King Saud University
College of Computer and Information Sciences
CSC111
Lab Final Exam

You need to write a program for a system that manages Cars in a car dealer shop. Here is the UML diagram:

Car	3 marks
- id: int - model: String - year: int - price: double	1
+ Car() + Car(id: int, model: String, year: int, price: double) + setters + getters	0.5 0.5 0.5 0.5

CarDealer	9 marks
- cars[]: Car - nCars: int + MAX_SIZE: `public static final` int	1
+ CarDealer() + getnCars(): int + findCar(i: int): int + addCar(id: int, model: String, year: int, price: double) : void + findMaxPrice () : int + findMinByModel (model: String): int + display(i: int): void	1 0.5 1 2 1.5 1 1

TestCarDealer	3 marks
+ main()	

The Class: Car

As shown in the UML diagram, write the class Car that has the attributes:

- `id`: the id of the Car,
- `model`: represent the model of the Car for example: “Toyota”
- `year:`: represents which year the car has been made for example: 2015
- `price:`: represents the price of the Car.

The methods of this class are:

- **Car**(): A default constructor. Put -1 in all attributes and the string “NA” in the model.
- **Car**(id, model, year, price): A constructor that initializes new Car with the initial values from the user.
- **Setter methods** (one for each): That sets the values for: id, model, year, price.
- **Getter Methods** (one for each): That returns the values of: id, model, year, price.

The Class: CarDealer

In the class `CarDealer` contains one array of objects that holds all the Cars of a car dealer shop. It also contains the variable nCars that stores the current number of Cars stored in the array of objects `cars[]`.
Note that the maximum number of Cars in the list is 100 (*hint: MAX_SIZE constant)*.
The methods of this class are:

- **CarDealer**: a constructor that initializes the attributes and creates an array of Cars of size *MAX_SIZE*
- **getnCars**: returns the current number of cars.
- **findCar:** this method receives id and search for that id in the array if found returns the index of the Car with that id or -1 if not found.
- **addCar**: this method will add a Car to the list but first you need to check if the id is not already entered to add. If the id is already in the array you should print “**Cannot add this car because the id is already in the list**”. If it is not possible to add the Car because the array is full, you should print an error message “**ERROR ADDING LIST IS FULL**”.
- **findMaxPrice**: returns the index of the first Car in the array with the maximum price. If it is not found, -1 is returned.
- **display**: if there is a car in index i, displays the Car details in that index otherwise print ERROR.

- **findMinByMode:** this method receives a ***model*** then return the index of the car that has the minimum price of the same ***model***. Otherwise, return -1.

The Main Class: TestCarDealer

- **main**: the `main` method will do the following:
  1. Create a `CarDealer` object.
  2. Ask the user to enter Car information or -1 to exit
  3. Display the Maximum price Car details
  4. then ask the user to enter a car model then print the car information of the same model with the minimum price.

Sample run:

```
Please enter car id (or -1 to exit): 123
Please enter the car model: audi
Please enter the year of the car: 2016
Please enter the price of the car: 100000
Please enter car id (or -1 to exit): 456
Please enter the car model: BMW
Please enter the year of the car: 2017
Please enter the price of the car: 150000
Please enter car id (or -1 to exit): 678
Please enter the car model: audi
Please enter the year of the car: 2016
Please enter the price of the car: 90000
Please enter car id (or -1 to exit): -1
The car with maximum price is:
Car ID: 456
Car model: BMW
Car year: 2017
Car price: 150000.0
Please enter the car model to find the minimum price of that model:audi
The car with minimum price of the model audi is:
Car ID: 678
Car model: audi
Car year: 2016
Car price: 90000.0
```