REPORT
OBJECT ORIENTED JAVA PROGRAMMING

***

This project aims at creating a data structure and object model for an app which functions like the Uber mobile application : it provides a service for customers who can book rides from one part of the city to another, selecting the type of the ride and the number of passengers, amongst various options.

MAIN CHARACTERISTICS OF THE CORE PART

We divided the diverse parts of the project into 5 packages : myUberCar, myUberRide and myUberPeople contains the various important objects of the model and the factories to create them. myUberTest contains Junit Tests for all the main classes and myUberOther is for all the classes which could not be part of the other packages.

myUberCar, myUberRide and myUberPeople :

These packages were the first to be created : they contain the classes of the physical objects which interacts in the process : the cars, the drivers, the customers, the rides.

For the cars and the rides, we used superclasses and inheritance : Standard, Berline and Van inherit from Car, whereas UberX, UberVan, UberPool and UberBlack inherit from Ride.

The Factory Pattern

For the object that had to be implemented by the clients (or “chosen” by them) such as the ride, we choose to implement a factory system. This allows us to separate the creation from the client code and makes our solution more “open-close” : one of the requirement was that it should be easy to add new types of rides or car, and with the factories it is the case.

The Visitor Pattern

To calculate the cost of a ride we used a visitor design pattern. The class ConcreteRideCostVisitor can either compute the cost of a single ride (using the visit methods) or return a HashMap of all the rides available as well as their cost through the method pricelist, which we use when we propose a list of rides to the client.

myUberOther :

In this package, we implemented all the classes that had no inheritance and therefore did not need a package of their own. For example, we created a class for the GPS coordinate or an enumeration for the types of Traffic in this package.

The class myUber

More importantly, this package hosts myUber, a class designed to hold all the parameters of a univers in which the program operates. Its attributes are :

- A number of customers
- A number of drivers
- A number of cars

(at this stage of the development, we decided to equalize the nuber of cars and drivers : each driver owns a single car. We made this simplification because we considered that it is not very far from the reality)

- A list of customers
- A list of cars
- A list of drivers
- A list of requested rides (rides requested by clients which haven't been accepted by drivers yet)
- A book of Rides : a list of all the rides that have been conducted in the univers.

This class also contains some useful methods, such as :

- myUber.initiation() : automatically fills the lists of drivers, cars and customers with random objects to match the attribute nbCustomers, nbDrivers and nbCars; It would indeed be very long to populate the univers by creating all those objects one by one. A limitation is that, since we havent't created a function which can generate a random human name yet, all the drivers and customers are named John Doe.

- myUber.requestRide(): This function allows the person running the program to take the place of a client who orders a Ride. First, it selects a random client in the client list and then all the decisions (destination of the ride, number of passengers, and later type of ride) are taken through input from the console.

myUberTest :

This package contains the Junit tests for most of the classes. We haven't testes 100 of the classes, especially when some of them were very similar (UberX and UberBlack for example), or very simple.

ADVANTAGES AND LIMITATIONS OF THE SOLUTION

TEST USE SCENARIO

To test almost all of the functionalities of our project at once, the best is to use the Junit est called myUberTest. It tests the methods of myUber which relies on every class and object that was created.

For the moment, the most advanced thing that the program can do is to simulate the request of a ride by a client. In myUberTest, we can verify that at the end of the run, the list of requestedRides of the univers contains at least one ride.