BLM 2012 OBJECT ORIENTED PROGRAMMING, Gr.1&Gr.2, PROJECT WORK V1.0

Common Rules:

- **Important Note:** This document might be updated to answer your FAQ, so please frequently check website for updates and check the version number of document to see if it is updated.
- This is a group project for teams of maximum 2 students.
- You will write a complete standalone application. A console application is the minimum requirement, such applications will be graded for a maximum of 100 points. If a GUI application is coded, such an application will be graded for a maximum of 120 points, meaning writing GUI code can give up to 20% bonus grade. Term project will have %10 weight in your cumulative grade.
- You must use the Eclipse GUI for this project. The source code must be compressed into a zip file that is named as the student numbers of the team numbers separated by a minus sign, such as 19011007-19011009.zip
- The source code must contain at least two non-trivial jUnit test cases. Creating test cases with jUnit will be the topic of the lecture on May 9th.
- The zip file will be sent to the lab coordinator assistant İbrahim Onur Sığırcı until May 17th.
- The lab coordinator will publish a schedule for the teams to demonstrate their projects in a later time.

Specific Rules for Group 1 (MSA):

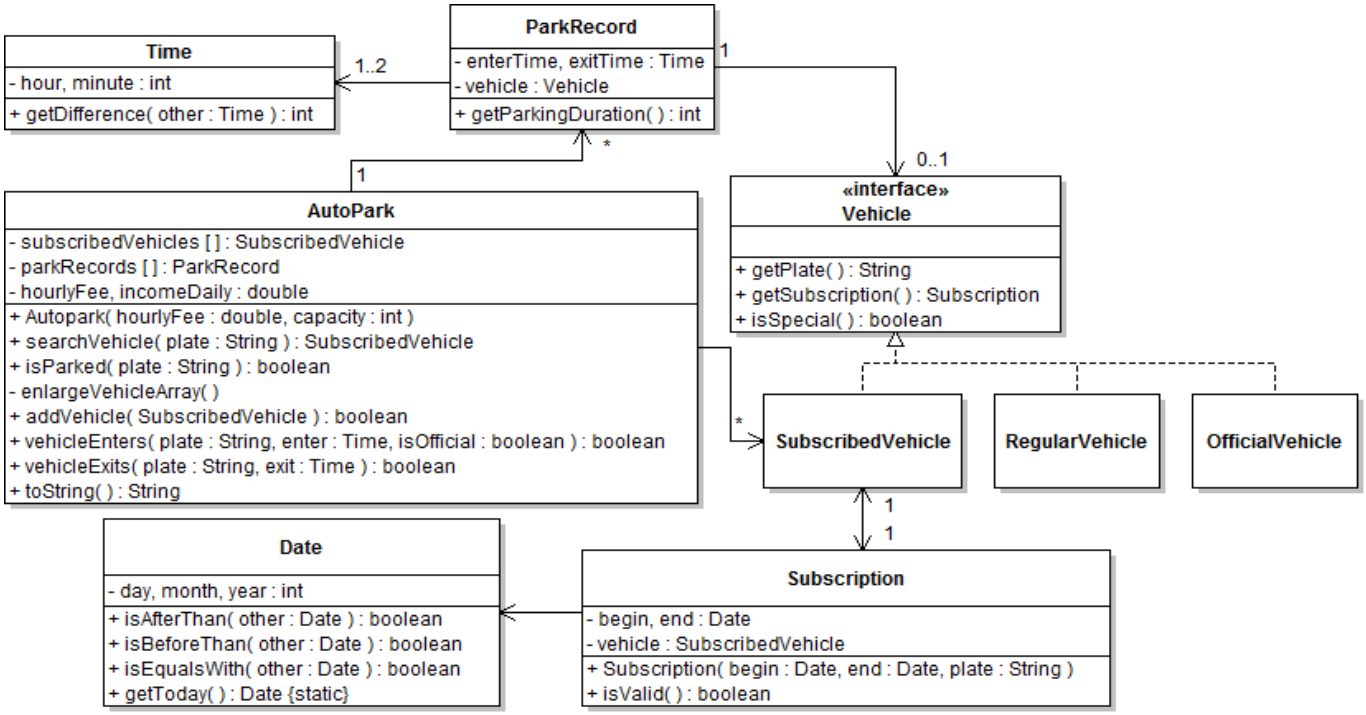
- All projects must be unique.

Specific Rules for Group 2 (YES):

- Some projects may not be unique: A team (CP) may choose to copy another team's (OR) unique code. CP must modify the unique code so that it becomes very hard for a programmer to understand that this is a copied & modified code. However, that team must put a text file their zip file that reveals the OR team's student numbers. Every OR team must give their code to only one other CP team. The projects' grade percentage will be distributed to the midterm percentages for CP teams.

PROJECT DESIGN:

You will implement a complete application for this project work. You must comply with the UML diagram below while coding. Please note that there may be hidden information (members, associations, etc.) in the diagram. Some other details that you need to know and must comply are given below:



About vehicles: A regular vehicle is a vehicle that does not have a subscription therefore it is subject to hourly parking fee. A subscribed vehicle always has a subscription, but it may be invalid (Hint: Notice isValid method of a subscription). An official vehicle is a vehicle that belongs to public service, i.e. police cars, ambulances, etc. They do not pay anything for parking; consider them as having permanently valid subscription.

About subscriptions: A subscription is invalid if today's date is not within the range of the subscription. The vehicle that is to be associated with a subscription is created in the constructor of the subscription.

About some methods of the class Autopark:

- addVehicle: Adds a vehicle to the table, if a vehicle with the same plate does not exist beforehand.
- isParked: Searches the vector for a vehicle with the given plate. In other words, it checks whether a vehicle with the given plate is parked or not.
- vehicleEnters: First of all, a vehicle that has already parked cannot enter again. When a vehicle enters to the park, its plate is searched within the vehicles array. This array contains vehicles with subscription. If such vehicle is found, the parking record that is to be created and to be added to the array is associated with that vehicle. Otherwise, this means that the vehicle is either a regular vehicle or an official vehicle. In this case, such a vehicle will be created and associated with the parking record. Official vehicles need not to be stored in the vehicles array as they can be easily distinguished by their markings.
- vehicleExits: This method determines whether the vehicle with the given plate, which is exiting the park, will pay a parking fee or not. If so, the fee to be paid is the multiplication of parking duration and the hourly fee. The parking fee must be added to the member incomeDaily.