CSE 215: Programming Language II Lab
Lab 16
Focusing on: JavaFX, GUI Components

JavaFX is a set of Java graphics libraries for creating Java GUI applications, just like Java AWT and Swing. It brings many new features on the table compared to its predecessors, and is a lot more flexible.

JavaFX is huge, with 36 packages. These are the commonly-used packages:

- javafx.application: JavaFX application. This is the core of the GUI application created with JavaFX.
- javafx.stage: top-level container. A stage is basically the window which holds all the GUI components (container, buttons, textboxes, radio buttons etc.)
- javafx.scene: scene and scene graph. In JavaFX, the components of the GUI application is treated as a tree. The root of the graph is the container. Each subsequent element is becomes the children of the root.
- javafx.scene.*: control, layout, shape, etc.
- javafx.event: event handling. Clicking a button, typing text in a text box is treated as an event. When such an event occurs, the necessary business logic to deal with the event is specified by the programmer. Thus, JavaFX makes use of **event handling** and **event driven programming**.
- javafx.animation: for animation.

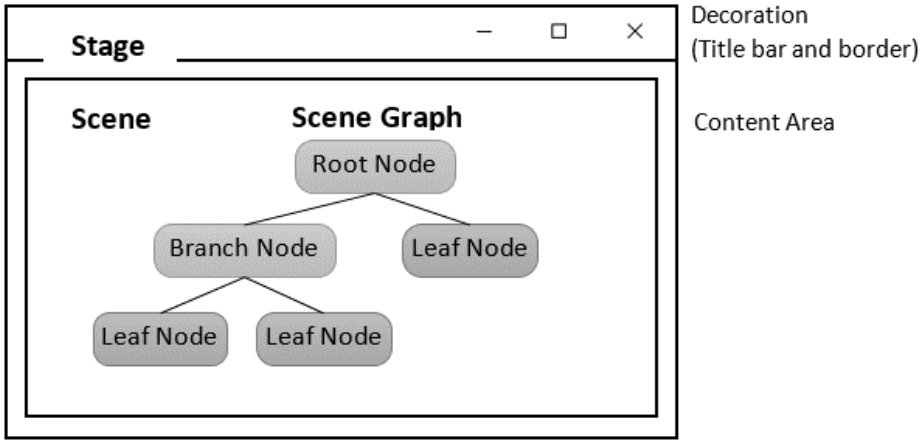


Figure 1: The tree structure of the GUI application

The components in a JavaFX GUI application is organized in a way similar to a movie theatre.

Stage: Stage is the window in which the application resides. It is the base.
Scene: Scene is analogous to the current layout that is being displayed to the user. There may be multiple scenes (e.g. various dialog boxes based on what action the user performs)
Scene graph: These can be thought of as the actors. The UI components (see Table 1) are the actors.

Table 1: UI Components under javafx.scene.control

1	**Label** A Label object is a component for placing text.
2	**Button** This class creates a labeled button.
3	**ColorPicker** A ColorPicker provides a pane of controls designed to allow a user to manipulate and select a color.
4	**CheckBox** A CheckBox is a graphical component that can be in either an on(true) or off (false) state.

5	**RadioButton** The RadioButton class is a graphical component, which can either be in a ON (true) or OFF (false) state in a group. RadioButtons in GUI applications typically represent mutually exclusive options, i.e. only one can be true at a time.
6	**ToggleGroup** Used in conjunction with radio buttons so that only one radio button is selected at a given time.
6	**ListView** A ListView component presents the user with a scrolling list of text items.
7	**TextField** A TextField object is a text component that allows for the editing of a single line of text.
8	**PasswordField** A PasswordField object is a text component specialized for password entry.
9	**Scrollbar** A Scrollbar control represents a scroll bar component in order to enable user to select from a range of values.
10	**FileChooser** A FileChooser control represents a dialog window from which the user can select a file.
11	**ProgressBar** As the task progresses towards completion, the progress bar displays the task's percentage of completion.
12	**Slider** A Slider lets the user graphically select a value by sliding a knob within a bounded interval.

Table 2: Shape components under javafx.scene.shape

1	**Line** A line is a geometrical structure joining two point.
2	**Rectangle** In general, a rectangle is a four-sided polygon that has two pairs of parallel and concurrent sides with all interior angles as right angles. In JavaFX, a Rectangle is represented by a class named **Rectangle**.
3	**Rounded Rectangle** In JavaFX, you can draw a rectangle either with sharp edges or with arched edges and The one with arched edges is known as a rounded rectangle.
4	**Circle** A circle is a line forming a closed loop, every point on which is a fixed distance from a centre point. In JavaFX, a circle is represented by a class named **Circle**.
5	**Ellipse** An ellipse is defined by two points, each called a focus. If any point on the ellipse is taken, the sum of the distances to the focus points is constant. The size of the ellipse is determined by the sum of these two distances. In JavaFX, an ellipse is represented by a class named **Ellipse**.
6	**Polygon** A closed shape formed by a number of coplanar line segments connected end to end. In JavaFX, a polygon is represented by a class named **Polygon**.
7	**Polyline** A polyline is same a polygon except that a polyline is not closed in the end. Or, continuous line composed of one or more line segments. In JavaFX, a Polyline is represented by a class named **Polygon**.
8	**Arc** An arc is part of a curve. In JavaFX, an arc is represented by a class named **Arc**. **Types Of Arc** In addition to this we can draw three types of arc's **Open, Chord, Round**.

Rules of creating a JavaFX application:

1. It must extend the `Application` class under `javafx.application.Application`
2. It must implement the method `void start(Stage primaryStage).`

Creating your very first JavaFX GUI Layout

Let's assume we want to place a text box, two buttons and a label in the scene.

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
import javafx.geometry.Insets;

public class HelloJavaFXWorld extends Application {
	public void start(Stage primaryStage) {
		// create a label
		Label txtBoxLabel = new Label("Sample Label");
		// create a text field
		TextField textField = new TextField();

		// create two buttons
		Button okBtn = new Button("OK");
		Button cancelBtn = new Button("Cancel");

		// put them into a Pane, more on Panes later
		FlowPane flowPane = new FlowPane();
		// define padding around the pane
		flowPane.setPadding(new Insets(20, 20, 20, 20));
		// set horizontal spacing between the elements
		flowPane.setHgap(20);
		// add the nodes to the pane
		flowPane.getChildren().addAll(txtBoxLabel, textField, okBtn, cancelBtn);

		// construct a scene with the given pane
		Scene scene = new Scene(flowPane);

		// define properties of the stage
		primaryStage.setTitle("Hello World of JAVAFX");
		// set the scene defined above as the scene to be shown in the stage
		primaryStage.setScene(scene);
		// show the stage
		primaryStage.show();
	}

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

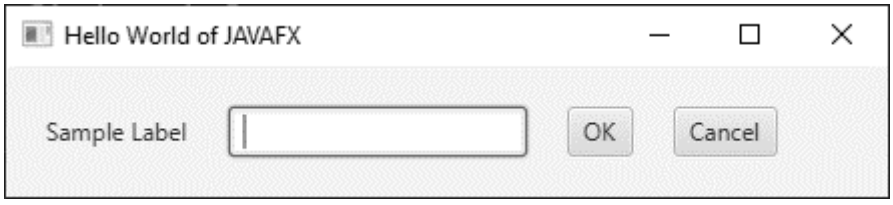


Figure 2: Expected output

Organizing components inside Scene:

Panes are just nodes that can have other nodes (buttons, checkboxes, radio buttons and even other panes). JavaFX offers various types of Panes, each arranging the elements in a systematic way.

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.

Figure 3: Various types of Panes

The panes also expose various types of helpful methods to set the spacing around the UI components.

```
pane.setSpacing(double spacing);
```

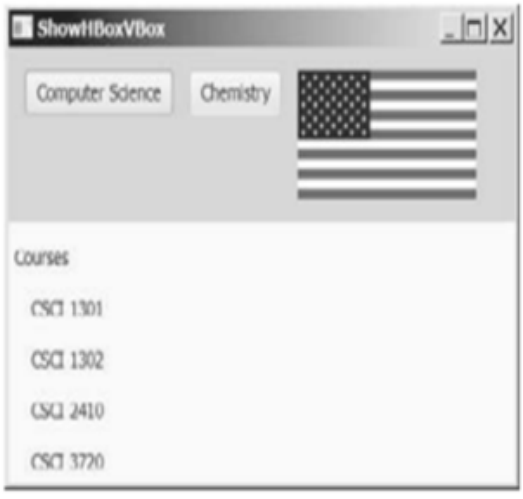
```
pane.setHgap(double spacing);
```

```
pane.setVgap(double spacing);
```

```
pane.setPadding(new Insets(double top, double left, double bottom, double right));
```

A key thing to note is that if a Pane object has `setSpacing()` method, it won't have the `setHgap()` and `setVgap()` methods.

Interactive Session



Write a program that displays two of the popular subjects' name **Computer Science and Chemistry** on the top in **Horizontal order** and few course lists in **Vertical order** with header **Courses.**
Hint: To accomplish this task you may use BorderPane, HBox, VBox.