Java Programming

JavaFX Basics

Review of Lecture 6

- **Polymorphism:**
  - Used to **write code that does not depends on specific objects**.
    - Create a superclass object
    - Assign subclass objects to supperclass references
    - Call superclass method **overridden** in subclasses
  - Dynamic binding
  - Abstract class – incomplete methods (abstract)

    public abstract void draw();
  - Used only **as superclasses in inheritance hierarchies.**
- Cannot instantiate objects of abstract superclasses.
- Polymorphism can be implemented with abstract superclasses and concrete classes that **override** abstract methods of the superclass.
- instanceOf operator
- getClass method of Object class
- A final method in a superclass cannot be overridden in a subclass

Review of Lecture 6

- A final class cannot be extended to create a subclass.
- All methods in a final class are implicitly final.
- Class **String** is an example of a final class.
- A Java interface **describes a set of methods that can be called on an object**.
- Interfaces offer a capability requiring that **unrelated classes implement a set of common methods.**
- All methods declared in an interface are **implicitly public abstract methods**.
- All fields are **implicitly public**, static and final.
- A concrete class must specify that it implements the interface and must implement each method in the interface with specified signature.
- In Java SE 8, interfaces also may contain public default methods with concrete default implementations.

Lesson 7 Objectives

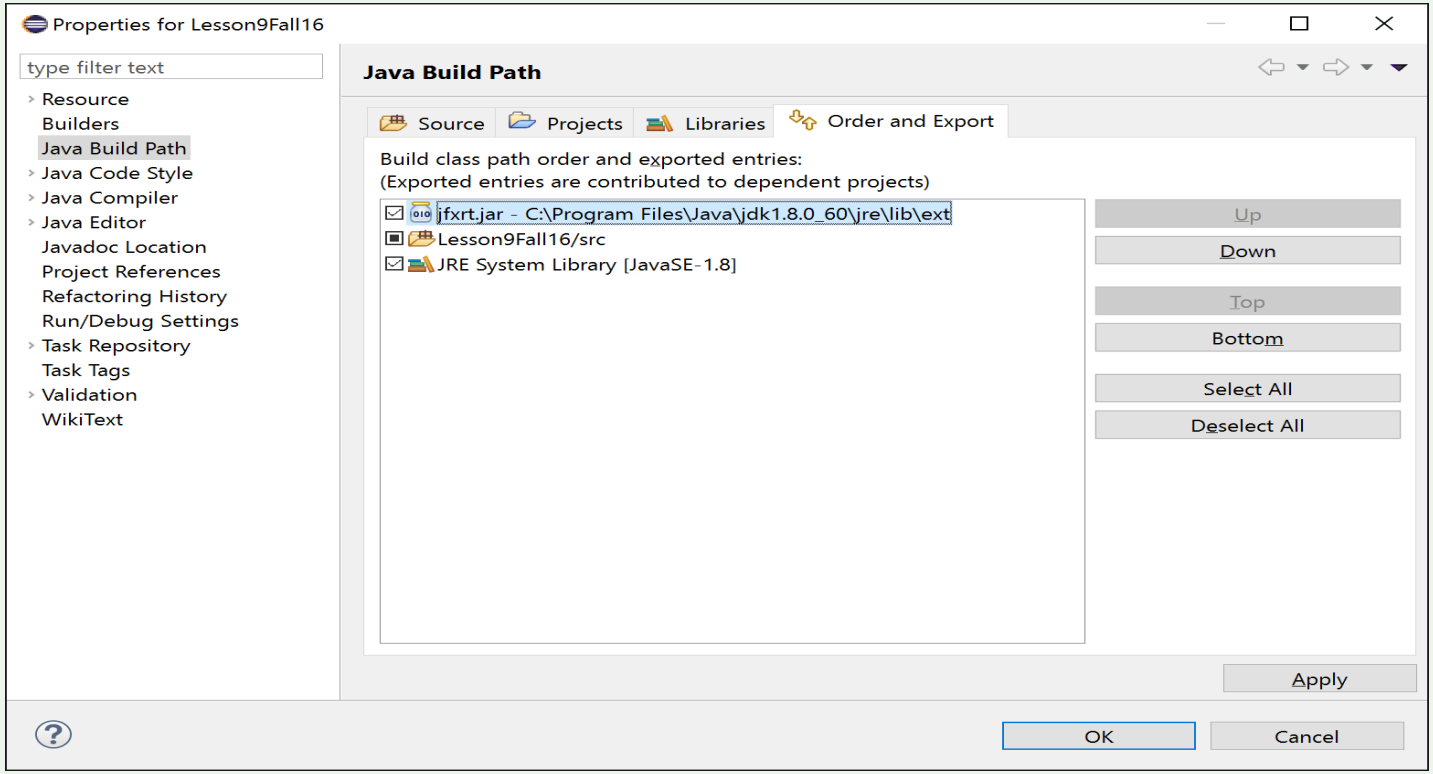
- Explain JavaFX.
- Develop JavaFX application using layout panes, buttons, labels, text fields, colors, fonts, images, image views, and shapes.

Introduction to JavaFX

- JavaFX is a new framework for developing Java GUI programs.
  - Allows the use of **powerful Java features**, such as generics, annotations, multithreading, and Lamda Expressions (introduced in Java SE 8).
  - Makes it easier for Web developers to **use JavaFX from other JVM-based dynamic languages**, such as Groovy and JavaScript.
  - Allows Java developers to **use other system languages**, such as Groovy, for writing large or complex JavaFX applications.

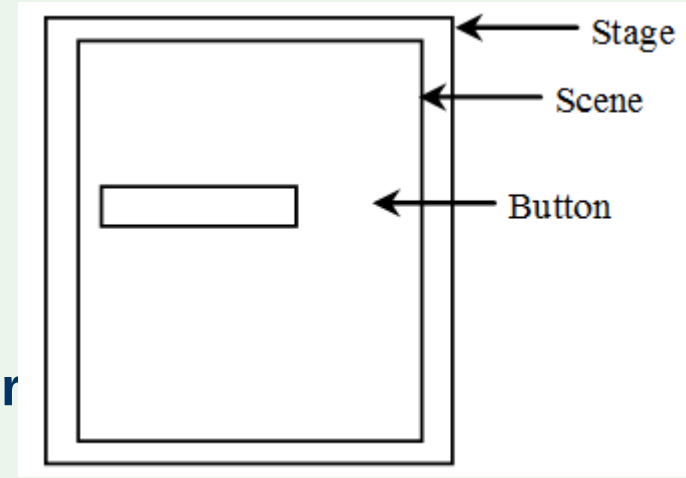
Configuring JavaFX in Eclipse Neone

- See my document on eCentennial:



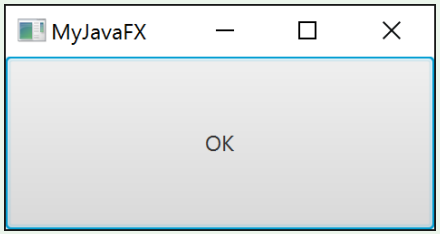
Anatomy of a JavaFX application

- The main class for a JavaFX application extends the javafx.application.**Application** class.
- The **start()** method is the **main entry** point for all JavaFX applications.
- A JavaFX application defines the **user interface container** by means of a **stage** and a **scene**.
  - The JavaFX **Stage** class is the top-level JavaFX container.
  - The JavaFX **Scene** class is the container for all content.

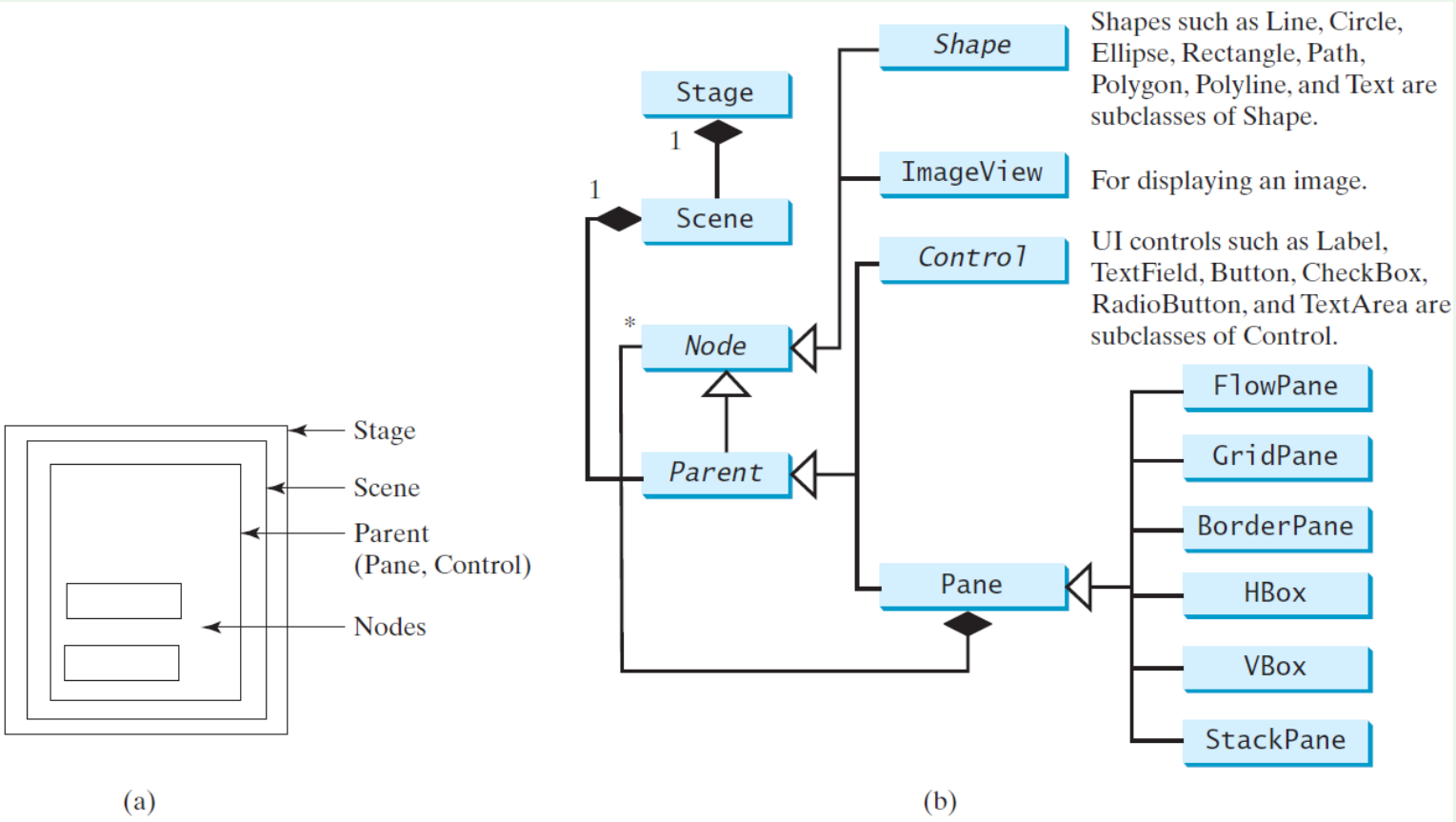


Anatomy of a JavaFX application

```
public class MyJavaFX extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a button and place it in the scene
    Button btOK = new Button("OK");
    Scene scene = new Scene(btOK, 200, 250);
    primaryStage.setTitle("MyJavaFX"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```



JavaFX Layout classes



Layout Panes

- JavaFX provides many types of panes for organizing nodes in a container.

Class	*Description*
`Pane`	Base class for layout panes. It contains the `getChildren()` method for returning a list of nodes in the pane.
`StackPane`	Places the nodes on top of each other in the center of the pane.
`FlowPane`	Places the nodes row-by-row horizontally or column-by-column vertically.
`GridPane`	Places the nodes in the cells in a two-dimensional grid.
`BorderPane`	Places the nodes in the top, right, bottom, left, and center regions.
`HBox`	Places the nodes in a single row.
`VBox`	Places the nodes in a single column.

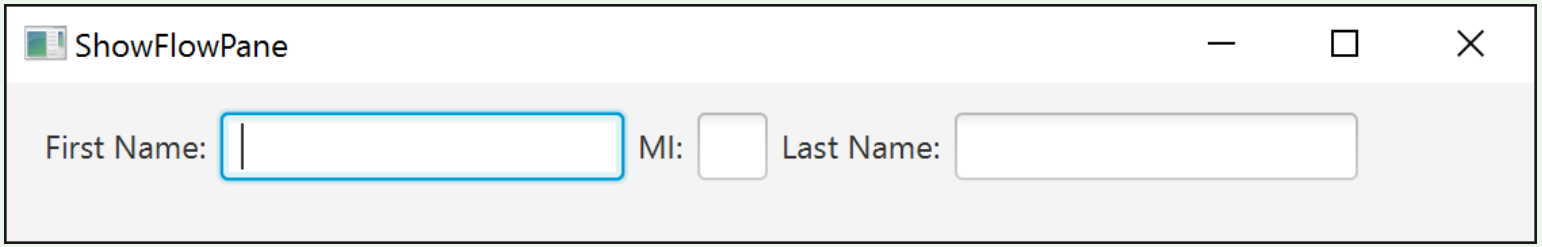
FlowPane layout

- **Lays out its children in a flow that wraps at the flowpane's boundary**

```
public class ShowFlowPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    FlowPane pane = new FlowPane();
    pane.setPadding(new Insets(11, 12, 13, 14));
    pane.setHgap(5);
    pane.setVgap(5);
    // Place nodes in the pane
    pane.getChildren().addAll(new Label("First Name:"),
      new TextField(), new Label("MI:"));
    TextField tfMi = new TextField();
    tfMi.setPrefColumnCount(1);
    pane.getChildren().addAll(tfMi, new Label("Last Name:"), new TextField());
```

FlowPane layout

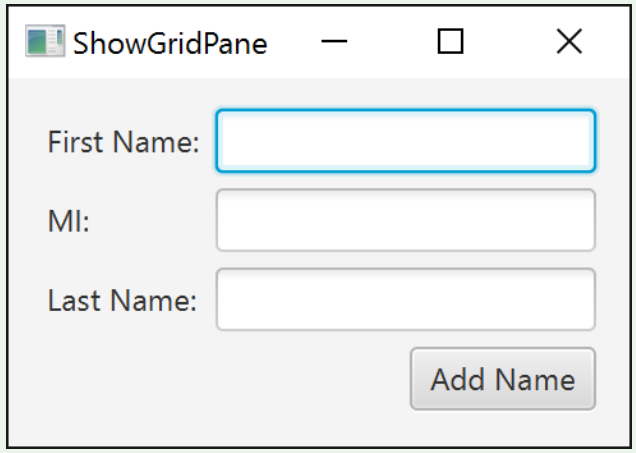
```
// Create a scene and place it in the stage
    Scene scene = new Scene(pane, 200, 250);
    primaryStage.setTitle("ShowFlowPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
public static void main(String[] args) {
    launch(args);
  }
}
```



GridPane layout

- **Lays out its children within a flexible grid of rows and columns**

```
public class ShowGridPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    GridPane pane = new GridPane();
    pane.setAlignment(Pos.CENTER);
    pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
    pane.setHgap(5.5);
    pane.setVgap(5.5);
    // Place nodes in the pane
    pane.add(new Label("First Name:"), 0, 0);
    pane.add(new TextField(), 1, 0);
    pane.add(new Label("MI:"), 0, 1);
    pane.add(new TextField(), 1, 1);
    pane.add(new Label("Last Name:"), 0, 2);
    pane.add(new TextField(), 1, 2);
```



GridPane layout

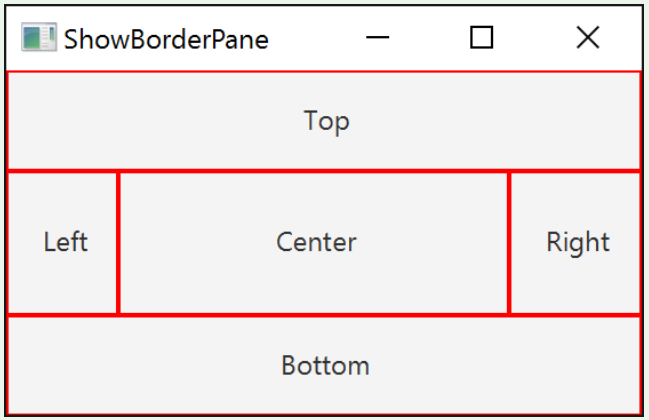
```
    Button btAdd = new Button("Add Name");
    pane.add(btAdd, 1, 3);
    GridPane.setHalignment(btAdd, HPos.RIGHT);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowGridPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
public static void main(String[] args) {
    launch(args);
  }
}
```

BorderPane layout

- **Lays out children in top, left, right, bottom, and center positions**

```
public class ShowBorderPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a border pane
    BorderPane pane = new BorderPane();
    // Place nodes in the pane
    pane.setTop(new CustomPane("Top"));
    pane.setRight(new CustomPane("Right"));
    pane.setBottom(new CustomPane("Bottom"));
    pane.setLeft(new CustomPane("Left"));
    pane.setCenter(new CustomPane("Center"));
```



BorderPane layout

```
// Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowBorderPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}
// Define a custom pane to hold a label in the center of the pane
class CustomPane extends StackPane {
  public CustomPane(String title) {
    getChildren().add(new Label(title));
    setStyle("-fx-border-color: red");
    setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
  }
}
```

XBox layout

- **Lays out its children in a single horizontal row:**

```
public class ShowHBoxVBox extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a border pane
    BorderPane pane = new BorderPane();
    // Place nodes in the pane
    pane.setTop(getHBox());
    pane.setLeft(getVBox());
    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowHBoxVBox"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```

XBox layout

```
private HBox getHBox() {
    HBox hBox = new HBox(15);
    hBox.setPadding(new Insets(15, 15, 15, 15));
    hBox.setStyle("-fx-background-color: gold");
    hBox.getChildren().add(new Button("Computer Science"));
    hBox.getChildren().add(new Button("Chemistry"));
    ImageView imageView = new ImageView(new Image("images/ca.gif"));
    hBox.getChildren().add(imageView);
    return hBox;
  }
```

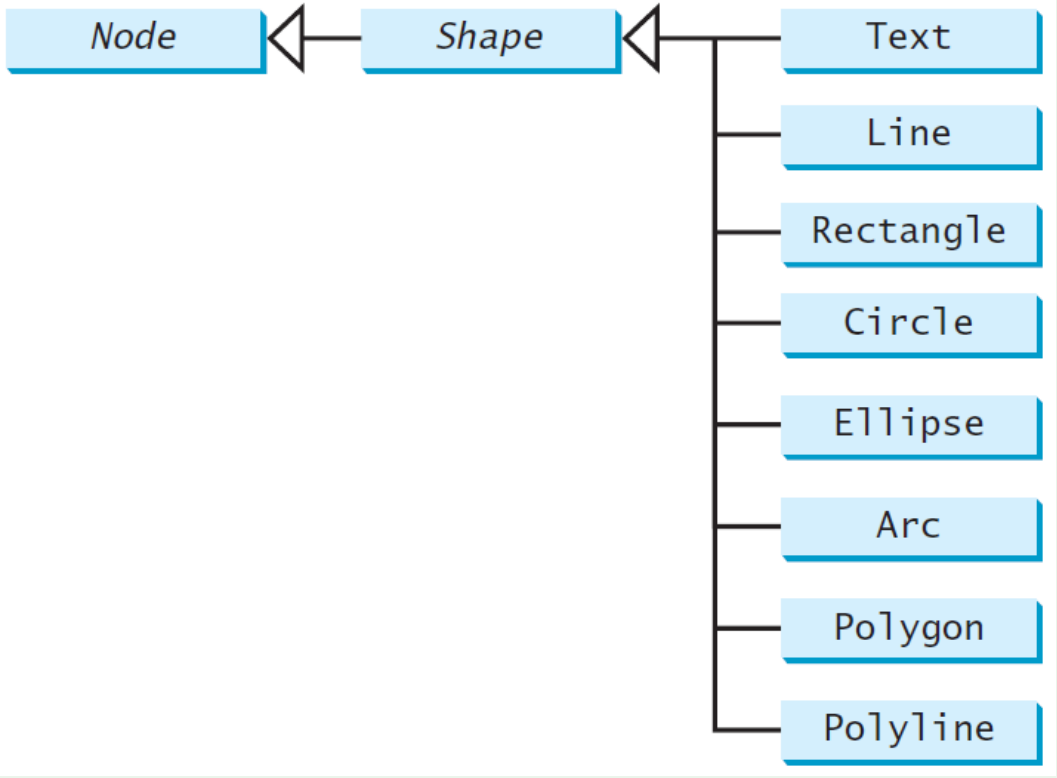
Vbox layout

- **Lays out its children in a single vertical column:**

```
private VBox getVBox() {
    VBox vBox = new VBox(15);
    vBox.setPadding(new Insets(15, 5, 5, 5));
    vBox.getChildren().add(new Label("Courses"));
    Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
        new Label("CSCI 2410"), new Label("CSCI 3720")};
    for (Label course: courses) {
      VBox.setMargin(course, new Insets(0, 0, 0, 15));
      vBox.getChildren().add(course);
    }
    return vBox;
  }
public static void main(String[] args) {
    launch(args);
  }
}
```

Shapes

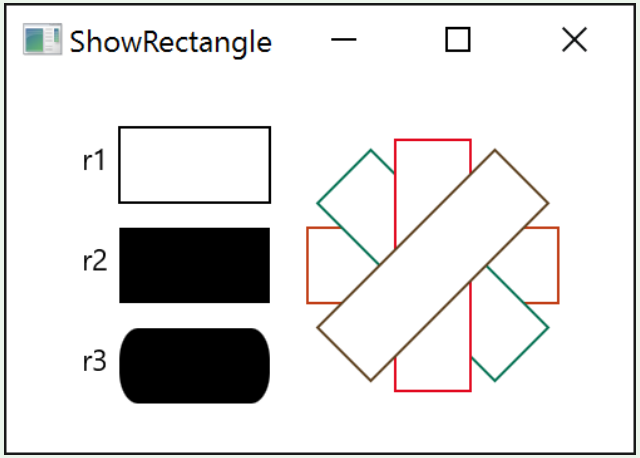
- JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.



Rectangle example

```
public class ShowRectangle extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create rectangles
    Rectangle r1 = new Rectangle(25, 10, 60, 30);
    r1.setStroke(Color.BLACK);
    r1.setFill(Color.WHITE);
    Rectangle r2 = new Rectangle(25, 50, 60, 30);
    Rectangle r3 = new Rectangle(25, 90, 60, 30);
    r3.setArcWidth(15);
    r3.setArcHeight(25);

    // Create a group and add nodes to the group
    Group group = new Group();
    group.getChildren().addAll(new Text(10, 27, "r1"), r1,
      new Text(10, 67, "r2"), r2, new Text(10, 107, "r3"), r3);
```



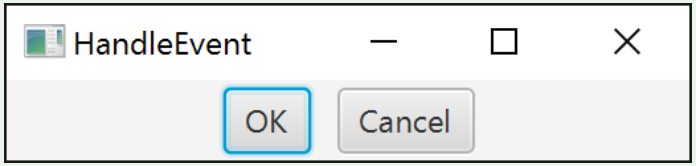
Rectangle example

```
for (int i = 0; i < 4; i++) {
      Rectangle r = new Rectangle(100, 50, 100, 30);
      r.setRotate(i * 360 / 8);
      r.setStroke(Color.color(Math.random(), Math.random(), Math.random()));
      r.setFill(Color.WHITE);
      group.getChildren().add(r);
    }

    // Create a scene and place it in the stage
    Scene scene = new Scene(new BorderPane(group), 250, 150);
    primaryStage.setTitle("ShowRectangle"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```

Event Handling in JavaFX

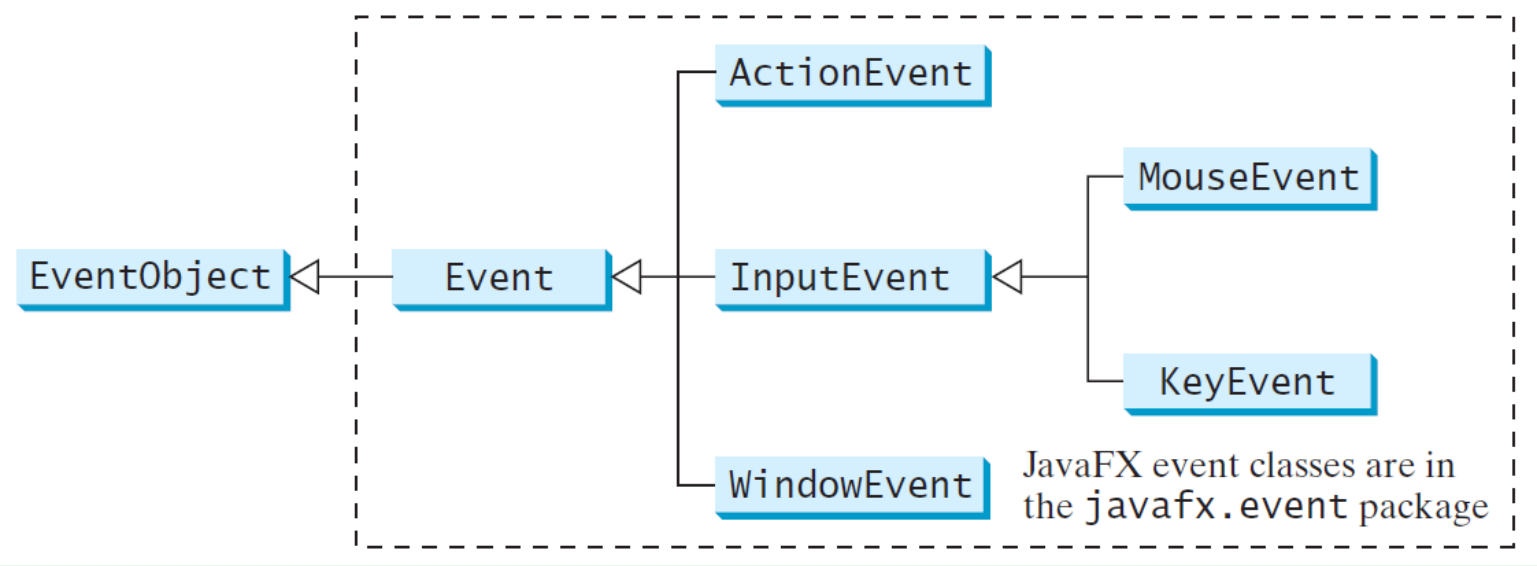
```
public class HandleEvent extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    HBox pane = new HBox(10);
    pane.setAlignment(Pos.CENTER);
    Button btOK = new Button("OK");
    Button btCancel = new Button("Cancel");
    OKHandlerClass handler1 = new OKHandlerClass();
    btOK.setOnAction(handler1);
    CancelHandlerClass handler2 = new CancelHandlerClass(); //create handler object
    btCancel.setOnAction(handler2); //register the handler object
    pane.getChildren().addAll(btOK, btCancel); //add buttons to the scene
```



Event Handling in JavaFX

```
    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("HandleEvent"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
class OKHandlerClass implements EventHandler<ActionEvent> {
  @Override
  public void handle(ActionEvent e) {     System.out.println("OK button clicked");   }
}
class CancelHandlerClass implements EventHandler<ActionEvent> {
  @Override
  public void handle(ActionEvent e) { System.out.println("Cancel button clicked"); }
}
```

Event classes in JavaFX

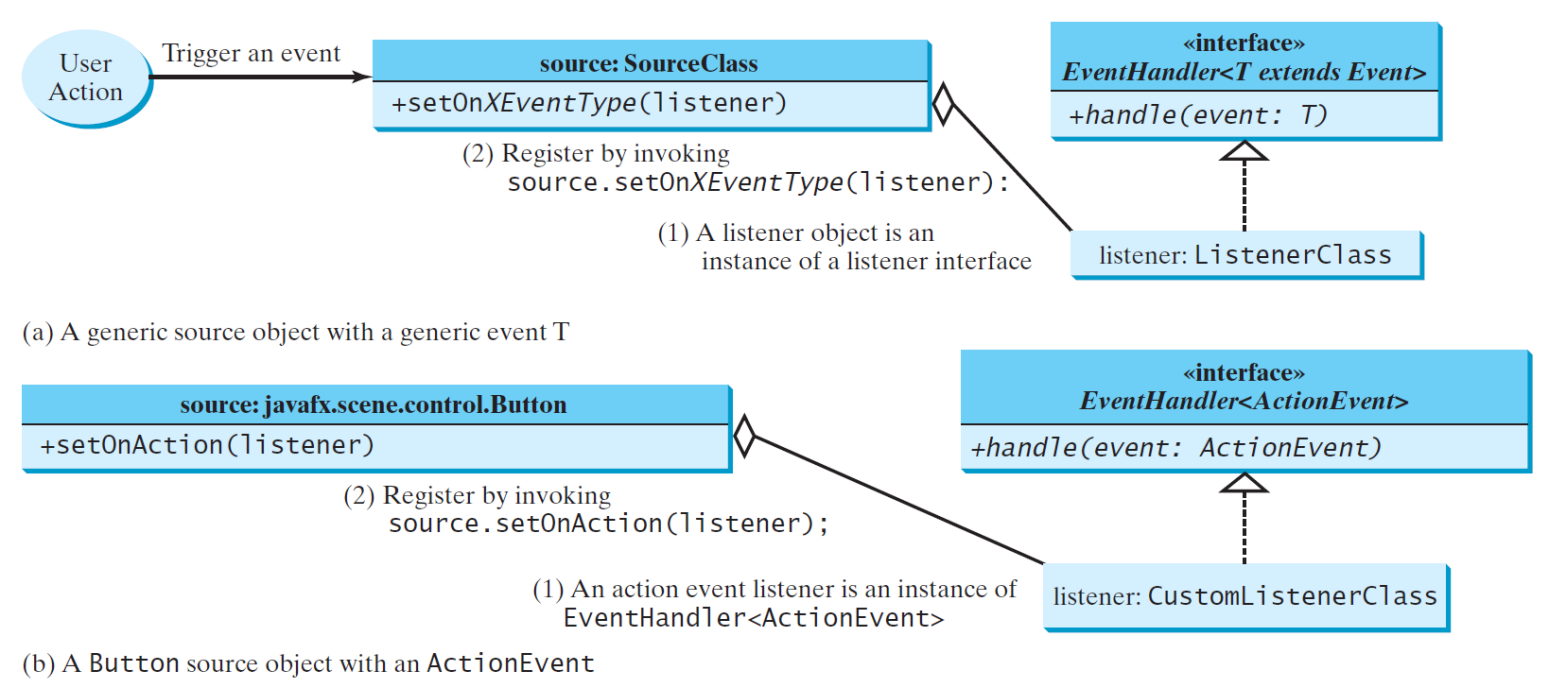


- You can identify the source object of the event using the getSource() instance method in the EventObject class.

Selected User Actions and Handlers

User Action	*Source Object*	*Event Type Fired*	*Event Registration Method*
Click a button	`Button`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Press Enter in a text field	`TextField`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Check or uncheck	`RadioButton`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Check or uncheck	`CheckBox`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Select a new item	`ComboBox`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Mouse pressed	`Node, Scene`	`MouseEvent`	`setOnMousePressed(EventHandler<MouseEvent>)`
Mouse released			`setOnMouseReleased(EventHandler<MouseEvent>)`
Mouse clicked			`setOnMouseClicked(EventHandler<MouseEvent>)`
Mouse entered			`setOnMouseEntered(EventHandler<MouseEvent>)`
Mouse exited			`setOnMouseExited(EventHandler<MouseEvent>)`
Mouse moved			`setOnMouseMoved(EventHandler<MouseEvent>)`
Mouse dragged			`setOnMouseDragged(EventHandler<MouseEvent>)`
Key pressed	`Node, Scene`	`KeyEvent`	`setOnKeyPressed(EventHandler<KeyEvent>)`
Key released			`setOnKeyReleased(EventHandler<KeyEvent>)`
Key typed			`setOnKeyTyped(EventHandler<KeyEvent>)`

The Delegation Model




```
Button btOK = new Button("OK");
OKHandlerClass handler = new OKHandlerClass();
btOK.setOnAction(handler);
```

Lambda Expressions

- Lambda expression is a new feature in Java 8. Lambda expressions can be viewed as an anonymous method with a concise syntax.
- For example, the following code in (a) can be greatly simplified using a lambda expression in (b) in three lines.

```
btEnlarge.setOnAction(
  new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent e) {
      // Code for processing event e
    }
  }
});
```

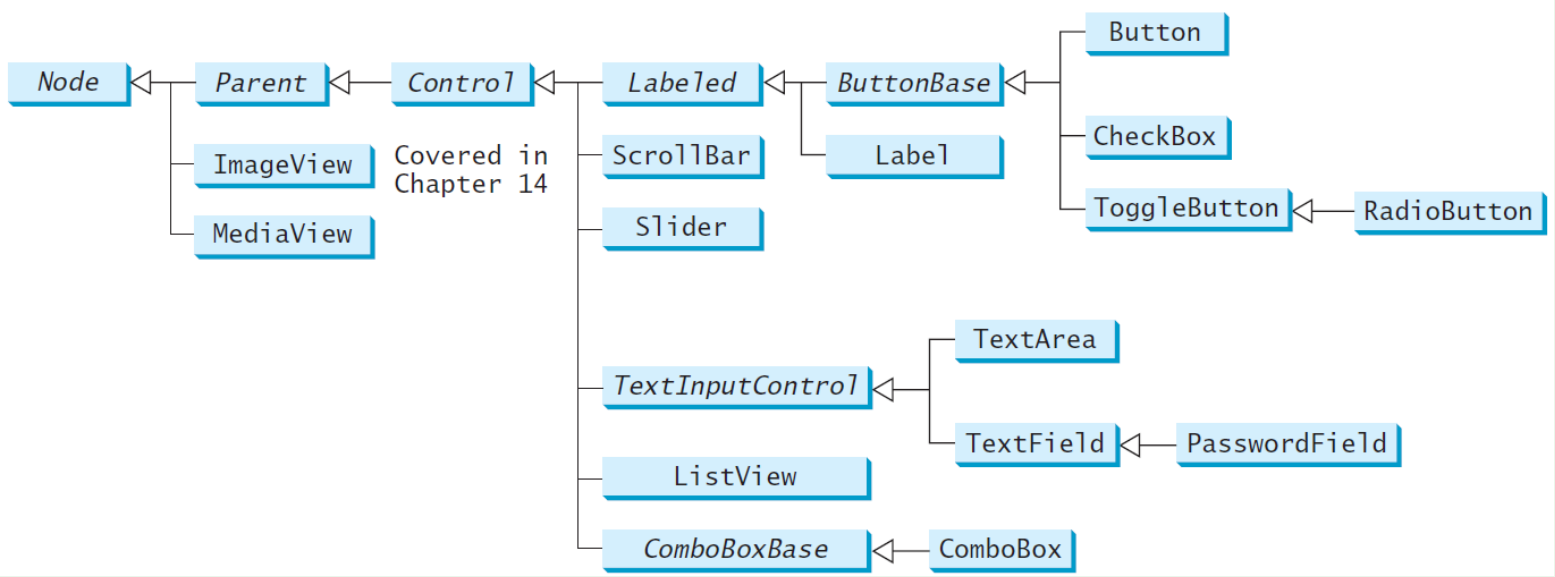
(a) Anonymous inner class event handler

```
btEnlarge.setOnAction(e -> {
  // Code for processing event e
});
```

(b) Lambda expression event handler

JavaFX GUI classes

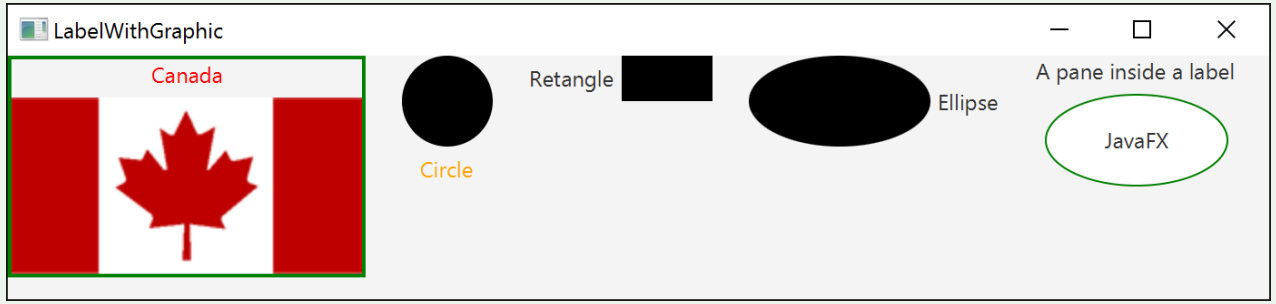
- The JavaFX UI controls are built by **using nodes in the scene graph**.
- They can take full advantage of the **visually rich** features of the JavaFX platform and are **portable** across different platforms.
- JavaFX CSS allows for **theming** and **skinning** of the UI controls.
- These controls reside in the **javafx.scene.control** package.



Using Label class

- **Used to display a text element and/or image**

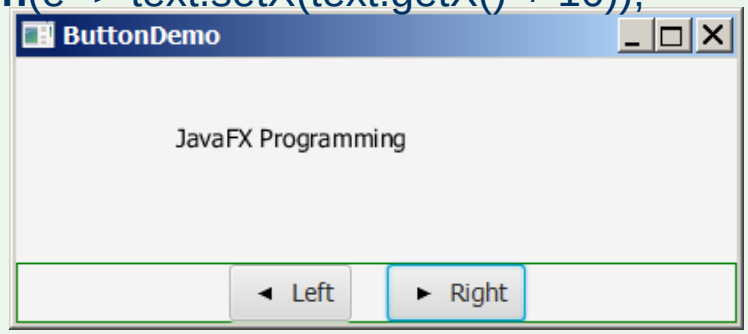
```
ImageView ca = new ImageView(new Image("image/ca.gif"));
Label lb1 = new Label("Canada", ca);
lb1.setStyle("-fx-border-color: green; -fx-border-width: 2");
lb1.setContentDisplay(ContentDisplay.BOTTOM);
lb1.setTextFill(Color.RED);
HBox pane = new HBox(20);
pane.getChildren().add(lb1);
……
```



Button class

- **Enables developers to process an action when a user clicks a button**

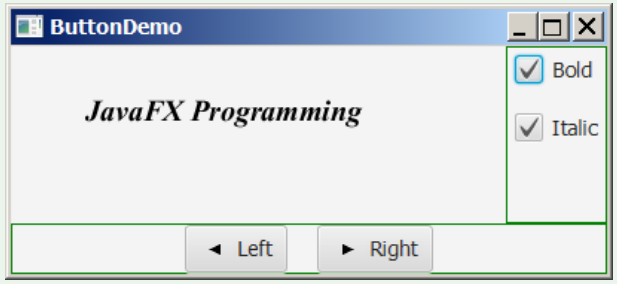
```
HBox paneForButtons = new HBox(20);
Button btLeft = new Button("Left", new ImageView("image/left.gif"));
Button btRight = new Button("Right",  new ImageView("image/right.gif"));
paneForButtons.getChildren().addAll(btLeft, btRight);
BorderPane pane = new BorderPane();
pane.setBottom(paneForButtons);
btLeft.setOnAction(e -> text.setX(text.getX() - 10));
btRight.setOnAction(e -> text.setX(text.getX() + 10));
……..
```



CheckBox class

- **Allow users to select or deselect a set of choices**

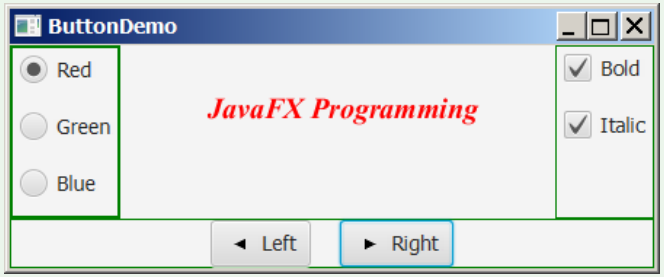
```
VBox paneForCheckBoxes = new VBox(20);
paneForCheckBoxes.setPadding(new Insets(5, 5, 5, 5));
paneForCheckBoxes.setStyle("-fx-border-color: green");
CheckBox chkBold = new CheckBox("Bold");
CheckBox chkItalic = new CheckBox("Italic");
paneForCheckBoxes.getChildren().addAll(chkBold, chkItalic);
pane.setRight(paneForCheckBoxes);
……..
```



RadioButton class

- **Allow users to select among mutually exclusive choices:**

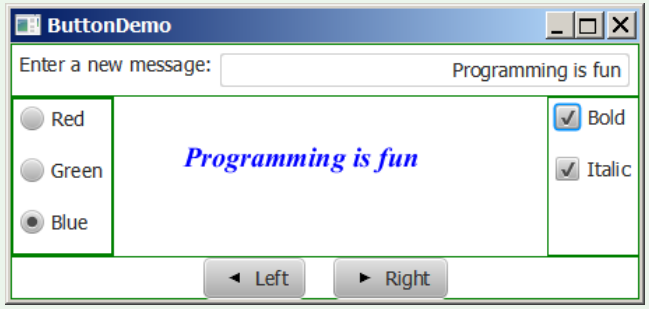
```
VBox paneForRadioButtons = new VBox(20);
paneForRadioButtons.setPadding(new Insets(5, 5, 5, 5));
paneForRadioButtons.setStyle("-fx-border-color: green");
paneForRadioButtons.setStyle("-fx-border-width: 2px; -fx-border-color: green");
RadioButton rbRed = new RadioButton("Red");
RadioButton rbGreen = new RadioButton("Green");
RadioButton rbBlue = new RadioButton("Blue");
paneForRadioButtons.getChildren().addAll(rbRed, rbGreen, rbBlue);
pane.setLeft(paneForRadioButtons);
……..
```



TextField class

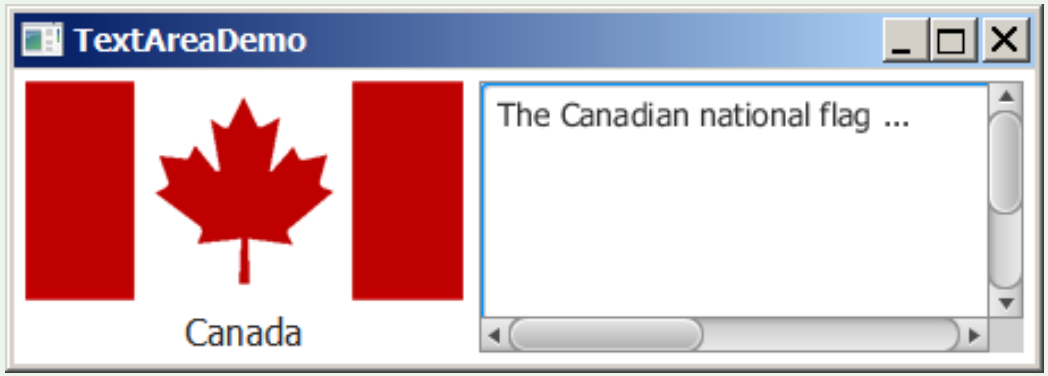
- **Used to accept and display text input:**

```
BorderPane paneForTextField = new BorderPane();
paneForTextField.setPadding(new Insets(5, 5, 5, 5));
paneForTextField.setStyle("-fx-border-color: green");
paneForTextField.setLeft(new Label("Enter a new message: "));
TextField tf = new TextField();
tf.setAlignment(Pos.BOTTOM_RIGHT);
paneForTextField.setCenter(tf);
pane.setTop(paneForTextField);
tf.setOnAction(e -> text.setText(tf.getText()));
……
```



TextArea class

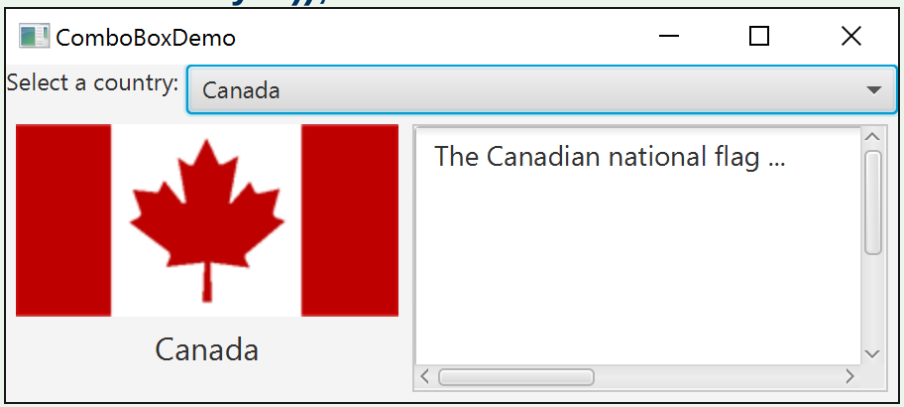
```
TextArea taDescription = new TextArea();
ScrollPane scrollPane = new ScrollPane(taDescription);
setCenter(scrollPane);
setPadding(new Insets(5, 5, 5, 5));
……
```



ComboBox

- **Enables users to choose one of several options**
- **Define the items as an observable list**

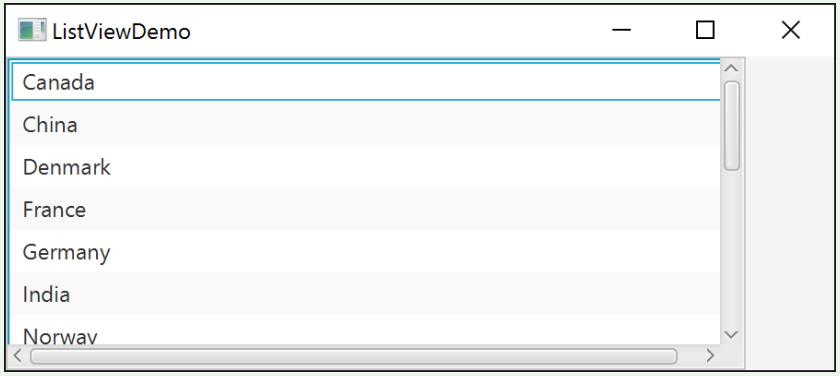
```
ComboBox<String> cbo = new ComboBox<>();
ObservableList<String> options = FXCollections.observableArrayList("Canada","USA");
BorderPane paneForComboBox = new BorderPane();
paneForComboBox.setLeft(new Label("Select a country: "));
paneForComboBox.setCenter(cbo);
pane.setTop(paneForComboBox);
cbo.setPrefWidth(400);
cbo.setValue("Canada");
cbo.getItems().addAll(items);
```




```
cbo.setOnAction(e -> setDisplay(items.indexOf(cbo.getValue())));
……
```

ListView - scrollable list of items

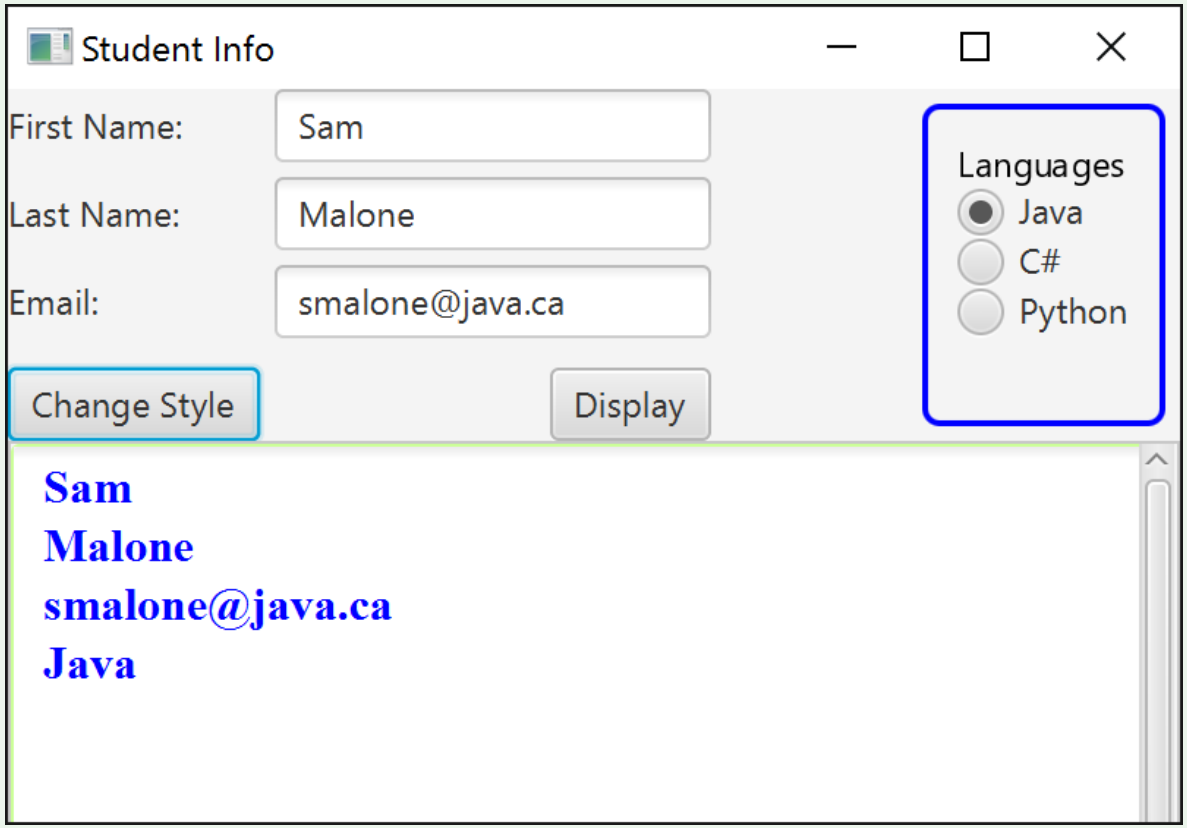
```
ListView<String> lv = new ListView<> (FXCollections.observableArrayList(flagTitles));
lv.setPrefSize(400, 400);
lv.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
FlowPane imagePane = new FlowPane(10, 10);
BorderPane pane = new BorderPane();
pane.setLeft(new ScrollPane(lv));
pane.setCenter(imagePane);
lv.getSelectionModel().selectedItemProperty().addListener(ov -> {
imagePane.getChildren().clear();
      for (Integer i: lv.getSelectionModel().getSelectedIndices()) {
        imagePane.getChildren().add(ImageViews[i]);
      }
  });
```



UI Controls example

- Use a BorderPane for the Scene layout
- Use a GridPane to create the entries form
- Use Label, TextField, Button controls for the form
- Use Radio buttons for mutually exclusive options
- Use TextArea to display the entries
- Use a ScrollPane to provide scrolling abilities.
- Use styles to provide change the format of controls and text.
- Use lambda expressions to handle events

UI Controls example



References

- Textbook
- JavaFX documentation: http://docs.oracle.com/javase/8/javase-clienttechnologies.htm
- http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/ui_controls.htm#JFXUI336
- http://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
- Liang, Introduction to Java Programming, 10th edition, 2015, Pearson Education.