CS 1073

FR04B

Assignment 9

Ebrahim Arefi

3621326

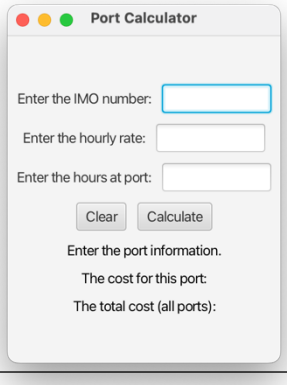
1) the source code for Question I (the JavaFX GUI):

PortCalculator

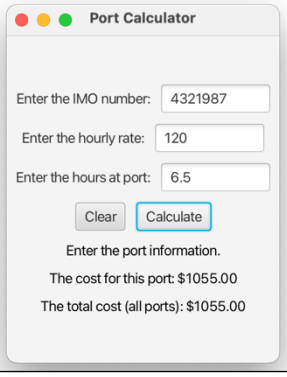
```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.text.Text;
import javafx.scene.layout.FlowPane;
import javafx.geometry.Pos;
import javafx.event.ActionEvent;
import java.text.NumberFormat;
import java.text.DecimalFormat;

/**
 * Port Fee Calculator Application
 *
 * Calculates the cost for ships to stop at ports.
 * Also allows reset with the Clear button.
 *
 * @author Ebrahim Arefi, 3621326
 */
public class PortCalculator extends Application {

    private TextField imoNumber;
    private TextField hourlyRate;
    private TextField hoursAtPort;
    private Text portMsg;
    private Text costMsg;
    private Text totalMsg;

    private double totalCost = 0.0;

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Port Calculator");

        imoNumber = new TextField();
        hourlyRate = new TextField();
        hoursAtPort = new TextField();

        imoNumber.setPrefWidth(100);
        hourlyRate.setPrefWidth(100);
        hoursAtPort.setPrefWidth(100);

        Label imoLabel = new Label("Enter the IMO number:");
        Label rateLabel = new Label("Enter the hourly rate:");
        Label hoursLabel = new Label("Enter the hours at port:");

        Button clearButton = new Button("Clear");
        clearButton.setOnAction(this::processClear);
```

```
        Button calcButton = new Button("Calculate");
        calcButton.setOnAction(this::processCalculate);

        portMsg = new Text("Enter the port information.");
        costMsg = new Text("The cost for this port:");
        totalMsg = new Text("The total cost (all ports):");

        FlowPane pane = new FlowPane(10, 10,
                imoLabel, imoNumber,
                rateLabel, hourlyRate,
                hoursLabel, hoursAtPort,
                clearButton, calcButton,
                portMsg, costMsg, totalMsg);

        pane.setAlignment(Pos.CENTER);
        pane.setVgap(10);
        pane.setHgap(10);

        Scene scene = new Scene(pane, 250, 300);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public void processCalculate(ActionEvent event) {
        double imo = Double.parseDouble(imoNumber.getText());
        double rate = Double.parseDouble(hourlyRate.getText());
        double hours = Double.parseDouble(hoursAtPort.getText());

        double cost = rate * hours;

        int lastDigit = (int) (imo % 10);
        if (lastDigit == 7) {
            cost = cost + 275.0;
        }

        totalCost = totalCost + cost;

        DecimalFormat df = new DecimalFormat("0.00");
        costMsg.setText("The cost for this port: $" + df.format(cost));
        totalMsg.setText("The total cost (all ports): $" + df.format(totalCost));
    }

    public void processClear(ActionEvent event) {
        imoNumber.clear();
        hourlyRate.clear();
        hoursAtPort.clear();
        totalCost = 0.0;

        portMsg.setText("Enter the port information.");
        costMsg.setText("The cost for this port:");
        totalMsg.setText("The total cost (all ports):");
    }
}
```

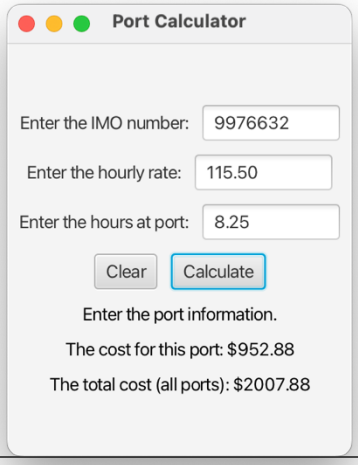
2) The sample output for Question I:



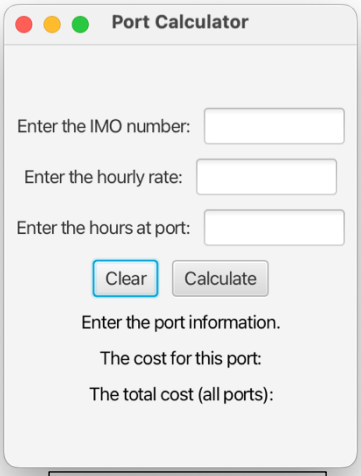
Before Computation



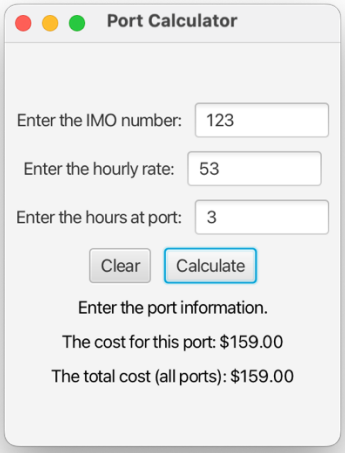
Cruise ship After Computation



Non Cruise ship



Clear



AfterClear

3) Source code for Question II part a (the 3 classes):

Excursion.java:

```
/**
 * Excursion is an abstract class.
 * It stores the customer's name, the number of days, the number of meals,
 * and the number of interpretive walks for the excursion.
 *
 * @author Ebrahim Arefi, 3621326
 */
public abstract class Excursion {

    /**
     * The customer's name.
     */
    protected String name;

    /**
     * The number of days for the excursion.
     */
    protected int daysNum;

    /**
     * The number of meals requested.
     */
    protected int mealsNum;

    /**
     * The number of interpretive walks.
     */
    protected int walksNum;

    /**
     * Constructs an Excursion with the given values.
     *
     * @param nameIn     the customer's name
     * @param daysNumIn  the number of days
     * @param mealsNumIn the number of meals
     * @param walksNumIn the number of walks
     */
    public Excursion(String nameIn, int daysNumIn, int mealsNumIn, int walksNumIn) {
        name = nameIn;
        daysNum = daysNumIn;
        mealsNum = mealsNumIn;
        walksNum = walksNumIn;
    }

    /**
```

```
     * Gets the customer's name.
     *
     * @return the customer's name
     */
    public String getName() {
        return name;
    }

    /**
     * Gets the number of days.
     *
     * @return the number of days
     */
    public int getDaysNum() {
        return daysNum;
    }

    /**
     * Gets the number of meals.
     *
     * @return the number of meals
     */
    public int getMealsNum() {
        return mealsNum;
    }

    /**
     * Gets the number of interpretive walks.
     *
     * @return the number of walks
     */
    public int getWalksNum() {
        return walksNum;
    }

    /**
     * Calculates the total cost of the excursion.
     *
     * @return the total cost
     */
    public abstract double calculateCost();

    /**
     * Retrieves the complimentary perk.
     *
     * @return the perk description
     */
    public abstract String getPerk();
}
```

BasicExcursion.java:

```
/**
 * BasicExcursion class represents a basic excursion.
 * Includes meals, interpretive walks, and provides a complimentary perk.
 *
 * @author Ebrahim Arefi, 3621326
 */
public class BasicExcursion extends Excursion {

    /**
     * The base rate per day for basic customers.
     */
    private static final double BASE_RATE = 125.00;

    /**
     * The cost of each meal.
     */
    private static final double MEAL_COST = 16.75;

    /**
     * The cost of each interpretive walk (normal rate).
     */
    private static final double WALK_COST = 26.50;

    /**
     * The discounted cost of each interpretive walk (if 3 or more walks).
     */
    private static final double WALK_DISCOUNT = 23.75;

    /**
     * Constructs a BasicExcursion with the given information.
     *
     * @param nameIn     the customer's name.
     * @param daysNumIn  the number of days.
     * @param mealsNumIn the number of meals.
     * @param walksNumIn the number of interpretive walks.
     */
    public BasicExcursion(String nameIn, int daysNumIn, int mealsNumIn, int
walksNumIn) {
        super(nameIn, daysNumIn, mealsNumIn, walksNumIn);
    }

    /**
     * Calculates the total cost for the basic excursion.
     *
     * Basic customers pay the standard rates listed above.
     * If they book 3 or more walks, a discount is applied.
     *
     * @return the total cost of the excursion.
     */
```

```
    public double calculateCost() {
        double total = BASE_RATE * getDaysNum();
        total += MEAL_COST * getMealsNum();

        if (getWalksNum() >= 3) {
            total += WALK_DISCOUNT * getWalksNum();
        } else {
            total += WALK_COST * getWalksNum();
        }

        return total;
    }

    /**
     * Retrieves the complimentary perk for basic customers.
     *
     * @return the complimentary perk.
     */
    public String getPerk() {
        return "Water Bottle";
    }
}
```

PremiumExcursion.java:

```
import java.util.Random;

/**
 * PremiumExcursion class represents a premium excursion.
 * Includes meals, interpretive walks, and assigns one random perk.
 *
 * @author Ebrahim Arefi, 3621326
 */
public class PremiumExcursion extends Excursion {

    /**
     * The base rate per day for premium customers.
     */
    private static final double BASE_RATE = 315.00;

    /**
     * The cost of each meal.
     */
    private static final double MEAL_COST = 16.75;

    /**
     * The cost of each interpretive walk.
     */
    private static final double WALK_COST = 20.95;

    /**
     * Constructs a PremiumExcursion with the given information.
     *
     * @param nameIn     the customer's name.
     * @param daysNumIn  the number of days.
     * @param mealsNumIn the number of meals.
     * @param walksNumIn the number of interpretive walks.
     */
    public PremiumExcursion(String nameIn, int daysNumIn, int mealsNumIn, int
walksNumIn) {
        super(nameIn, daysNumIn, mealsNumIn, walksNumIn);
    }

    /**
     * Calculates the total cost for the premium excursion.
     * Premium customers get one free meal per day and one free walk.
     *
     * @return the total cost of the excursion.
     */
```

```
    public double calculateCost() {
        double total = BASE_RATE * getDaysNum();

        int mealsExpenses = getMealsNum() - getDaysNum();
        if (mealsExpenses < 0) {
            mealsExpenses = 0;
        }
        total += mealsExpenses * MEAL_COST;



        return total;
    }

    /**
     * Retrieves a random complimentary perk for premium customers.
     *
     * @return the complimentary perk.
     */
    public String getPerk() {
        Random rand = new Random();
        int num = rand.nextInt(3) + 1;
        String perk;

        if (num == 1) {
            perk = "Walking Poles";
        } else if (num == 2) {
            perk = "Backpack";
        } else {
            perk = "Binoculars";
        }

        return perk;
    }
}
```

4) Source code for Question II part c (the JavaFX GUI front-end):

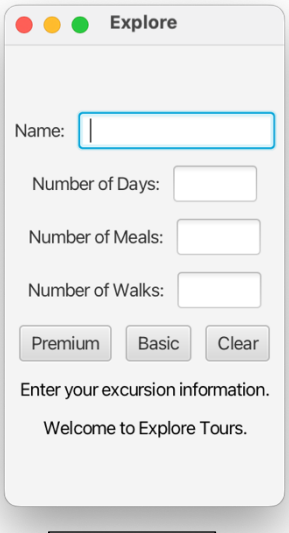
ExcursionApp.java:

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.TextField;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.scene.text.Text;
import javafx.scene.layout.FlowPane;
import javafx.geometry.Pos;
import javafx.event.ActionEvent;
import java.text.NumberFormat;

/**
 * ExcursionApp class allows customers to enter their information
 * and choose between a Basic or Premium excursion.
 *
 * @author Ebrahim Arefi, 3621326
 */
public class ExcursionApp extends Application {

    private TextField nameField;
    private TextField daysField;
    private TextField mealsField;
    private TextField walksField;
    private Text costMsg;
    private Text perkMsg;

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Explore");

        nameField = new TextField();
        daysField = new TextField();
        mealsField = new TextField();
        walksField = new TextField();

        nameField.setPrefWidth(140);
        daysField.setPrefWidth(60);
        mealsField.setPrefWidth(60);
```

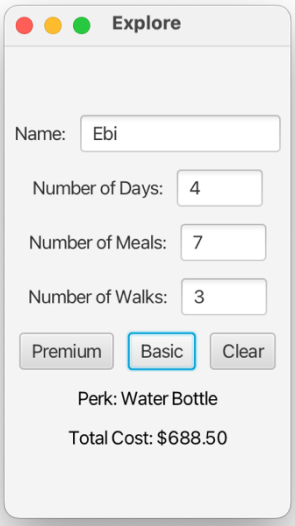
```
    walksField.setPrefWidth(60);

    Label nameLabel = new Label("Name:");
    Label daysLabel = new Label("Number of Days:");
    Label mealsLabel = new Label("Number of Meals:");
    Label walksLabel = new Label("Number of Walks:");

    Button premiumButton = new Button("Premium");
    premiumButton.setOnAction(this::processPremium);

    Button basicButton = new Button("Basic");
    basicButton.setOnAction(this::processBasic);

    Button clearButton = new Button("Clear");
    clearButton.setOnAction(this::processClear);

    costMsg = new Text("Welcome to Explore Tours.");
    perkMsg = new Text("Enter your excursion information.");

    FlowPane pane = new FlowPane(10, 10,
            nameLabel, nameField,
            daysLabel, daysField,
            mealsLabel, mealsField,
            walksLabel, walksField,
            premiumButton, basicButton, clearButton,
            perkMsg, costMsg);

    pane.setAlignment(Pos.CENTER);
    pane.setHgap(10);
    pane.setVgap(12);

    Scene scene = new Scene(pane, 200, 330);
    primaryStage.setScene(scene);
    primaryStage.show();
}

public void processBasic(ActionEvent event) {
    String name = nameField.getText();
    int days = Integer.parseInt(daysField.getText());
    int meals = Integer.parseInt(mealsField.getText());
    int walks = Integer.parseInt(walksField.getText());

    BasicExcursion basic = new BasicExcursion(name, days, meals, walks);
    double total = basic.calculateCost();

    NumberFormat currency = NumberFormat.getCurrencyInstance();
    costMsg.setText("Total Cost: " + currency.format(total));
    perkMsg.setText("Perk: " + basic.getPerk());
```

```
    }

    public void processPremium(ActionEvent event) {
        String name = nameField.getText();
        int days = Integer.parseInt(daysField.getText());
        int meals = Integer.parseInt(mealsField.getText());
        int walks = Integer.parseInt(walksField.getText());

        PremiumExcursion premium = new PremiumExcursion(name, days, meals,
walks);
        double total = premium.calculateCost();

        NumberFormat currency = NumberFormat.getCurrencyInstance();
        costMsg.setText("Total Cost: " + currency.format(total));
        perkMsg.setText("Perk: " + premium.getPerk());
    }

    public void processClear(ActionEvent event) {
        nameField.clear();
        daysField.clear();
        mealsField.clear();
        walksField.clear();
        costMsg.setText("Welcome to Explore Tours.");
        perkMsg.setText("Enter your excursion information.");
    }
}
```

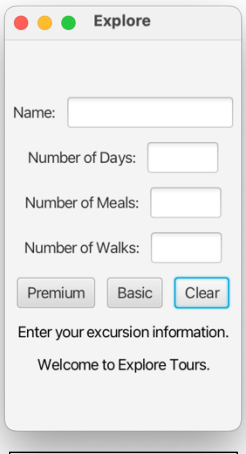
5) The sample output for Question II:



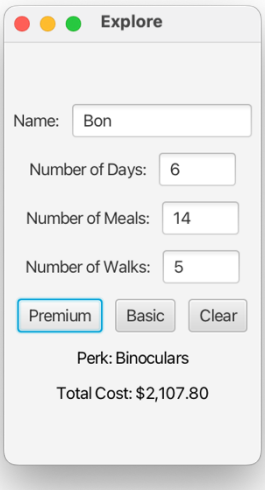
Q2_Initial



Q2_Basic



Q2_Clear



Q2_Premium