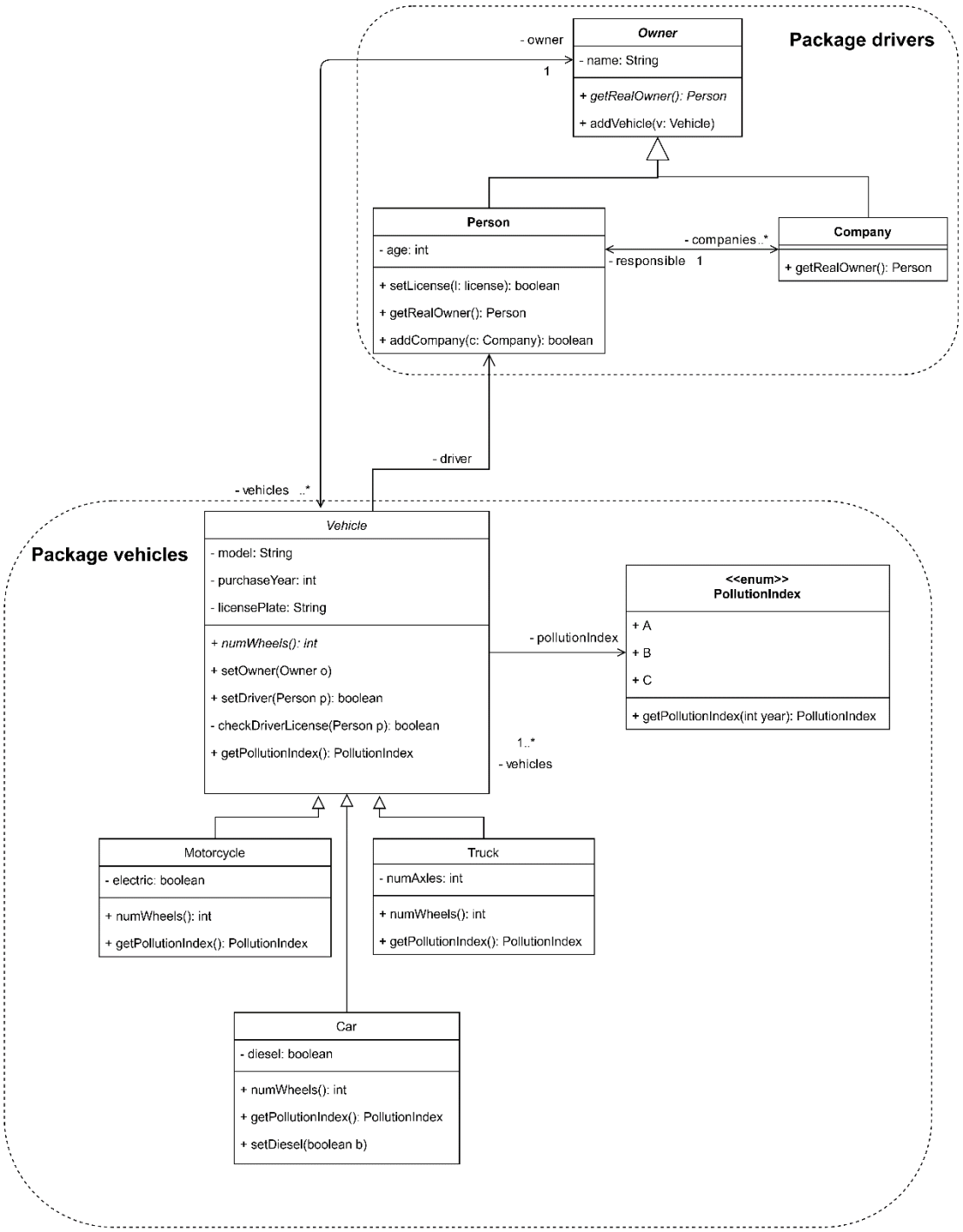
Third Assignment

Introduction to Object Oriented Programming with Java

Kevin de la Coba Malam
Marcos Bernuy López

Section 1



This is the class diagram of section one. In it you can clearly see the package drivers in which we can find the Owner abstract class from which Company and Person inherit. As you can see in the diagram each vehicle will have an owner and a driver. The owner can be either a person or a company and the driver must be a person, so in case that the driver isn't established the driver will be the same as the owner. This can only happen when the owner is a person, in case that the owner is a company we have the function "getRealOwner()" which will return the

person in charge of the company who will then become the driver. The class diagram also illustrates how each owner has N vehicles which are stored in an ArrayList<Vehicle>.

Section 2

For this section we implemented the FineReader class as it was asked to. The process that we followed to be able to read the fines from the txt file was using a FileInputStream an InputStreamReader and a BufferedReader. Once all these objects haven been established we create a loop reading line by line until no more lines are read, using the function readline(). For each line we will get each element separated by the ";" using line.split(), and we will create the fines which then will be stored in an ArrayList and returned.

Section 3

In this section we implement the license package which will have the License class and the PermitKind enumeration. The implementation of the class followed the requirements asked and in order to achieve having a unique identifier for each license we used a static variable which keeps track of the number of licenses and therefore it will associate the next identifier to each new license created. This class also manages the points of the license which can't be less than 0. To avoid getting less than 0 points we use control errors in the removePoints() function.

The enumeration represents the type of permits available, which are A, B, and C1 which will have associated the minimum age required and the vehicle class they belong to.

In addition to the testers provided we created new functions in the TesterLicense class testing the case in which a license is suspended, and more cases for the permits and the ages required both expecting correctness and erroneousness from the tests.

Section 4

In this section we implement the relation between fines and licenses represented in FineProcessor and how a fine can decrease the points of a license. With the process() function we go through every fine and depending on the points of each fine these will be removed from the license of the driver. In order to let the FineProcessor class behave as it should for each situation shown in the section we used if statements. Besides the behaviour we also guaranteed that for each case the proper message is printed. In this section we also implemented new tests.

Section 5

In this section we must first decide the design to follow and then we will implement it. In the design phase we decided to create an Itv class and a Garage class from the itvs package. The

Itv class will have the Garage that it belongs to, some comments and the date in which the Itv was passed.

Regarding the implementation, we had to change the FineProcessor class because in case the car that is being fine hasn't passed the Itv it will get an additional point removed. Another change that was necessary was in the Vehicle and Truck classes, in which we will implement a general function (it affects cars, motorcycles and trucks) to pass the Itv called passItv() and specified functions checkPassedItv() and timeRemaining() which will be the same for the Car and Motorcycle classes but different for the Truck class. This is the reason why we decided to implement the method as abstract in the Vehicle class. These differences is regarding the time which each vehicle has left for the next Itv depending on the age of the vehicle. The timeRemaining() method since it wasn't specified we decided to return the number of days left for the next Itv. In case that you are late for the Itv, meaning you could get fined, it will return the value 0. Every time a fine of a vehicle is detected and the Itv hasn't been passed, the infraction is printed in a file called "ITV_expired.txt".

In this section we also created tests in the TesterITV.java file which tests the correctness and erroneousness of each vehicle and each range of age of the vehicle. The file also tests the proper management of the ITV and the changes made when fines are involucrate.

Class Diagram

This is the class diagram with all the steps that have been previously explained with the part of the ITV that had to be designed by us.