**FUNCTIONAL REQUIREMENTS DOCUMENT**

**PROGRAM INTRODUCTION**

The Problem:

In thinking about what our program will do, the following scenario came to mind as a problem our program (with its functional requirements) could solve -

A delivery company needs to begin a vehicle fleet in order to complete their deliveries. To establish a fleet contract a minimum of three vehicles need to be ordered from the fleet provider.

The Solution:

We will provide a program for the fleet company to provide vehicle options to the delivery company. In the program we will ensure a minimum of three vehicles are ordered to meet the fleet requirement and store the order details in a file as a type of order invoice to display to the user.

**PROGRAM DETAILS**

User Input and Program Flow:

Our program will allow a user to select a vehicle type to order from the options of Car, SUV, or Truck. It will then prompt the user to select options given for that vehicle type. This process will repeat for three different vehicles chosen and customized by the user. Once the three vehicles and their respective options have been selected by the user, an option to place the order and create the invoice will be given.

If the customer chooses to place the order, the invoice will be created and displayed back to the user. If the customer chooses not to place the order, they’ll be given the option to add more vehicles or simply cancel the order process and exit the program. If they choose to add more vehicles the vehicle types and options menu will take the user through their next vehicle selection and then prompt once again whether they would like to place the order.

Program Execution Details:

*The program begins with the introduction:*

“This program will walk you through the creation of your company’s new vehicle fleet. A minimum of three vehicles will need to be ordered for the creation of your fleet. Let’s begin selecting your first vehicle and its options.”

*The first prompt is given:*

“Enter ‘1’ for Car, ‘2’ for SUV, ‘3’ for Truck : ”

*A switch statement is used to apply the user’s answer and create the correct vehicle object.*

*For this example, we’ll assume the user has entered ‘1’ for Car. The program will then begin the prompts for the user to select the Car options :*

“2 seats or 4 seats? : “

“4”

“what color? : “

“blue”

“Transmission type?

Enter 1 for Automatic or 2 for Manual : “

“1”

*The car object with the selected properties is added to the array of objects. The prompt process is repeated 2 more times. Then the user is prompted with the option to place the order:*

“Would you like to place the fleet order?

Enter 1 for yes or 2 for no : “

“1”

*A printWriter object is created and the three vehicles data is printed to a file in an order invoice format. A File object and Scanner object are created to read out the order invoice file to the user on the console. The program then exits with a farewell message.*

**FUNCTIONAL REQUIREMENTS**

This program will meet the functional requirements for the project assignment as shown in the sections below.

Use Inheritance and Polymorphism:

Our program will have one base class called Car and it will have two child classes called SUV and

Truck. The main base class will contain data common to most cars such as seats, paint color, and transmission type. The two child classes will inherit from these attributes in the base class and extend with their own unique attributes.

Include a collection of references to derived class objects:

In our problem scenario at least three vehicles are required to start a fleet. As the user selects these vehicles and these vehicle objects are created, we will then add references to these vehicle objects to an array to form the collection and use in our file creation.

Make use of Polymorphism as it processes these derived class objects:

Our program will make use of polymorphism in the processing of the derived objects through constructor overloading. Both of the child classes will overload the constructor from the base class. For example, our child class Truck will overload the car constructor by adding a new data member bool towPackage.

Be able to read and write data to a file:

The project will include both a printWriter object for writing object data to a file as well as a file and a scanner object to read the data from the created file. We will use the data from three vehicles to write and read data from a file as a type of order invoice file.

Use exceptions to handle any error that occurs while reading or writing:

In our Method signature for main() we will declare throw statements to handle any exceptions for file reading and writing. We will also use specific try and catch blocks for handling any other possible error scenario.

Diagram

Description automatically generated