CAR DEALER PROJECT

ONUR BAHAR

# Contents

1. [Software Design Notes 3](#_bookmark1)

* 1. [Problem 3](#_bookmark2)
  2. [Approaching Methods& Solution 3](#_bookmark3)

1. [Class Diagram](#_bookmark5) 5
   1. [CarDealar Class](#_bookmark6) 5
   2. [Vehicle Abstract Class](#_bookmark7) 5
   3. [PetrolCar Class](#_bookmark8) 5
   4. [ElecticCar Class](#_bookmark9) 6
   5. [Bus Class](#_bookmark10) 6
   6. [Truck Class](#_bookmark12) 6

[References](#_bookmark26) 7

## Software Design Notes

### Problem

**Java** is a [general-purpose,](http://en.wikipedia.org/wiki/General_purpose_programming_language) [concurrent,](http://en.wikipedia.org/wiki/Concurrent_computing) [class-based](http://en.wikipedia.org/wiki/Class-based), [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) [computer programming](http://en.wikipedia.org/wiki/Computer_programming_language) [language](http://en.wikipedia.org/wiki/Computer_programming_language) that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another. In this java application, it is necessary to create an information system for the second hand vehicle seller. There are 4 types of vehicles that the seller will sell and these vehicles have various different features. There are 5 different aspects that I should pay attention to in this information system that I will prepare. These are:

1. Keeping records of the prices of the vehicles sold  
2. to delete the information of the paid vehicles from the system  
3. Keeping a record of how many cars were sold in total   
4. Keeping a record of the total deposit paid   
5. Printing the information of the vehicles sold

### Approaching Methods& Solution

Java supports object-oriented programming techniques that are based on a hierarchy of classes and well-defined and cooperating objects.

**Classes:** A class is a structure that defines the data and the methods to work on that data. When you write programs in Java, all program data is wrapped in a class, whether it is a class you write or a class you use from the Java API libraries. Classes in the Java API libraries define a set of objects that share a common structure and behavior.

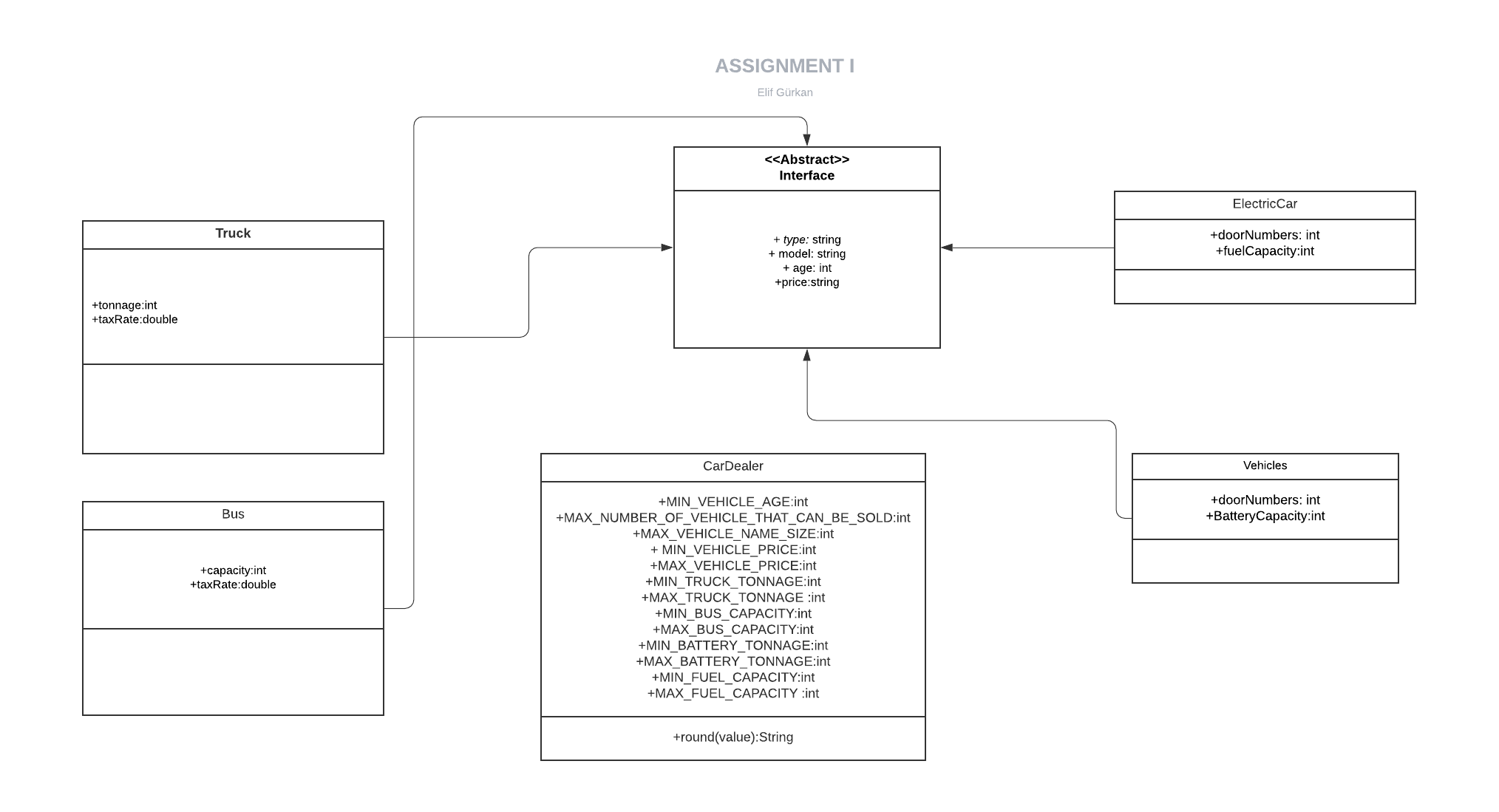
**Objects:** An instance is a synonym for object. A newly created instance has data members and methods as defined by the class for that instance.

**Inheritance and Polymorphism:** One object-oriented concept that helps objects work together is inheritance. Inheritance defines relationships among classes in an object-oriented language. The relationship is one of parent to child where the child or extending class inherits all the attributes (methods and data) of the parent class. In Java, all classes descend from java.lang.Object and inherit its methods. Figure 1 shows the class hierarchy as it descends from java.lang.Objectfor the classes in the user interface example above. The java.lang.Object methods are also shown because they are inherited and implemented by all of its subclasses, which is every class in the Java API libraries. java.lang.Objectdefines the core set of behaviors that all classes have in common.

While designing the program, I thought that the vendor might sell more vehicle types in the future. For this, I first designed the abstract class Vehicles. I have made a design so that all new classes that I will create and that may come in the future will inherit from this class. The most important benefit of this design is that if a motorcycle class needs to be added in the future, it will continue to run without any changes in the code. I used super methods in the constructor methods of the classes that inherited from the Vehicles class and added their unique features separately. I created getter and setter methods in accordance with the Encapsulation concept.

In the Main class, I created the necessary statements and loops for file reading and writing operations. First, I stored the data I read from the file line by line in a simple array. I created appropriate objects by entering the loop and control statements I created according to the first data in the line I read the code. I stored these objects I created with ArrayList. When the read data is "print", I printed the necessary data in the output.txt file. I deleted all the data I had stored after each print command to conform to the design of the program.

## Class Diagram



**Figure 1. Class Diagram**

### Cardealer class

A class called CarDealer will be provided for you in order to give information about the car dealer, such as the capacity of the dealer etc. For this purpose, CarDealer class will contain several fields.

### Vehicle Abstract Class

This class is abstract inheritor class for all vehicle type. This class contains basic fields

### PetrolCar Class

This class for Petrol Car type vehicles. It has 2 special fields called doorNumbers and fuelCapacity.

### Electic Car Class

This class for Electric Car type vehicles. It has 2 special fields called doorNumbers and batteryCapacity

### **2.5** bus class

This class for Petrol Car type vehicles. It has 2 special fields called capacity and taxRate

### **2.6** Truck class

This class for Petrol Car type vehicles. It has 2 special fields called taxRate and tonnage

## References

1-["Programming Language Popularity".](http://www.langpop.com/) 2009. Retrieved 2009-01-16.

2- ["TIOBE Programming Community Index"](http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html). 2009. Retrieved 2009-05-06

3-<https://www.researchgate.net/publication/306918533_Final_Report_Java_Programming_Language_A_Simple_project_to_Draw_Paint_Java_language>

4-<https://stackoverflow.com>

5-https://lucid.app/documents#/dashboard