CMSC 215

Programming Project 3 – Trip Cost Estimator

Daniel Smolsky

Section 1 – Approach

My approach to this project was to implement the GUI in an organized way similar to how GUI formatting was taught in the textbook. First, I defined all my GUI nodes and placed them on a pane. Next, I set the properties for the UI for things like alignment and node sizes. Finally, I created an action event that would create TripCost objects when the Calculate button is pressed. The method triggered by the event would output the final trip cost calculated within the TripCost object to a textfield node in the GUI.

Section 2 – Assumptions

My GUI assumes the user only enters numerical input into the textfields, anything else will raise type mismatch exceptions. Additionally, there are only a few settings to the desired output. For example, the trip cost calculator assumes currency is always dollars and there are only a couple types of expenses, for example there is no textfield for emergency expenses of travel expenses other than gas costs.

Section 3 – Not Implemented

Section 4 – User Guide

Extract the Project3.java file from the .zip file and run the program. The JavaFX library is not standard with all java JDK versions so ensure the library is installed if it is not. The GUI loads up with initial example values, modify these values and press the “Calculate” button to calculate the cost of your trip.

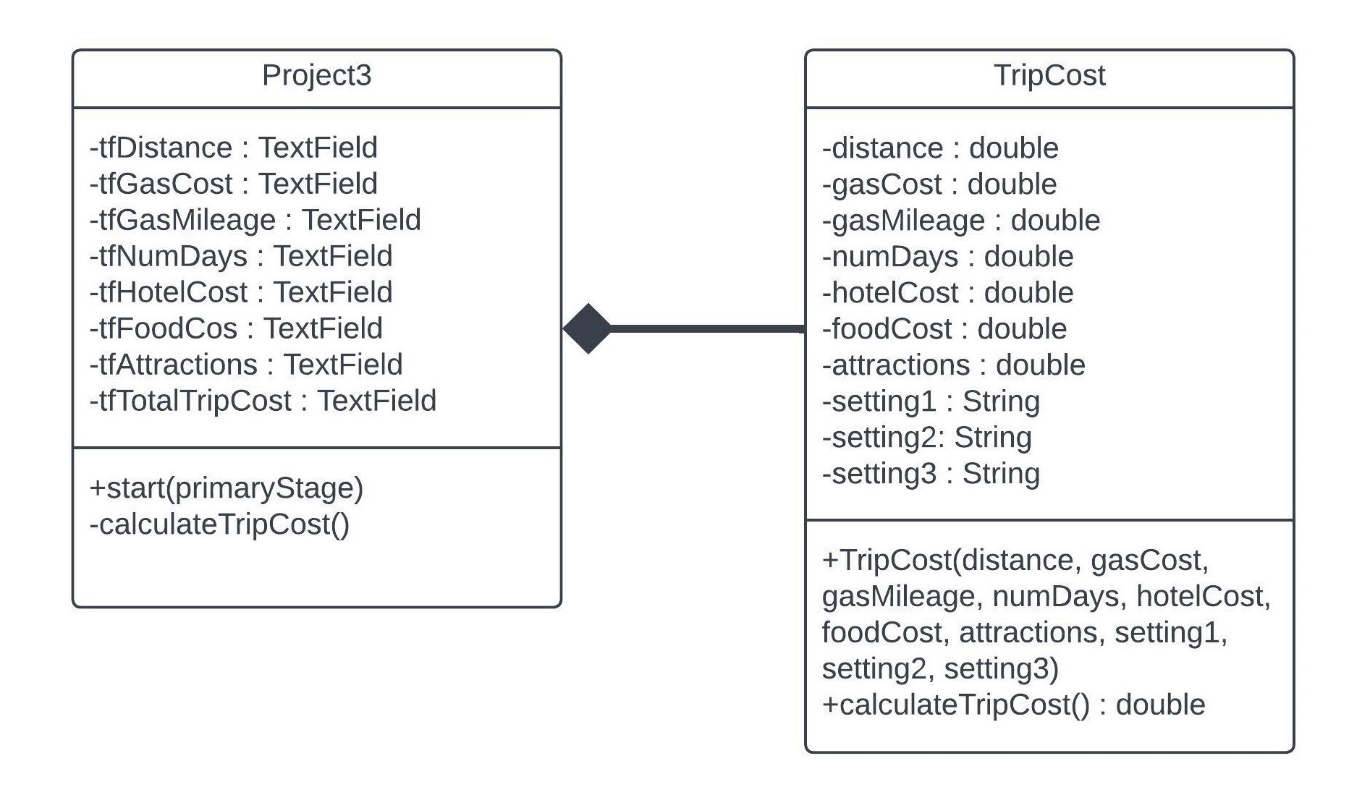
Section 5 – Lessons Learned

Implementing this project, I learned how to create JavaFX GUI applications and how to create inner classes within public classes. As part of learning JavaFX, I learned how to create different types of node objects, different types of panes to hold the node objects, and how to use onAction methods to process user input. I also learned how to use lambda expressions to simplify inner class event processing.

Section 6 – Possible Improvements

The main improvement I wanted to investigate was to create onAction event handlers for any textfield inputs to prevent non numerical inputs. I’m not entirely sure how to implement this, because as far as I understand, such handlers would trigger after the textfield already have a value input, so I don’t know how I would make the handler first check if the input is numerical before placing it into the textfield. Additionally, I tried to follow the design guidelines as closely as possible and I wasn’t sure how much creative freedom I had, so I didn’t implement any animation nodes or other cool graphics we learned about in the textbook.

Section 7 – UML



Section 8 – Source Code

My code can also be found on my [Github](https://github.com/Tarquinen/School-repo/blob/main/CMSC%20215/Week6/Project3.java).

*/\**

*\* Daniel Smolsky*

*\* Programming Project 3: Trip Cost Estimator*

*\* Feb 17, 2024*

*\* This class creates a GUI that allows the user to input various trip costs and*

*\* calculates the total trip cost.*

*\*/*

import javafx.application.Application;

import javafx.geometry.HPos;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

import javafx.scene.control.ComboBox;

public class Project3 extends Application {

*// Create text fields, buttons, and combo boxes*

    private TextField tfDistance = new TextField("1000");

    private TextField tfGasCost = new TextField("3.95");

    private TextField tfGasMileage = new TextField("31");

    private TextField tfNumDays = new TextField("2");

    private TextField tfHotelCost = new TextField("150");

    private TextField tfFoodCost = new TextField("125");

    private TextField tfAttractions = new TextField("78");

    private TextField tfTotalTripCost = new TextField();

    private Button btCalculate = new Button("Calculate");

    private ComboBox<String> cbDistanceMeasurement = new ComboBox<>();

    private ComboBox<String> cbGasCostPerVol = new ComboBox<>();

    private ComboBox<String> cbDistancePerVol = new ComboBox<>();

    public void **start**(Stage *primaryStage*) {

*// Create UI*

        GridPane gridPane = new GridPane();

        gridPane.setHgap(5);

        gridPane.setVgap(5);

*// Place nodes*

        gridPane.add(new Label("Distance:"), 0, 0);

        gridPane.add(tfDistance, 1, 0);

        gridPane.add(cbDistanceMeasurement, 2, 0);

        gridPane.add(new Label("Gasoline Cost:"), 0, 1);

        gridPane.add(tfGasCost, 1, 1);

        gridPane.add(cbGasCostPerVol, 2, 1);

        gridPane.add(new Label("Gas Mileage:"), 0, 2);

        gridPane.add(tfGasMileage, 1, 2);

        gridPane.add(cbDistancePerVol, 2, 2);

        gridPane.add(new Label("Number Of Days:"), 0, 3);

        gridPane.add(tfNumDays, 1, 3);

        gridPane.add(new Label("Hotel Cost:"), 0, 4);

        gridPane.add(tfHotelCost, 1, 4);

        gridPane.add(new Label("Food Cost:"), 0, 5);

        gridPane.add(tfFoodCost, 1, 5);

        gridPane.add(new Label("Attractions:"), 0, 6);

        gridPane.add(tfAttractions, 1, 6);

        gridPane.add(btCalculate, 1, 7);

        gridPane.add(new Label("Total Trip Cost"), 0, 8);

        gridPane.add(tfTotalTripCost, 1, 8);

*// Set combo boxes*

        cbDistanceMeasurement.getItems().addAll("Miles", "Kilometers");

        cbDistanceMeasurement.getSelectionModel().select(0);

        cbGasCostPerVol.getItems().addAll("Dollars/Gallon", "Dollars/Liter");

        cbGasCostPerVol.getSelectionModel().select(0);

        cbDistancePerVol.getItems().addAll("Miles/Gallon", "Kilometers/Liter");

        cbDistancePerVol.getSelectionModel().select(0);

*// Set properties for UI*

        gridPane.setAlignment(Pos.CENTER);

        GridPane.setHalignment(cbDistanceMeasurement, HPos.RIGHT);

        GridPane.setHalignment(cbGasCostPerVol, HPos.RIGHT);

        GridPane.setHalignment(cbDistancePerVol, HPos.RIGHT);

        btCalculate.setAlignment(Pos.CENTER);

        btCalculate.prefWidthProperty().bind(tfDistance.widthProperty());

        cbDistanceMeasurement.setPrefWidth(130);

        cbGasCostPerVol.setPrefWidth(130);

        cbDistancePerVol.setPrefWidth(130);

        tfTotalTripCost.setEditable(false);

        tfDistance.setAlignment(Pos.BOTTOM\_RIGHT);

        tfGasCost.setAlignment(Pos.BOTTOM\_RIGHT);

        tfGasMileage.setAlignment(Pos.BOTTOM\_RIGHT);

        tfNumDays.setAlignment(Pos.BOTTOM\_RIGHT);

        tfHotelCost.setAlignment(Pos.BOTTOM\_RIGHT);

        tfFoodCost.setAlignment(Pos.BOTTOM\_RIGHT);

        tfAttractions.setAlignment(Pos.BOTTOM\_RIGHT);

        tfTotalTripCost.setAlignment(Pos.BOTTOM\_RIGHT);

*// Process events*

        btCalculate.setOnAction(*e* -> calculateTripCost());

*// Create a scene and place it in the stage*

        Scene scene = new Scene(gridPane, 400, 300);

        gridPane.requestFocus();

*primaryStage*.setTitle("Trip Cost Estimator");

*primaryStage*.setScene(scene);

*primaryStage*.show();

    }

    private void **calculateTripCost**() {

*// Get values from text fields*

        double distance = Double.parseDouble(tfDistance.getText());

        double gasCost = Double.parseDouble(tfGasCost.getText());

        double gasMileage = Double.parseDouble(tfGasMileage.getText());

        double numDays = Double.parseDouble(tfNumDays.getText());

        double hotelCost = Double.parseDouble(tfHotelCost.getText());

        double foodCost = Double.parseDouble(tfFoodCost.getText());

        double attractions = Double.parseDouble(tfAttractions.getText());

        String setting1 = cbDistanceMeasurement.getValue();

        String setting2 = cbGasCostPerVol.getValue();

        String setting3 = cbDistancePerVol.getValue();

*// Create a TripCost object*

        TripCost tripCost = new TripCost(distance, gasCost, gasMileage, numDays, hotelCost, foodCost,

        attractions, setting1, setting2, setting3);

*// Display total trip cost in text field*

        tfTotalTripCost.setText(String.format("$%.2f", tripCost.calculateTripCost()));

    }

    private class TripCost {

*// Create variables*

        private double distance;

        private double gasCost;

        private double gasMileage;

        private double numDays;

        private double hotelCost;

        private double foodCost;

        private double attractions;

        private String setting1;

        private String setting2;

        private String setting3;

*// Constructor for TripCost*

        public TripCost (double *distance*, double *gasCost*, double *gasMileage*, double *numDays*, double *hotelCost*, double *foodCost*,

            double *attractions*, String *setting1*, String *setting2*, String *setting3*) {

            this.distance = *distance*;

            this.gasCost = *gasCost*;

            this.gasMileage = *gasMileage*;

            this.numDays = *numDays*;

            this.hotelCost = *hotelCost*;

            this.foodCost = *foodCost*;

            this.attractions = *attractions*;

            this.setting1 = *setting1*;

            this.setting2 = *setting2*;

            this.setting3 = *setting3*;

        }

*// Calculate total trip cost*

        public double **calculateTripCost**() {

            double totalTripCost = 0;

            if (setting1.equals("Kilometers")) {

                distance = distance \* 0.621371;

            }

            if (setting2.equals("Dollars/Liter")) {

                gasCost = gasCost \* 3.78541;

            }

            if (setting3.equals("Kilometers/Liter")) {

                gasMileage = gasMileage \* 0.425144;

            }

            totalTripCost = (distance / gasMileage) \* gasCost + (hotelCost + foodCost) \* numDays + attractions;

            return totalTripCost;

        }

    }

    public static void **main**(String[] *args*) {

        launch(*args*);

    }

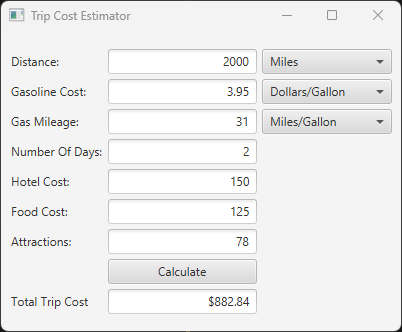
}

Section 9 – Test Plans

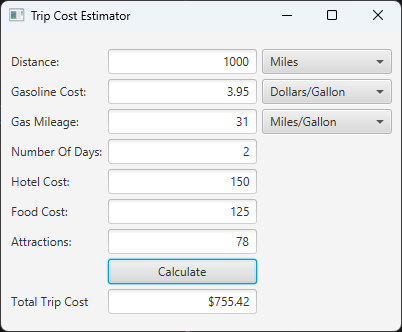
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Purpose | Pos/Neg Test | Input Values | Expected Result | Pass/Fail |
| 1 | Calculate button updates total trip cost textfield | Positive | 2000, 3.95, 31, 2, 150, 125, 78, miles, dollars/gal, miles/gallon | $882.84 | Pass |
| 2 | default values produce expected result from test plan required GUI template | Positive | 1000, 3.95, 31, 2, 150, 125, 78, miles, dollars/gal, miles/gallon | $755.42 | Pass |
| 3 | Combo boxes apply unit change multiplier to the output | Positive | 1000, 3.95, 31, 2, 150, 125, 78, kilometers, dollars/liter, kilometers/liter |  | Pass |
| 4 | Program fails to calculate trip cost on non-numerical textfield input | Negative | 1000, three, twenty, 2, 150, 125, 100, miles, dollars/gal, miles/gal | Total Trip Cost is not updated | Pass |
| 5 | Total trip cost textfield is not editable | Positive | Type 1000 into Total Trip Cost textfield | Can’t type into the textfield | Pass |
| 6 | Nodes are not bound to pane and resizing window leads to non-functional GUI | Negative | Modify the size of the scene | Some textfields or labels become unreadable | Pass |

Section 10 – Screen Shots

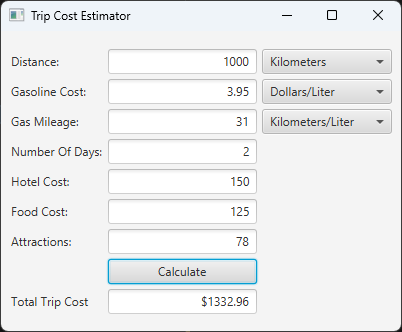
Test 1 –



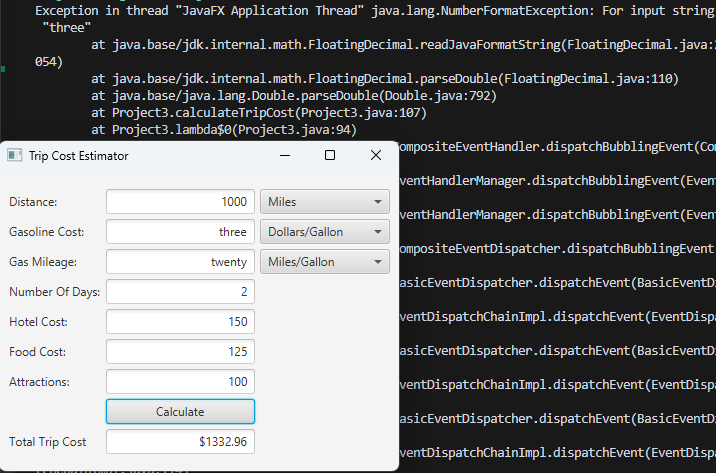
Test 2 –



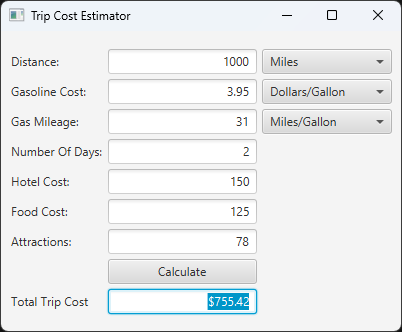
Test 3 –



Test 4 –



Test 5 –



Test 6 –

