*Petrol Station simulation Report: Group 11*

Aston University Engineering and Applied Sciences CS1410: Java Programming Development

*Perkins, Tristan*

*Minguell, Kelvin*

*Bartosevicius, Matas*

*Ramalho, Milton*

*Aston University*

Table of Contents

[Distinguishing between Library Components and Client Code Data 3](#_Toc481568691)

[Packages 3](#_Toc481568692)

[Separation of Model-View Structure 3](#_Toc481568693)

[Code Layout 3](#_Toc481568694)

[(Level 1) Required Changes for the Library Classes to Support Simulations 3](#_Toc481568695)

[More types of vehicles 3](#_Toc481568696)

[Multiple types of fuel with different prices 3](#_Toc481568697)

[Parking away from the pump during shopping 3](#_Toc481568698)

[Vehicles breaking down during the simulation 4](#_Toc481568699)

[UML Class Hierarchy 5](#_Toc481568700)

[(Level 1 only) Sequence Diagram for a Scenario in the Simulation 5](#_Toc481568701)

[The results of the simulations in tabular form; 6](#_Toc481568702)

[Results and Implications 6](#_Toc481568703)

[Building and Running the Code 6](#_Toc481568704)

# Distinguishing between Library Components and Client Code Data

This description should explain the Rationale behind design decisions.

## Packages

*Start Here*

### Separation of Model-View Structure

*Start Here*

## Code Layout

The code layout for the client code utilizes a general structure of variables, followed by a constructor and the intended methods.

Notes: ***Code Structure***

Explain the number of packages used

Explain the separation of Model-View Understanding.

# (Level 1) Required Changes for the Library Classes to Support Simulations

The initial changes to the classes to support the simulation can entail with the following:

## More types of vehicles

If different types of vehicles need to be created, it’ll require creating a brand-new class that requires stating the name of the vehicle. However, this requires updating the Config class and adding the information for the intended vehicle, by adding the seed, probabilities and other important variables associated.

## Multiple types of fuel with different prices

Multiple Type of fuel would require creating an array for the Pump class with the number of type of fuels it has. Have it implemented and outputting both on the console and the GUI that the vehicle used this fuel for this pump. To give the pump fuel types easy use, would mean to create a new GUI component to create a new window representing to add a new text field to write the fuel type and fuel price data. Having the GUI to manipulate the price and the fuel type in a new window would allow the user to define their fuel and parameters easily.

## Parking away from the pump during shopping

Initially, the idea of dedicated parking

## Vehicles breaking down during the simulation

A vehicle breaking down can

# UML Class Hierarchy

*Image Here*

# (Level 1 only) Sequence Diagram for a Scenario in the Simulation

*Image Here*

# The results of the simulations in tabular form;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Results and Implications

The result does show the current size of the queue and clearly states the information of whether it is a

# Building and Running the Code

To Build and Run the Program this can be done in two different ways: one is to open the package, open the ‘aston.simulator’ package then run the runGUI class as it contains a main() method, required to run code independently. This will compile and present the standard GUI window, which is the Petrol Station Parameter Window. The other way is to select on the petrolStation source file, then selecting the Build Button to create an executable Java file so that it can run independently as long as one defining and appropriate class has the standard main() method.