Term 4 Assignment 3

Quicka Self-driving

You are the Lead Systems Engineer for the Quicka ridesharing platform and are in charge of developing all the necessary software for drivers, riders, and self-driving cars.

For this assignment, you are asked to write the Control Operation System (aka COS-101) software for the Quicka Autonomous Vehicle. The computers onboard the QAV are outfitted with a JVM capable of running Java code.

When designing and writing your systems, you may use any IDE of your choice, but you must ensure that your code is able to run on the JVM.

Mission 4.3: Charging Stations

In this mission, we are tasked with developing an Charging Station network for the Quicka Autonomous Vehicles that can serve as charging and docking points for delivery vehicles.

Task 1: Docking

We would like to define a universal interface for our charging stations and vehicles for docking. This way, when we develop new vehicles or stations we can implement the universal interface and know that it will behave in the same way.

Instructions

1. Create a Java interface called `Dockable` which has the following methods specified:
   - a method called `canDock` that takes no parameters and returns a boolean
   - a method called `dock` that takes a `Vehicle` as a parameter and returns an integer

Task 2: Vehicles and Stations

Now we want to test our docking interface, but first we want to define an abstract class for our vehicles from which we will model all of our designs after.

Instructions

1. Create an abstract class called `Vehicle` that has the following properties:
    - a private attribute called `name` of type String
    - a protected attribute called `chargeRemaining` of type double
    - a protected attribute called `maxChargeCapacity` of type double
    - appropriate getters/setters for the attributes
    - an abstract method called `printInfo`
    - an abstract method called `calculateTotalChargeNeeded` that returns a double
2. Create a class called `ChargingStation` that uses the `Dockable` interface
3. Make sure that the `ChargingStation` class has the following properties:
    - a private array of `Vehicle` objects that represents the docked vehicles
        - The index of each object represents its docking number/bay
4. Implement all of the necessary methods in `ChargingStation`
    - `canDock` returns true if there is space for a vehicle to dock or false if the docking array is full
    - `dock` must take a `Vehicle` as a parameter and add it to the array of docked vehicles:
        - if the vehicle is added to the array, return the index (charging port number) of the vehicle
        - if the vehicle cannot be added to the array return -1
    - `calculateTotalChargeNeeded` must return the charge needed of the all of the docked vehicles
        - if the `ChargingStation` has one vehicle docked that has a maximum charge capacity of 500 kAH with a chargeRemaining of 200 kAH then `calculateTotalChargeNeeded` should return `300`
        - if the `ChargingStation` has two vehicles docked that both have a maximum charge capacity of 500 kAH with a chargeRemaining of 100 kAH then `calculateTotalChargeNeeded` should return `800`
    - `printInfo` must print out the name of the `ChargingStation` and a list of the docked vehicles (see sample output)
        - if no vehicles are docked then it should print `none`
5. Create a subclass of `Vehicle` called `TestVehicle` that implements the abstract methods of the `Vehicle` class:
    - `printInfo` should print out the name and weight of the `TestVehicle`
    - `calculateTotalChargeNeeded` should return the `maxChargeCapacity` less the `chargeRemaining` of the `TestVehicle`

Hint: the docking array may be partially filled.

Sample Input/Output

The following demo code produces the sample output:

```
public class AssignmentDemo {

  public static void main(String[] args) {
    ChargingStation chargePrime = new ChargingStation("Prime", 5);
    TestVehicle dragon = new TestVehicle("Dragon", 500);

    dragon.setChargeRemaining(100);

    System.out.println("> Station info:");
    chargePrime.printInfo();

    System.out.println("> A vehicle can dock at the station: " +
                       chargePrime.canDock());

    System.out.println("> Approaching vehicle info:");
    dragon.printInfo();

    int bayNumber = chargePrime.dock(dragon);
    System.out.println("> Vehicle docked in bay number " + bayNumber);

    System.out.println("> Station info:");
    chargePrime.printInfo();

    System.out.println("> A vehicle can dock at the station: " +
                       chargePrime.canDock());

    TestVehicle unicorn = new TestVehicle("Unicorn", 500);
    unicorn.setChargeRemaining(100);

    System.out.println("> Approaching vehicle info:");
    unicorn.printInfo();

    bayNumber = chargePrime.dock(unicorn);
    System.out.println("> Vehicle docked in bay number " + bayNumber);

    System.out.println("> Station info:");
    chargePrime.printInfo();

    System.out.println("> A vehicle can dock at the station: " +
                       chargePrime.canDock());
  }
}
```

```
> Station info:
Name: Prime
Charge needed: 0.0
```

```
Docked vehicles:
none
> A vehicle can dock at the station: true
> Approaching vehicle info:
Name: Dragon
Charge: 20.0%
> Vehicle docked in bay number 0
> Station info:
Name: Prime
Charge needed: 400.0
Docked vehicles:
Dragon
> A vehicle can dock at the station: true
> Approaching vehicle info:
Name: Unicorn
Charge: 20.0%
> Vehicle docked in bay number 1
> Station info:
Name: Prime
Charge needed: 800.0
Docked vehicles:
Dragon
Unicorn
> A vehicle can dock at the station: true
```

Submission Instructions

Submit only your files `Dockable.java`, `Vehicle.java`, `ChargingStation.java`, and `TestVehicle.java` in a compressed `.zip` folder labeled with your student number an underscore and `assignment6`.

For example if your student number is `12345` submit a file `12345_assignment6.zip` that contains `Dockable.java`, `Vehicle.java`, `ChargingStation.java`, and `TestVehicle.java`.

Late submissions incur a 10% penalty per day maximum of 5 days late.

Marking

Section	Marks
Task 1: Docking	5
Task 2: Vehicles and Stations	*24*
Task 2.1: Vehicle	7
Task 2.2: ChargingStation	13
Task 2.5: TestVehicle	4

Section	Marks
Total	**29**