

**BINUS UNIVERSITY**

**Assignment Cover Letter**

**(Individual Work)**

**BINUS INTERNATIONAL**

Student Information: Surname Given Names Student ID Number

1. Jevon Danaristo 2440043591

Course Code : COMP6699

Class : L2AC

Major : Computer Science

Title of Assignment : Driver Attendance Log 2

Type of Assignment : Final Project

Submission Pattern

Due Date : Submission Date:

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating** BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality** By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

(Name of Student)

Jevon Danaristo

**“Driver Attendance Log”**

**Name: Jevon Danaristo**

**ID: 2440043591**

1. **PROJECT SPECIFICATIONS**
   1. **Program description**

Driver Attendance Log 2 (DAL2 for short) is an application dedicated to recording a driver's activity and computing the driver's salary. This program is an overhaul of the first Driver Attendance Log application using Java as its programming language. DAL2 is trying to solve problems in the mining industry where many variables such as operating location, number of routes taken, and the type of vehicle used in operation determine the driver's salary.

* 1. **JavaFx**

This project is made with the help of JavaFX. JavaFX is a java platform that provide tools for building a desktop application. JavaFX was choose as this project framework because it is faster to build a program with JavaFX drag and drop controls.

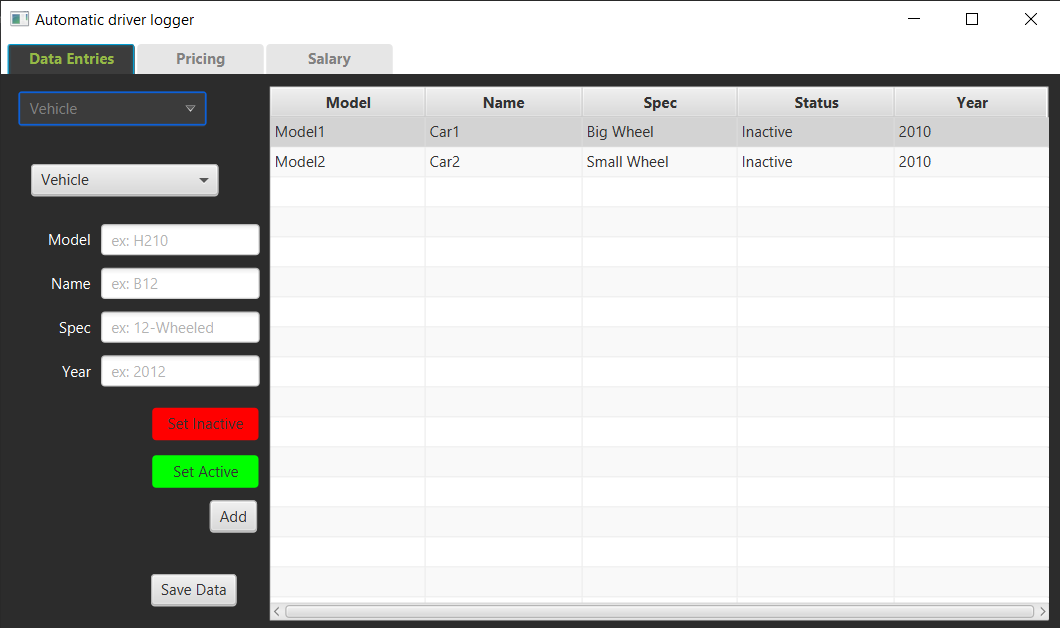
JavaFX create a user interface using a drag and drop method. The UI information will later be saved in FXML file. The FXML later can use controllers to interact with standard java code.

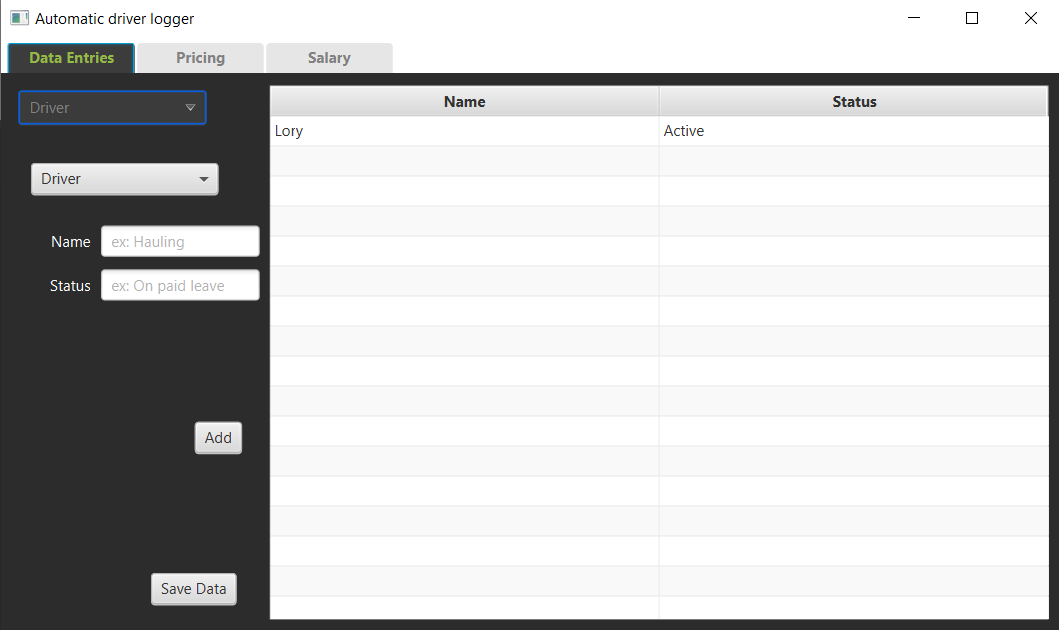
1. **Solution Design**

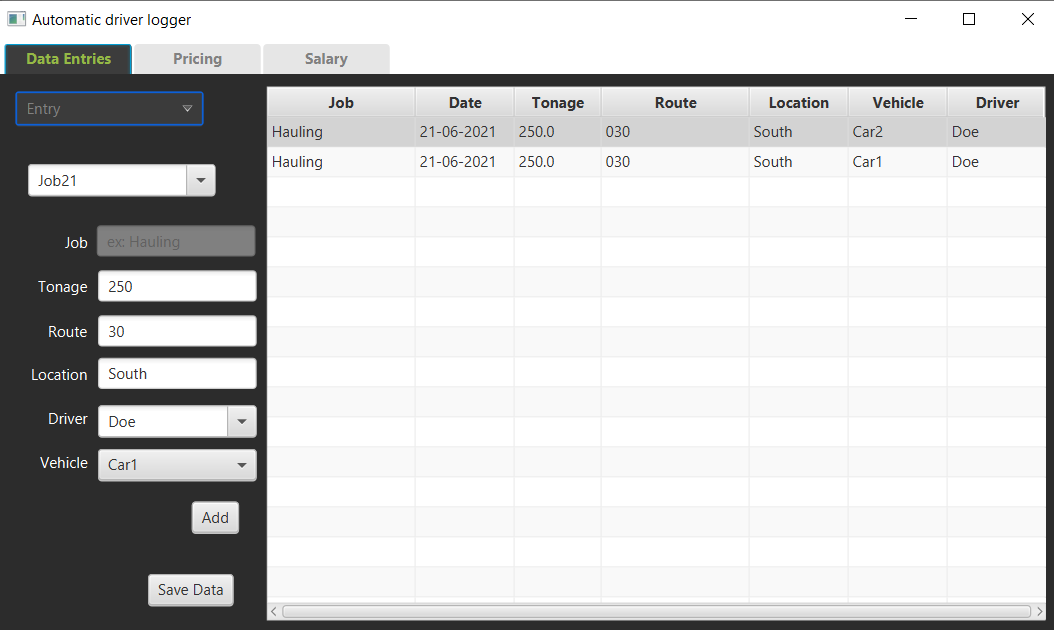
DAL2 is a single window program with several tabs. The tabs organize the application based on its functionality and contain several elements.

1. **Application Tabs**
   1. **Entry Tab**

The entry tab is the first tab that is shown after running the application. Three entry modes made up the entry tab: Log entry is where all the activity data is stored, Driver entry is where all driver’s data is stored, and Vehicle entry is where all the vehicle data is stored. All of the modes have their own save function and a table view to display all of the three mode’s respective data.

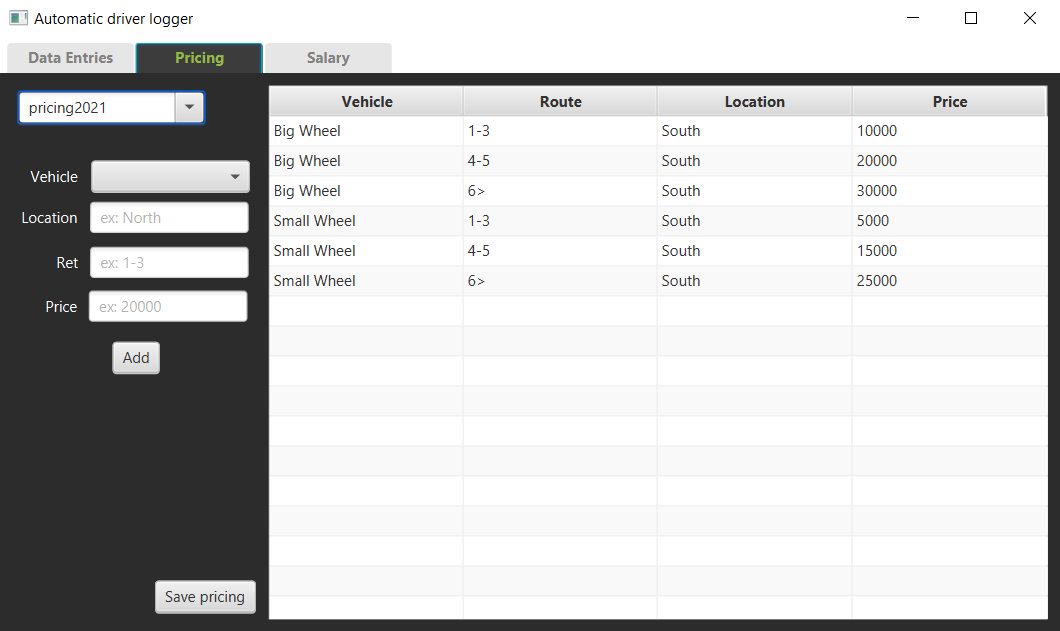
****

****

****

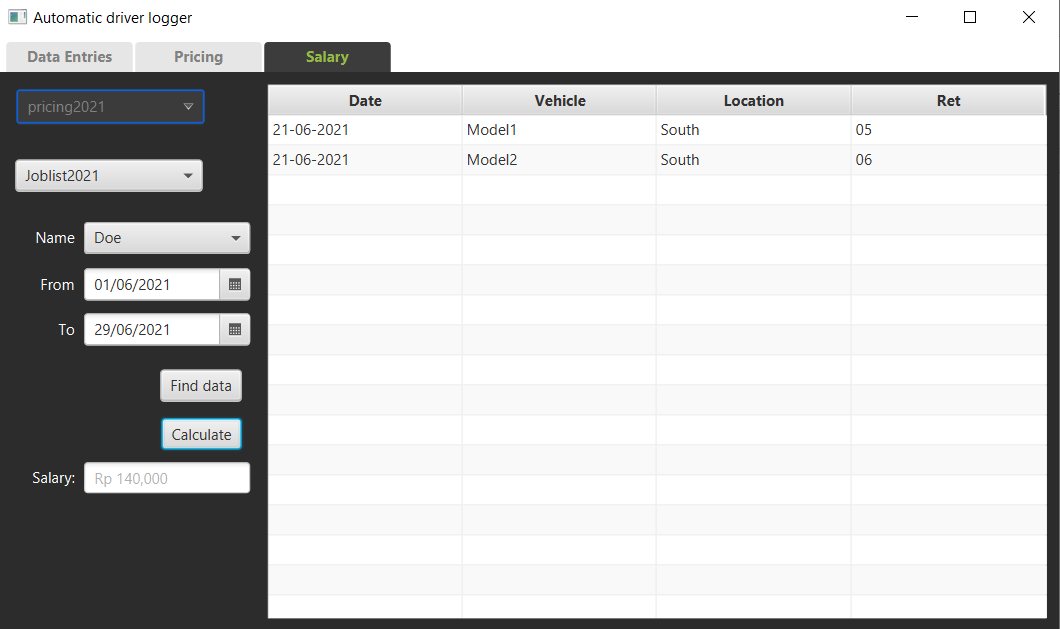
* 1. **Pricing Tab**

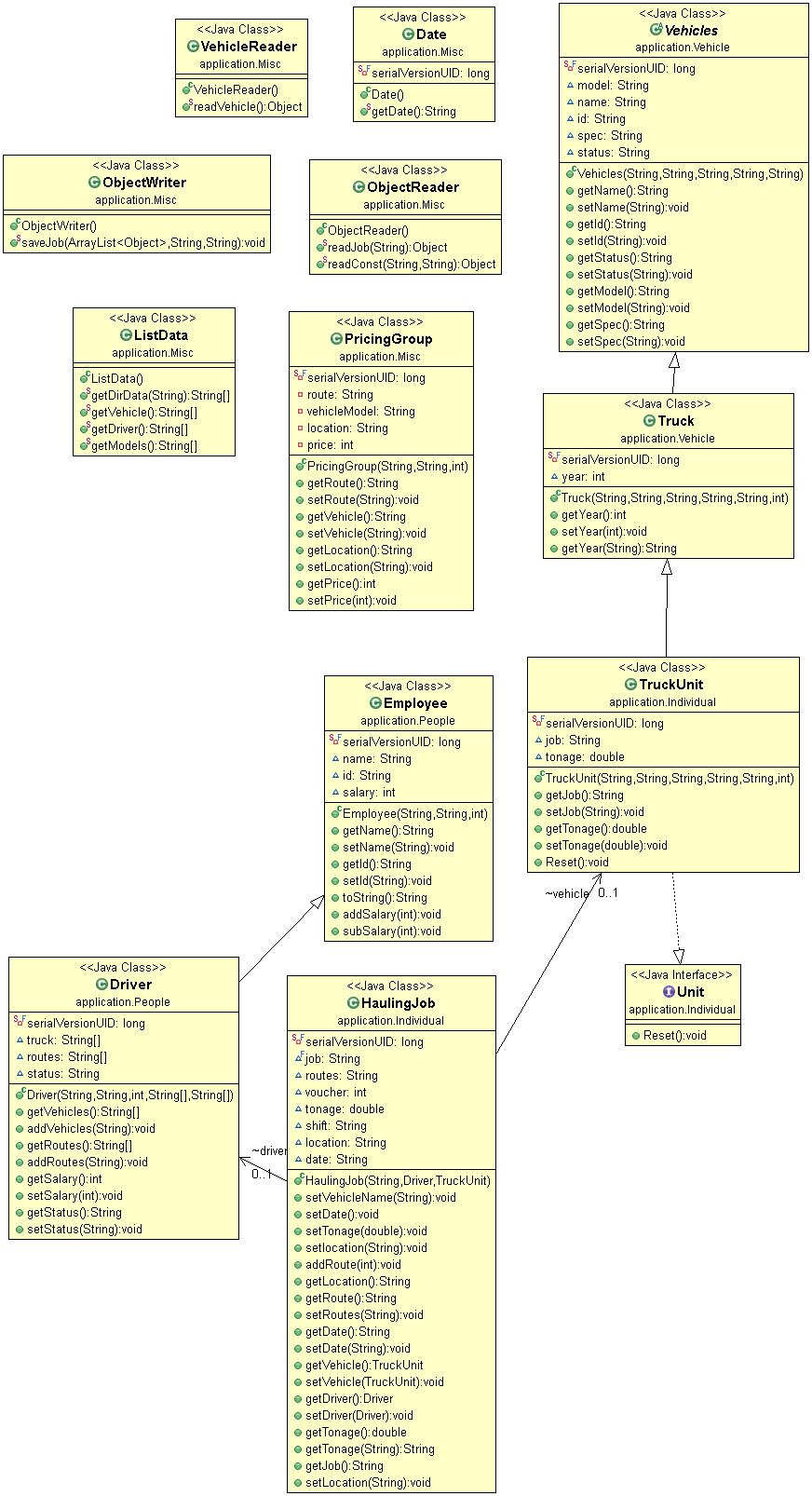
The pricing tab provides entries to determine the salary output based on variables in the data entries such as routes, vehicle type, and the location of operation. All of the entries will later then be used to make a pricing object. The pricing tab also provides a saving function and can only add the existing types of vehicles and drivers.

****

* 1. **Salary Tab**

The salary tab is the last tab in this application. The function of this tab is to view the activity data of each driver in a set amount of time determined by the user. Previous data set such as log data and pricing data will then be used to determine the driver's salary.

****

1. **Class Diagram**
2. **Algorithms**
   1. **Creating New Window**

To create a new window FXML loader is used to load all elements in the FXML

Parent homeRoot = FXMLLoader.*load*(getClass().getResource("Sample.fxml"));

* 1. **User Input**

All of the user input is taken from the JavaFX *text field* and takes the value inside the text field with the *getValue* method that later will be processed accordingly. Before doing so, it is imperative to inject FXML and create a field declaration like the following code

*@FXML* private TextField tonageField;

And later use the *getValue* method to get the value inside the *text field*

tonageField.getValue()

* 1. **Creating Object With the User Input**

In the application, most entry will be stored in an object



* 1. **Displaying Object**

After creating an object, the program will store the object inside an observable array and display all the object attributes inside *table view* columns.



* 1. **Saving Objects**

Objects are saved using *ObjectOutputStream* in *saveJob* function that takes Inpath and Mdir as arguments, Inpath is the filename while Mdir is the directory of the files.



* 1. **Reading Objects**

Objects are loaded using *ObjectInputStream* in *readConst* function that takes dir and filename as arguments, dir is the directory o while filename is the name of the file that the program will search.



* 1. **Getting Current Date**

The log entry requires date as one of its information, to get the date *localdatetime* is used alongside *datetimeformatter* to create the desired date format.



* 1. **Calculating Salary**

Calculating the *employee’s* salary is done with series of loops and conditions. The most inner loop iterates over *routes* data from the entry log and check if the *log’s* *vehicle* type and *variable* match any *pricing’s vehicle* type and *variable*. The second condition will check if there is any hyphen (-) character or greater than (>) character. The second condition will determine either the *route* will be compared in range or greater than manner. If the third condition is fulfilled, the current *pricing’s* *price* will be added to the *employee’s salary*



1. **Reflection**
   1. **Thoughts**

I personally consider this final project application as an improvement to my previous project application by trying to recreate it using OOP and Java programming language. I think Java is quite straightforward and is superior in terms of creating GUI applications compared to python based on my experience in both of my final projects. On the other hand, plotting data is quite difficult and made me remove it entirely from the program.

* 1. **Difficulties**

Most of the difficulties come from the JavaFX methods and data types. Most of the JavaFX data types is not compatible to primitive data type. For example, the *ObservableList* creates plenty of problems and require me to change some of the class attributes into an observable version of the attribute. Another difficulty is solving errors related to JavaFX. Since JavaFX often time does not mention the line or even the main cause of an error.

* 1. **Future work**

This final project gave me a better understanding of OOP, Java programming language, and JavaFX. In future work, this will come as a good personal reference for when I am trying to make another application or web application.