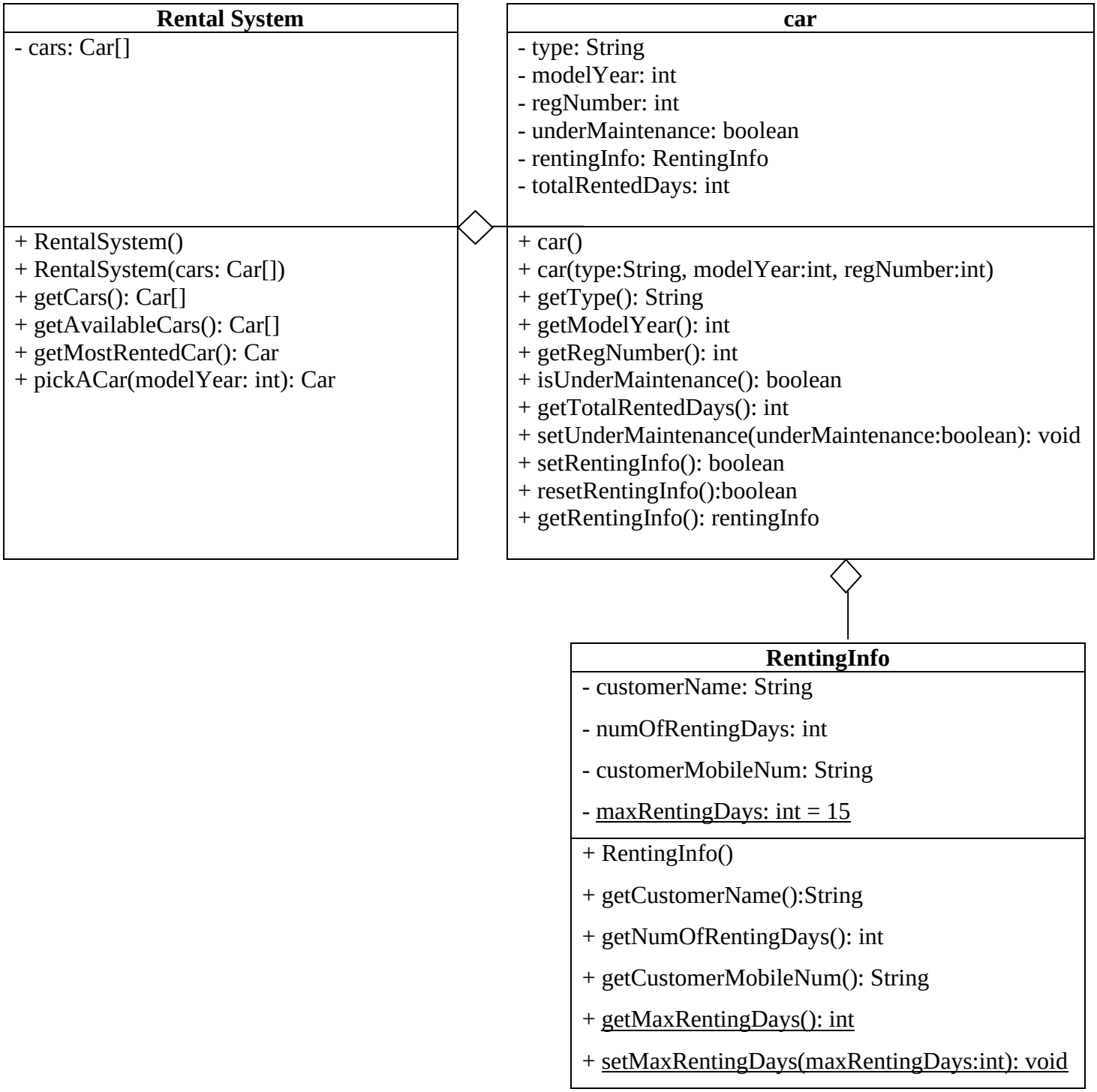
Object-Oriented Problem Solving Lab	Spring 2022
Midterm Exam	May 12th
الاسـم :	الرقم الجامعي:

Write a Java program that implements the system shown in the following UML class diagram, which represents a car rental system نظام تأجير سيارات:



1. **For the *RentingInfo* class:**
   - **The *numOfRentingDays* represents the number of days the car is rented for.**
   - **The *no-arg* constructor must read all instance data fields from the user. You must prompt the user to enter the values with appropriate messages. For the *numOfRentingDays* you must make sure that the entered value is positive and less than the *maxRentingDays*. If not, you must keep on prompting the user to enter a valid value with the message "Number of renting days must be between 0 and …" where .., is the value of *maxRentingDays*.**
   - **The mutator of *maxRentingDays* must modify the *maxRentingDays* only if the new value is greater than 3. Otherwise, it must print the statement "Could not modify maximum renting days".**

2. **For the *Car* class:**
   - **The *underMaintenance* (تحت الصيانة) data field must be always initialized to false, and the *totalRentedDays* data field must be always initialized to 0.**
   - **The *rentingInfo* data field must be always initialized to null. This field indicates whether the car is rented or not, if it contains null then the car is not rented. Otherwise, if it contains a reference, this means that the car is rented, and the reference is to a *RentingInfo* object that includes info of the car rental.**
   - **The *no-arg* constructor must initialize the *type*, *modelYear*, *regNum* by reading their values from the user.**
   - **The constructor *car(type:String, modelYear:int, regNumber:int)* must initialize the *type*, *modelYear*, *regNum* with the corresponding parameters values.**
   - **The *setRentingInfo* method is invoked when a car is to be rented to set the renting info of the car. It must first check that the *rentingInfo* is not already set (car is not rented) which means that it does not contain a reference. If so, the *rentingInfo* data field must be set by creating a new *rentingInfo* object using the *no-arg* constructor and the method must return true. Otherwise, if the *rentingObject* is not null (car is rented) it must return false.**
   - **The *resetRentingInfo* method is invoked whenever a car is returned from rent to reset the renting info of the car to indicate that it's not rented and update the total renting days. It must first check that the car is currently rented (*rentingInfo* data field is not null). If so, it must add the number of renting days of the *rentingInfo* data field to the *totalRentedDays* of the car, then it must set the rentingInfo data field to null and return true. Otherwise, if the *rentingInfo* is already null, the method must return false.**

3. **For the *RentalSystem* class:**
   - **The *no-arg* constructor must ask the user to enter the number of cars in the system, then create the *cars* array accordingly. Then, create the car objects using the *no-arg* constructor of the *Car* class.**
   - **The constructor *RentalSystem(cars: Car[])* must initialize the *cars* array using the passed array by creating a new array and new object for each element.**
   - **The *getAvailableCars* method must return an array that includes cars in the system that are available for renting. In order for a car to be available for renting, it must be unrented and not under maintenance.**
   - **The *getMostRentedCar* method must return a reference to the car that has been most rented: the car with the <u>maximum total rented days</u>.**
   - **The *pickACar* method must take a model year, then pick a car randomly with a model year that is newer (greater) than the passed year.**

4. **In your main class:**

- **Define a method named *printCar* that takes a car object and prints its details as follows:**

  Car type: ….
  Car registration number: …
  Car model year: …

  Then if the car is under maintenance the method must print on a new line: *Car is under maintenance*. If not, it must print: *Car is not under maintenance*.

  Then based on the value of *rentingInfo* of the car:
  - **If the car is not rented it must print: *The car is not rented*.**
  - **If the car is rented it must print: The car is printed to (customer name in *rentingInfo*) with mobile number (customer mobile number in *rentinInfo*) for (number of renting days in *rentingInfo*).**

- **In your main method:**
  - **Create a *RentalSystem* object using the *no-arg* constructor of the *RentalSystem* class.**
  - **Continuously display the following menu to the user and take the number that represents the user choice to know what he wants to do:**

    What to do :

    1. ***Rent a car:* This option must read a model year from the user, then invoke the method *pickACar* and pass the entered model year. For the picked car, the details of the car must be printed by invoking the *printCar* method and the method *setRentingInfo* must be invoked on the picked car to enter *rentingInfo*.**
    2. ***Return a car:* this option must take the registration number of the car to be returned from the user, then find the car in the arrays of cars of the rental system, and invoke the method *resetRentingInfo* on the car.**
    3. ***Print most rented car:* this option must invoke the *getMostRentedCar* method of the rental system. Then print the details of the returned car by invoking the *printCar* method.**
    4. ***Print all available cars:* this option must invoke the *getAvailableCars* method, and print the available cars info in a table as follows:**

    Car Registration Number Car Type Car Model Year Total Rented days

    .
    .
    .

Good Luck 😊