CSE112 Object Oriented Programing
Spring 2019, Project#3
Due: May 14, 2019 23:50

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

In this project, you can work in pairs. For every method in the source code, you have to specify (comment) who is responsible for that method. The demonstration will be done individually, so is grading.

Vehicle Hiring System

You are supposed to design and implement a Vehicle Hiring System that serves for the VPark company. Basically, there are different types of vehicles, some can be hired for different time periods, with different fees and may be with some extra costs. Some certain types of vehicles can be booked for future rentals, some can be delivered to the renter's position to be picked up, and some can be dropped off another location rather than the rental office. In some cases, the customer(s) may require a vehicle which can carry some extra load for example to move to another city. So, not only car(s), but also trucks can be offered to the customers.

Unless otherwise stated, all vehicles can be remotely delivered, and dropped off.

You are free to make any assumptions you wish.

`Vehicle` class:

- `plateNumber, numberOfTires`. Both set during construction, and obviously cannot be changed.
- `dailyFee`: different for every subtype.
- `getDailyFee():` returns `numberOfDays*dailyFee`.
- Every **vehicle can be rented**.

The vehicles are classified as Car and Truck.

`Car` class:

- `color, seatingCap, numOfDoors.` All can be changed.

Cars are classified as Sports, StationWagon(SW), and SUV.

- **`Sports`**: has horse power (HP) attribute. **Can be booked**.
- **`SW`** : loadingCap: small amount of load can be carried by SW.
- **`SUV`** has an additional attribute that stores the type of wheel drive(wd). Neither remote delivery, nor remote dropping off is possible.

`Truck` class:

- `loadingCap`: defines the maximum shipping capacity of a truck.
- Any truck **must be booked** at least 7 days before rental. Neither remote delivery, nor remote dropping off is possible.

Trucks are classified as Small Trucks and Transport Trucks.

- **`SmallTrucks`**.
- **`TransportTrucks`**: an additional attribute that shows whether the truck goes abroad or not. Neither remote delivery, nor remote dropping off is possible.

`bookMe():` takes the arguments specifying the starting, and ending dates of rental. If there is no available vehicle, `SorryWeDontHaveThatOneException` is thrown.

`cancelMe():` cancels the booking. If the date of cancellation is after the starting date of rental, `NoCancellationYouMustPayException` is thrown.

`rentMe()`: takes the arguments specifying the starting, and ending dates of rental, if allowed remote delivery and dropping off locations are also taken. If there is no available vehicle, `SorryWeDontHaveThatOneException` is thrown.

`dropMe()`: the vehicle is returned back, and the total price is shown.

`loadMe()`: takes the argument specifying the amount of additional load. If the total load of vehicle exceeds loadingCap, `OverWeightException` is thrown.

`VehiclePark` class: the test class has no direct access to vehicle instances. So, any kind of inquiry is sent to the `VehiclePark` with correct argument list(s). Then, `VehiclePark` class invokes the corresponding methods of vehicle instances.

- Stores the list of all vehicles, booked vehicles, rented vehicles, registered customers.
- `displayVehicles()`: displays the whole list of vehicles.
- `displayAvailableVehicles()`: takes a date pair, and lists the vehicles those are available between the given dates. Should be overloaded so that a particular type can be listed.
- `addVehicle()`: used to add a new vehicle, when bought, into the park. Only admin can invoke.
- `removeVehicle()`: used to remove a vehicle, when sold, from the park. Only admin can invoke.
- `bookVehicle(), cancelBooking(), rentVehicle(), dropVehicle(), load()`: all must have corresponding arguments, specified by the methods they're supposed to invoke. Only registered users can invoke.
- `dailyReport()`: used to prepare the daily report of rental system into a file, i.e., the lists of vehicles, and registered users. File name is passed as argument. Only admin can invoke.

`Person` class: Your application can be used by anyone. But if s/he is not registered only the below actions are allowed:

- `display()`, `displayAvailable()`
- `register()`: asks name, contact info (phone, address, etc.). If the Person instance is not already in the registered used list, it is added (Registered users should also be written into a file, and when the app is run, the file is loaded into memory.).

  There are two subclasses of Person:
  - **`RegisteredUser`**: has id, name, and contact info attributes. Registered users can book, rent vehicles; can cancel bookings, rentals; can load vehicles if allowed. Also, registered users can search the vehicles by type.
  - **`Admin`**: has id, name, contact info, userName. Can add new vehicles to the store, can remove vehicles, and can ask for daily reports by providing a file name.

`TestClass`: used as the main interface. When the app is run, the lists of customers, vehicles and admins are loaded from file. Note that, by default anyone running the app is an instance of "Person". Below is given a brief flow of menus. And, attached to this description, you can also find a **template** of the test class. You are supposed to implement the methods in test class.
Main menu:

- (Press 1) Display all vehicles
- (Press 2) Display available vehicles: asks the dates, returns to the main menu
- (Press 3) Register me: asks the name and contact info, does the registration, and returns to the main menu.

- (Press 4) Continue as registered user: asks the id, and calls userMenu.
- (Press 5) Continue as admin: calls adminMenu.
- (Press 6) Quit : quits the app, if necessary by updating the registered user file.

userMenu:

- (Press 1) Display all vehicles
- (Press 2) Display available vehicles: asks the dates, returns to the userMenu
- (Press 3) Display available vehicles by type: asks the dates, the vehicle type, returns to the userMenu
- (Press 4..n) Book Vehicle, cancel my booking, rent vehicle, drop vehicle. All returns to userMenu.
- (Press n+1) Quit: returns back to main menu.

adminMenu:

- (Press 1) Display all vehicles
- (Press 2) Display available vehicles: asks the dates, returns to the adminMenu
- (Press 3) Add a new vehicle to the system: asks the necessary attribute values, and adds the item into inventory. Returns to adminMenu.
- (Press 4) Remove vehicle: asks which item to be removed, removes from the inventory. Returns to adminMenu.
- (Press 5) Reports: creates daily reports. Returns to adminMenu.
- (Press 6) Quit: returns back to main menu. Note that, if necessary all files should be updated.