Basics of Programming II - Course 2022/2023

Marina Sokolova & Jorge D. Laborda

Lab assignment 3.- Inheritance in Java

1. Goals of this assignment

In this practice we will incorporate a very important element of OOP and that we have already study in theory class: inheritance. In this first part of the practice We will program in Java the class hierarchy shown by the UML diagram of the Figure 1.

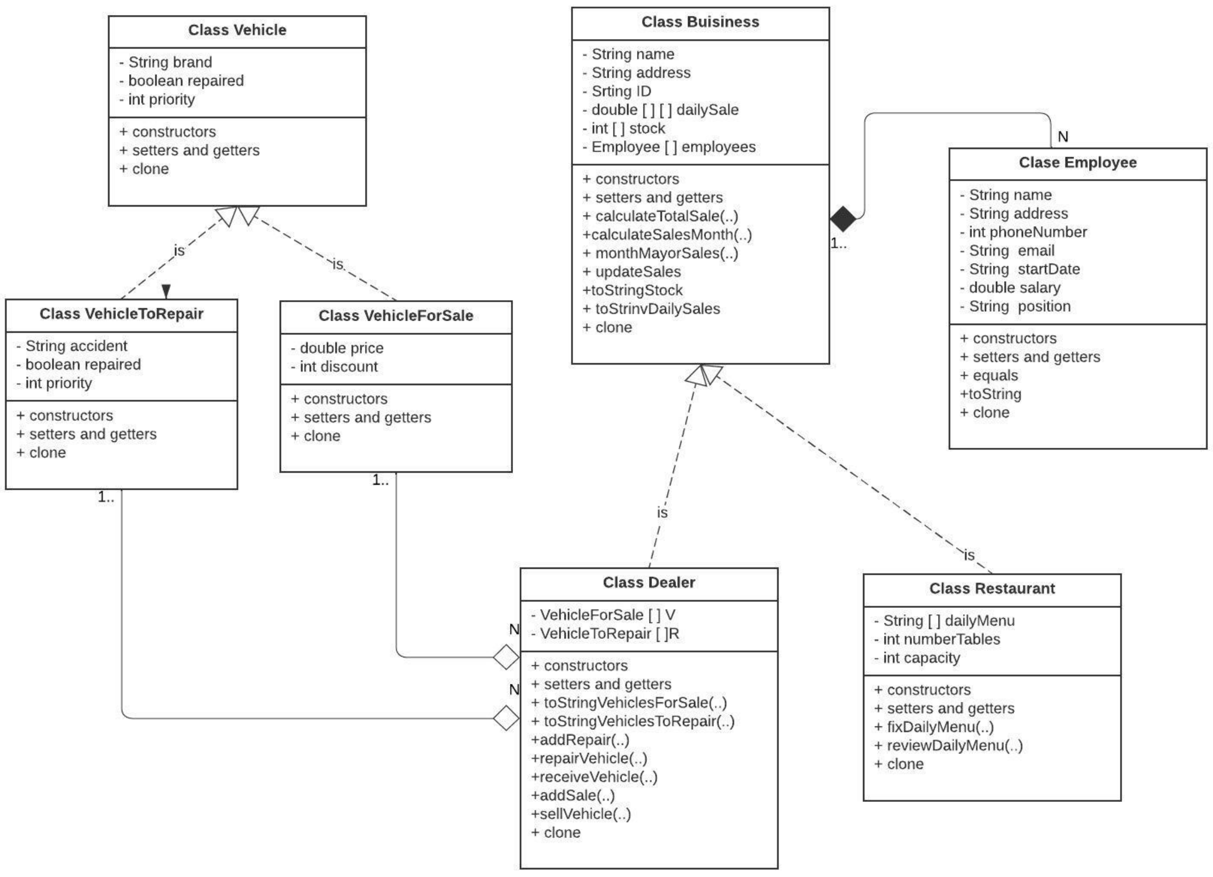

Figure 1: UML Diagram for lab assignment lab3.

2. Student´s work

- Implements the class hierarchy described in the UML diagram in Java, all of it in a package called **PaqComercio**.
- Assign the classes and the attributes of each class the access modifiers appropriate.
- Since the classes in the diagram, like other Java classes, are subclasses of Object, overrides the following methods for all classes: or is equal() or toString().
- To check your code, it is recommended that you use one or more core Java classes.

You will place these classes in another package different from the previous one called **PaqPruebas**. To be able to use the public classes of the **PaqComercio** package you will have to write this:

import PaqComercio.*

2.1. Description of the classes

3. Business Class

The Business Class has the following attributes and methods:

- **dailySale**: a 12 x 31 matrix where the amount is stored in each box for the total sales made for each day of the month.
- **stock**: each box stores the stock for each of the items in the trade.
- **calculateTotalVentas()**: returns the sum of all sales made by the trade from the *DailySales* array.
- **calculateSalesMonth(month)**: given a month passed as an argument, returns the total of sales made in that month.
- **monthMayorSales()**: returns the month in which the most sales have been made.
- **updateSales(amount)**: from the current day and month, the box will be updated of the matrix *DailySales* with an amount that is passed to the method as argument.
- **clone()**: makes a deep copy of the Business.

3.1. Dealer Class

The Dealer Class has the following attributes and methods:

- **addRepair(vehicle)**: adds a vehicle to the *VehiclesToRepair* vector (note that the vehicles to be repaired have an assigned priority and that the vector where they are stored has to be ordered based on that priority).
- **repairVehicle(index)**: given a position of the vector, sets the *repaired* attribute to true of the corresponding vehicle.
- **receiveVehicle(licensePlate)**: given a license plate, search the vector of *VehiclesToRepair* if there is a vehicle with that license plate and if it is repaired, it returns it and deletes it.
- **addSale(vehicle)**: adds a vehicle to the vector of vehicles to sell.
- **sellVehicle(index)**: given a position of the vector, it eliminates the vehicle that occupies that position.

3.2. Restaurant Class

The Restaurant Class has the following attributes and methods:

- **dailyMenu**: a vector where each position stores the menu of the day for the corresponding day of the week.
- **fixDilyMenu(menu, dayOfWeek)**: from a String that contains the menu of the day and from the day of the week, saves the menu in the *dailymenu* vector in the position correspondent.
- **reviewDailyManu(dayOfWeek)**: returns a String with the menu of a day.

3.3. VehicleToRepair Class

The VehicleToRepair Class has the following attributes and methods:

- **priority**: value between 1 and 3 where the value 1 corresponds to the vehicles that have higher priority when it comes to being repaired.