CSE 1241 - COMPUTER PROGRAMMING I
Programming Assignment # 4
DUE DATE: 12/01/2024 - 23:59 (No extension)

In this homework, you will implement a program with an object-oriented approach.

1. Implement a `Vehicle` class with the following UML diagram.

Vehicle	
-	plateNumber: String
-	size: int
+	Vehicle (plateNumber: String, size: int)
+	getPlateNumber(): String
+	getSize(): int
+	getVehicleInfo(): String

- A Vehicle object represents a vehicle with a license plate number and size.
- The data field of size may take the values of 1, 2, and 4.
  (*You may simply assume that 1: motorcycle, 2: car, 4: truck*)
- You have to use ***this*** keyword in the implementation of the constructor.
- Consider the following example which demonstrates the creation of a Vehicle object:
  ```
  Vehicle vehicle = new Vehicle("34CSE1141", 2);
  ```
- `getPlateNumber` method should return the plate number of the vehicle.
- `getSize` method should return the size of the vehicle.
- `getVehicleInfo` method returns a string containing the vehicle's license plate number and the size of the vehicle.
  An example string as the following:
  ```
  Vehicle Info
  Plate Number : 34CSE1141
  Size : 2
  ```

2. Implement a `ParkPlace` class with the following UML diagram.

ParkPlace	
-	size: int
-	vehicle: Vehicle
+	ParkPlace (vehicle: Vehicle)
+	getSize(): int
+	getVehicle(): Vehicle

- A ParkPlace object represents a suitable place that the vehicle will be parked.
  A ParkPlace object is created with a size of the vehicle and the vehicle itself.
- You have to use ***this*** keyword in the implementation of the constructor.

- Consider the following example which demonstrates the creation of a `ParkPlace` object:
  `ParkPlace parkPlace = new ParkPlace(vehicle);`
- `getSize` method returns the size of the `ParkPlace` object.
  - The size of the `ParkPlace` might be suitable for a motorcycle, a car, or a truck.
- `getVehicle` method returns the `Vehicle` object of the `ParkPlace` object.
  - This represents which vehicle is parked in that place.

3. Implement a `Ticket` class with the following UML diagram.

Ticket	
-	vehicle: Vehicle
-	entryDate: java.util.Date
-	exitDate: java.util.Date
-	totalPrice: double
+	numberOfTickets: int
+	Ticket(vehicle: Vehicle, entryDate: java.util.Date)
+	calculatePrice (hourlyPrice: double, exitDate: java.util.Date): double
+	getTicketInfo (): String
+	getVehicle(): Vehicle
+	getPrice(): double

- When a vehicle is parked, a Ticket object is created containing the vehicle itself and entry date of the vehicle.
- You have to use ***this*** keyword in the implementation of the constructor.
- Consider the following example which demonstrates the creation of a Ticket object:
  `Ticket ticket = new Ticket(vehicle, entryDate);`
- `CalculatePrice` method takes a double value of hourly price and exit date of the vehicle, then calculates the price of parking service by considering how many hours the vehicle is parked and the size of the vehicle. The `totalPrice` field of the `Ticket` object should also be updated in this method.
- You can calculate the parking cost with the following formula:

  Parking cost = vehicleSize * hourlyPrice * numberOfHours

  It should be noted that the numberOfHours value should be rounded up. (Example: If a car parked for 3 hours and 20 minutes, then you should charge the parking cost with 4 hours.)

- The `getTicketInfo` method returns a string value based on two conditions.

- If the vehicle does not exit from the car park up to now, then the string should have the vehicle's plate number and entry date.

  Example:
  ```
  Ticket Info
  Plate Number : 34CSE1141
  Entry : Fri Dec 15 11:03:48 EET 2017
  ```

- If the vehicle exited from the car park, then the string should have the vehicle's plate number, entry date, exit date, and the total price for parking service.
  Example:
  ```
  Ticket Info
  Plate Number : 34CSE1141
  Entry : Fri Dec 15 11:03:48 EET 2017
  Exit  : Fri Dec 15 16:03:48 EET 2017
  Hour  : 5
  Fee   : 50.0 TLs
  ```

- `getVehicle` method returns the `Vehicle` object of the `Ticket` object.
- `getPrice` method returns the `totalPrice` field of the `Ticket` object.

4. Implement a `CarPark` class with the following UML diagram.

CarPark	
-	capacity: int
+	parkPlaceArray: ParkPlace[]
+	ticketArray:Ticket[]
-	hourlyPrice: double
+	CarPark(capacity: int, hourlyPrice: double)
+	parkVehicle (vehicle: Vehicle, entryDate: java.util.Date): Ticket
+	exitVehicle (ticket: Ticket, exitDate: java.util.Date): Vehicle
+	getTotalIncome (): double
+	printVehicleList(): void
+	printTickets(): void

- A `CarPark` object has
  - a `capacity` indicating the total size of the car park,
  - a `ParkPlaceArray` which shows suitable park places,
  - a `TicketArray` indicating the tickets of the car park, and
  - a `hourlyPrice` field indicating the price of the parking service for an hour.

- A `CarPark` object is created with a capacity and an hourly price values.
- You have to use ***this*** keyword in the implementation of the constructor.
- Consider the following example which demonstrates the creation of a `CarPark` object with 100 place capacity and hourly price of 5 TLs:

```
CarPark park = new CarPark(100, 5);
```

- `parkVehicle` method is invoked when a new vehicle is entered to the car park. It takes the vehicle object and entry date of the vehicle as arguments. Inside the method you should perform the following in order:
    a) First of all, it should check whether there is a suitable place for that vehicle in the car park. If there is not enough space, it should print a warning message. Example: `Car park is full!`
    b) If there is enough space, it should create a new `ParkPlace` object with the `Vehicle` object.
    c) This method should create a new `Ticket` object using the `Vehicle` object and the entry date of the vehicle.
    d) After car is parked successfully, it should print a message such as: `The vehicle with 34CSE1141 plate number is parked.`
    e) In the end, this method should return a `Ticket` object.

- `exitVehicle` method is invoked when the vehicle exits from the car park. It takes the ticket object and exit date of the vehicle as arguments and performs the following in order:
    - It should search for the exact place of the vehicle in `parkPlaceArray` by considering license plate number of the vehicle.
    - It updates the exit date of the ticket object and calculates price of parking service.
    - It should print a message as the following: `The price for vehicle with 34CSE1141 plate number is: 50.0 TLs`
    - Then, it should save the ticket in the `ticketArray`.
    - Additionally, this method should reorganize `parkPlaceArray` by counting the filled park places and resize the `parkPlaceArray` based on that count. In other words, there should not be an empty place in `parkPlaceArray`.
    - In the end, this method should return the `Vehicle` object of the `Ticket` object.

- `getTotalIncome` method calculates the total income of the car park by considering the `ticketArray`.

- `printVehicleList` method prints out the vehicles currently in the car park. This method must invoke `getVehicleInfo`() method of the `Vehicle` object.

- `printTickets` method prints out the tickets processed until now. This method must invoke `getTicketInfo` method of the `Ticket` object.

5. Write a test program in which the following are performed in order.
    a. Create a new `CarPark` object with the capacity of 10 and the hourly price of 5 TLs.
    b. Create 5 vehicles with different license plate numbers and the following sizes:
        - 4, 2, 1, 2, 4.
    c. These vehicles should try to enter to the car park with one-hour delays.
        - You may assume that the first car will be entered in current time.
        - If the current time is `Fri Dec 15 03:02:20,` then the second car will be entered at `Fri Dec 15 04:02:20,` the third car will be entered at `Fri Dec 15 05:02:20,` etc.
    d. Assume that the car park is filled with the first four vehicles and there is no space for the fifth one.
    e. Print the content of vehicle list at that time.
        - Invoke `printVehicleList` method of `CarPark` object.
    f. After a while, two vehicles exit from the car park (you can pick a random value for the total number of hours).
    g. Invoke the `printVehicleList` method of the `CarPark` object.
    h. After a while, remaining vehicles exit from the car park.
    i. Print the total income earned until now by calling the `getTotalIncome` method of the `CarPark` object.
    j. Print the total number of tickets processed until now by accessing the `numberOfTicket` field of the `Ticket` class.
    k. Print the details of each tickets by invoking `printTickets` method of `CarPark` object.

This is a simple scenario to test your class implementations. There might be other test cases too. Therefore, please pay attention to use the same class, method and variable names in your implementations. You are allowed to increase the number of methods in the classes; however, you cannot decrease the number of them. *You are not allowed to use ArrayLists in the homework! You can only use Arrays.*

Submission Instructions

Please zip and submit your files using filename YourNumberHW4.zip
(ex: 150713852HW4.zip) to Canvas system (under Assignments tab).
Your zip file should contain the following 10 files:

1. 5 Java source files: Vehicle.java, ParkPlace.java, Ticket.java, CarPark.java, Test.java
2. 5 Java class files: Vehicle.class, ParkPlace.class, Ticket.class, CarPark.class, Test.class

Important Notes:

- ***You have to implement and use the methods listed above.***
- ***You can add more methods to your code if you need more.***
- ***You should print the values with two digits after the decimal point.***
- ***Your programs should execute correctly for different test cases.***
- ***Selected parts of your submissions will be graded! If you only submit the implementation of a single part, you might get a grade of 0!***

Please use the ***default package*** *in Eclipse IDE* for the assignments. Otherwise, the submitted code may not be compiled on another computer.

Submission Notes:

1. Write a comment at the beginning of each program to explain the purpose of the program.
2. Write your name and student ID as a comment.
3. Include necessary comments to explain your actions.
4. Select meaningful names for your variables and class names.
5. You are allowed to use the materials that you have learned in lectures & labs.
6. Do not use things that you did not learn in the course.
7. In case of any form of **copying and cheating** on solutions, all parts will get **ZERO** grade. You should submit your own work. In case of any forms of cheating or copying, both giver and receiver are equally culpable and suffer equal penalties.
   All types of plagiarism will result in zero grade from the homework.
8. No late submission will be accepted.
9. After completing your code, format your code! Please select all lines and press **Ctrl+Shift+F**