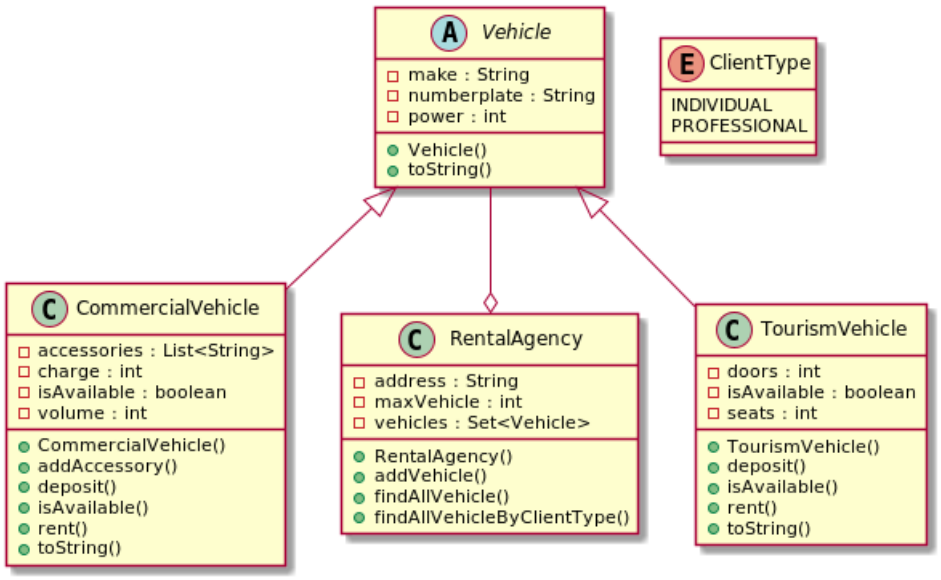
Programmation Objet Avancée - TP4

—o000o——o000o—

Classes, héritage et polymorphisme

1 Agence de location

Nous souhaite implémenter un programme permettant de consulter les véhicules d'une agence de location à partir de ce diagramme de classe simplifié :



1. Créez les classes de ce diagramme en vous appuyant sur la description ci-dessous, en respectant le principe d'encapsulation et en maximisant la factorisation de votre code

   - `Vehicle` est une classe abstraite ayant pour attributs :
     - **numberplate** de type `String`
     - **power** de type `Integer`
     - **make** de type `String`

     Cette classe doit contenir également un constructeur et la redéfinition de `String toString()`.
   - `TourismVehicle` est une classe qui étend la classe `Vehicle` et qui à pour paramètres :
     - **doors** de type `Integer`
     - **seats** de type `Integer`

     Cette classe doit contenir également un constructeur et la redéfinition de `String toString()`.
   - `CommercialVehicle` est une classe qui étend la classe `Vehicle` et qui à pour paramètres :
     - **volume** de type `Integer`
     - **charge** de type `Integer`

     Cette classe doit contenir également un constructeur et la redéfinition de `String toString()`.
   - `RentalAgency` est une classe ayant pour attributs :
     - **address** de type `String`
     - **vehicles**

- **maxVehicle** de type `Integer`

Cette classe fourni également les méthodes suivantes :

- `addVehicle(Vehicle vehicle)`. Cette méthode ne permet pas d'ajouter 2 fois le même véhicule. Elle ne permet pas non plus d'ajouter plus de véhicules que le nombre autorisé
- `findAllVehicle()` permettant de retourner les descriptions de tous les véhicules de l'agence
- `findAllVehicleByClientType(ClientType clientType)` permettant de retourner les descriptions des véhicules de l'agence selon le type du client (INDIVIDUAL, pour les véhicules de tourisme, PROFESSIONAL, pour les véhicules utilitaires)

Solution :

```java
package poo.tp4.agency;

public abstract class Vehicle implements Rentable {
    private final String numberplate;
    private final int power;
    private final String make;
    private boolean isAvailable = true;

    public Vehicle(String numberplate, int power, String make) {
        this.numberplate = numberplate;
        this.power = power;
        this.make = make;
    }

    @Override
    public void rent() {
        if (!isAvailable) {
            throw new IllegalStateException();
        }

        isAvailable = false;
    }

    @Override
    public boolean isAvailable() {
        return isAvailable;
    }

    @Override
    public void deposit() {
        isAvailable = true;
    }

    @Override
    public String toString() {
        return "Vehicle{" +
            "numberplate='" + numberplate + '\'' +
            ", power=" + power +
            ", make='" + make + '\'' +
```

```
            ", isAvailable=" + isAvailable +
            '}';
    }
}
```

java/Vehicle.java

Solution :

```
package poo.tp4.agency;

import java.util.ArrayList;
import java.util.List;

public class CommercialVehicle extends Vehicle implements
    Customizable {
    private final int volume;
    private final int charge;
    private final List<String> accessories = new ArrayList<>();

    public CommercialVehicle(String numberplate, int power, String
        make, int volume, int charge) {
        super(numberplate, power, make);
        this.volume = volume;
        this.charge = charge;
    }

    @Override
    public void addAccessory(String accessory) {
        accessories.add(accessory);
    }

    @Override
    public String toString() {
        return "CommercialVehicle{" +
            "volume=" + volume +
            ", charge=" + charge +
            ", accessories=" + accessories +
            "} " + super.toString();
    }
}
```

java/CommercialVehicle.java

Solution :

```
package poo.tp4.agency;

public class TourismVehicle extends Vehicle {
    private final int doors;
    private final int seats;
```

```java

    public TourismVehicle(String numberplate, int power, String
        make, int doors, int seats) {
        super(numberplate, power, make);
        this.doors = doors;
        this.seats = seats;
    }

    @Override
    public String toString() {
        return "TourismVehicle{" +
            "doors=" + doors +
            ", seats=" + seats +
            "} " + super.toString();
    }
}
```

java/TourismVehicle.java

Solution :

```java
package poo.tp4.agency;

import java.util.HashSet;
import java.util.Set;

public class RentalAgency {
    private final String address;
    private final Set<Vehicle> vehicles = new HashSet<>();
    private final int maxVehicle;

    public RentalAgency(String address, int maxVehicle) {
        this.address = address;
        this.maxVehicle = maxVehicle;
    }

    public void addVehicle(Vehicle vehicle) {
        if (vehicles.size() >= maxVehicle) {
            throw new IllegalArgumentException("Number of vehicles
                reached!");
        }

        vehicles.add(vehicle);
    }

    public void findAllVehicle() {
        vehicles.forEach(System.out::println);
    }

    public void findAllVehicleByClientType(ClientType clientType) {
        switch (clientType) {
            case INDIVIDUAL:
```

```
                    vehicles.stream().filter(v -> v instanceof
                        TourismVehicle).forEach(System.out::println);
                    break;
                case PROFESSIONAL:
                    vehicles.stream().filter(v -> v instanceof
                        CommercialVehicle).forEach(System.out::println);
                    break;
            }
        }
}
```

java/RentalAgency.java

Solution :

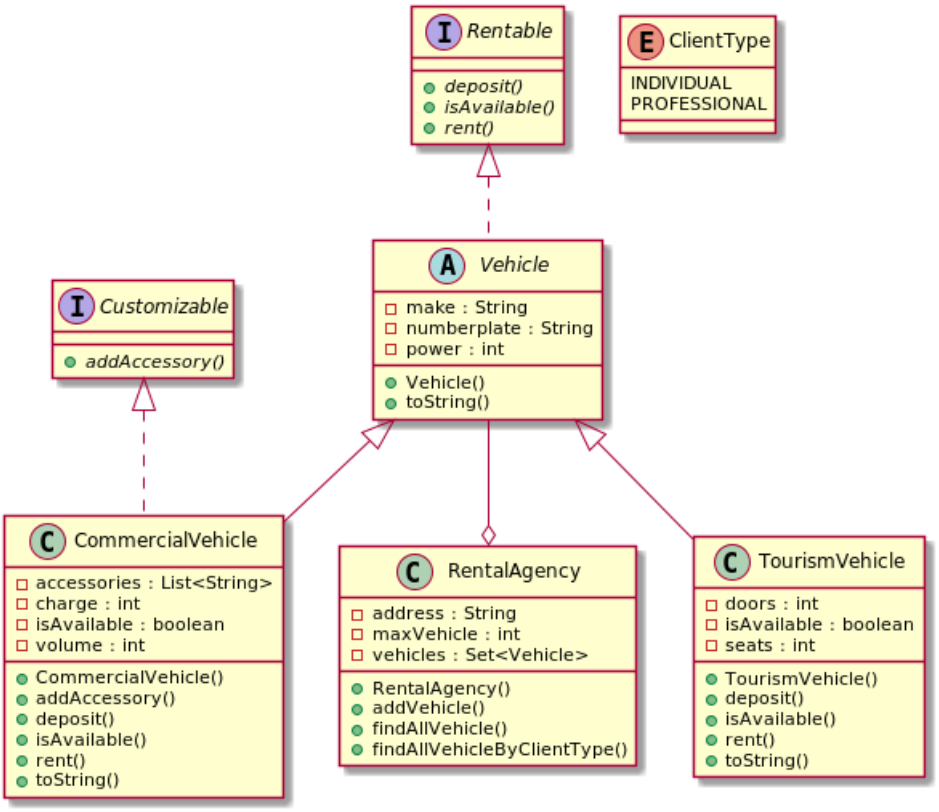
```
package poo.tp4.agency;

public enum ClientType {
    PROFESSIONAL, INDIVIDUAL
}
```

java/ClientType.java

2. Écrire une classe de test

2 Agence de location, la suite

Nous souhaitons mettre en place un système de location de véhicules ainsi que la personnalisation des véhicules utilitaires. Pour cela, considérons le diagramme de classe suivant :



- Créez les interfaces `Rentable` et `Customizable` avec les méthodes indiquées
- Apportez les modifications nécessaires aux différentes classes

Solution :

```java
package poo.tp4.agency;

public interface Rentable {
    void rent();

    boolean isAvailable();

    void deposit();
}
```

java/Rentable.java

Solution :

```java
package poo.tp4.agency;

public interface Customizable {
    void addAccessory(String accessory);
}
```

java/Customizable.java