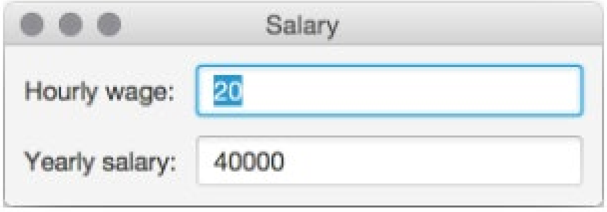
Practical Lab Week 8

Objective: the objective of this lab is to practice JavaFX basics and scene builder usage.

1. Complete the following program which uses a grid pane to present the salary of a staff in hourly wage and yearly salary.



Hourly wage: 20

Yearly salary: 40000

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

public class SalaryLabelGuiFx extends Application {
   @Override
   public void start(Stage applicationStage) {
      int hourlyWage;
      int yearlySalary;
      Scene scene = null;         // Scene contains all content
      GridPane gridPane = null;   // Positions components within scene
      Label wageLabel = null;     // Label for hourly salary
      Label salaryLabel = null;   // Label for yearly salary
      TextField salField = null;  // Displays yearly salary
      TextField wageField = null; // Displays hourly wage
      Insets gridPadding = null;

      gridPane = new GridPane();
      scene = new Scene(gridPane);

      // Calculate yearly salary
      hourlyWage = 20;
      yearlySalary = hourlyWage * 40 * 50;

      // Set hourly and yearly salary
      wageLabel = new Label("Hourly wage:");
      salaryLabel = new Label("Yearly salary:");

      // Create wage and salary text fields
      wageField = new TextField();
      wageField.setPrefColumnCount(15);
      wageField.setEditable(false);
      wageField.setText(Integer.toString(hourlyWage));

      salField = new TextField();
      salField.setPrefColumnCount(15);
      salField.setEditable(false);
      salField.setText(Integer.toString(yearlySalary));
```

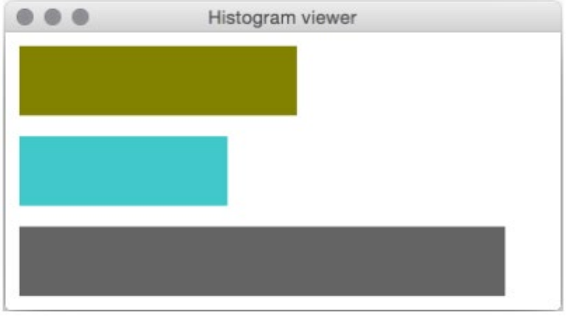
```
        gridPane.add(wageLabel, 0, 0);
        gridPane.add(wageField, 1, 0);
        gridPane.add(salaryLabel, 0, 1);
        gridPane.add(salField, 1, 1);

        gridPadding = new Insets(10, 10, 10, 10);
        gridPane.setPadding(gridPadding);
        gridPane.setHgap(10);
        gridPane.setVgap(10);

        applicationStage.setScene(scene);     // Set window's scene
        applicationStage.setTitle("Salary"); // Set window's title
        applicationStage.show();             // Display window
     }

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

2. Complete the program below to draw a simple histogram with three bins as shown in the figure



```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

public class HistogramViewerFx extends Application {
   @Override
   public void start(Stage applicationStage) {
      Pane pane = new Pane();                      // Create an empty pane
      Scene scene = new Scene(pane);               // Create a scene
containing the pane
      Canvas canvas = new Canvas(400, 200);        // Create a canvas in
which to draw

      // Get the canvas' graphics context to draw
      GraphicsContext graphicsContext = canvas.getGraphicsContext2D();

      // Draw 1st bin as an olive colored rectangle at (10,10)
```

```
        // with width = 200 and height = 50
        Color binColor1 = Color.rgb(128, 128, 0);
        graphicsContext.setFill(binColor1);
        graphicsContext.fillRect(10, 10, 200, 50);

        // Draw 2nd bin as a teal blue rectangle at (10,75)
        // with width = 150 and height = 50
        Color binColor2 = Color.rgb(0, 200, 200);
        graphicsContext.setFill(binColor2);
        graphicsContext.fillRect(10, 75, 150, 50);

        // Draw 3rd bin as a gray rectangle at (10,140)
        // with width = 350 and height = 50
        Color binColor3 = Color.rgb(100, 100, 100);
        graphicsContext.setFill(binColor3);
        graphicsContext.fillRect(10, 140, 350, 50);

        pane.getChildren().add(canvas);                 // Add canvas to pane
        applicationStage.setTitle("Histogram viewer"); // Set window's title
        applicationStage.setScene(scene);               // Set window's scene
        applicationStage.show();                        // Display window
    }

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

3. Create the Canvas in Exercise 2 in Scene Builder. Then load the scene in Java.

Solution contains two parts: java class file (HistogramSceneBuilder.java) and fxml file (histogram.fxml) .
Below is HistogramSceneBuilder.java.

```
import java.io.IOException;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

public class HistogramSceneBuilder extends Application {
      @Override
      public void start(Stage primaryStage) throws IOException {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("histogram.fxml"));
            Pane root = loader.load();
            // Create a scene and place it in the stage
            Scene scene = new Scene(root);
            primaryStage.setTitle("Histogram viewer");
            primaryStage.setScene(scene);
            primaryStage.show();
      }
      public static void main(String[] args) {
            Application.launch(args);
      }
}
```

Below is histogram.fxml.

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.canvas.Canvas?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.shape.Rectangle?>


<Pane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="200.0" prefWidth="400.0"
xmlns="http://javafx.com/javafx/18" xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <Canvas height="200.0" width="400.0" />
      <Rectangle arcHeight="5.0" arcWidth="5.0" fill="OLIVE" height="50.0"
stroke="BLACK" strokeType="INSIDE" width="200.0" x="10.0" y="10.0" />
      <Rectangle arcHeight="5.0" arcWidth="5.0" fill="#00c8c8"
height="50.0" stroke="BLACK" strokeType="INSIDE" width="150.0" x="10.0"
y="75.0" />
      <Rectangle arcHeight="5.0" arcWidth="5.0" fill="#646464"
height="50.0" stroke="#646464" strokeType="INSIDE" width="350.0" x="10.0"
y="140.0" />
   </children>
</Pane>
```

4. You are given an ArrayList of Person objects, where each Person has two attributes “firstName” and “lastName”. Create a scene in JavaFx which displays the array with a table.

```
public class Person {

    private String firstName = null;
    private String lastName = null;

    public Person() {
    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
}


import java.io.IOException;
import java.util.ArrayList;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.TableView;
import javafx.scene.control.TableColumn;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class TableViewer extends Application {
      @Override // Override the start in the Application
      public void start(Stage primaryStage) throws IOException {
            // Create a pane and set its properties
            ArrayList<Person> people = new ArrayList<>();
            people.add(new Person("John", "Doe"));
            people.add(new Person("Jane", "Deer"));
            people.add(new Person("Amy", "Lee"));

            TableView<Person> table = new TableView<>();
            TableColumn<Person, String> firstNameColumn = new
TableColumn<>("First Name");
            firstNameColumn.setCellValueFactory(new
PropertyValueFactory<>("firstName"));
            firstNameColumn.prefWidthProperty().bind(table.widthProperty().
      multiply(0.5));

            TableColumn<Person, String> lastNameColumn = new
TableColumn<>("Last Name");
            lastNameColumn.setCellValueFactory(new
PropertyValueFactory<>("lastName"));

      lastNameColumn.prefWidthProperty().bind(table.widthProperty().multipl
y(0.5));
            table.getColumns().add(firstNameColumn);
            table.getColumns().add(lastNameColumn);

            for (Person person: people) {
                  table.getItems().add(person);
            }

            // Create a scene and place it in the stage
            VBox vbox = new VBox(table);
            Scene scene = new Scene(vbox);
            primaryStage.setTitle("My Table");
            primaryStage.setScene(scene);
            primaryStage.show();
      }
      public static void main(String[] args) {
            Application.launch(args);
      }


}
```