecureRandom;
import javafx.fxml.FXML;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Polygon;
import javafx.scene.transform.Transform;

public class DrawStarsController {
   @FXML private Pane pane;
   private static final SecureRandom random = new SecureRandom();

   public void initialize() {
      // points that define a five-pointed star shape
      Double[] points = {205.0,150.0, 217.0,186.0, 259.0,186.0,
         223.0,204.0, 233.0,246.0, 205.0,222.0, 177.0,246.0, 187.0,204.0,
         151.0,186.0, 193.0,186.0};

      // create 18 stars
      for (int count = 0; count < 18; ++count) {
         // create a new Polygon and copy existing points into it
         Polygon newStar = new Polygon();
         newStar.getPoints().addAll(points);

         // create random Color and set as newStar's fill
         newStar.setStroke(Color.GREY);
         newStar.setFill(Color.rgb(random.nextInt(255),
            random.nextInt(255), random.nextInt(255),
            random.nextDouble()));

         // apply a rotation to the shape
         newStar.getTransforms().add(
            Transform.rotate(count * 20, 150, 150));
            pane.getChildren().add(newStar);
        }
    }
}
```