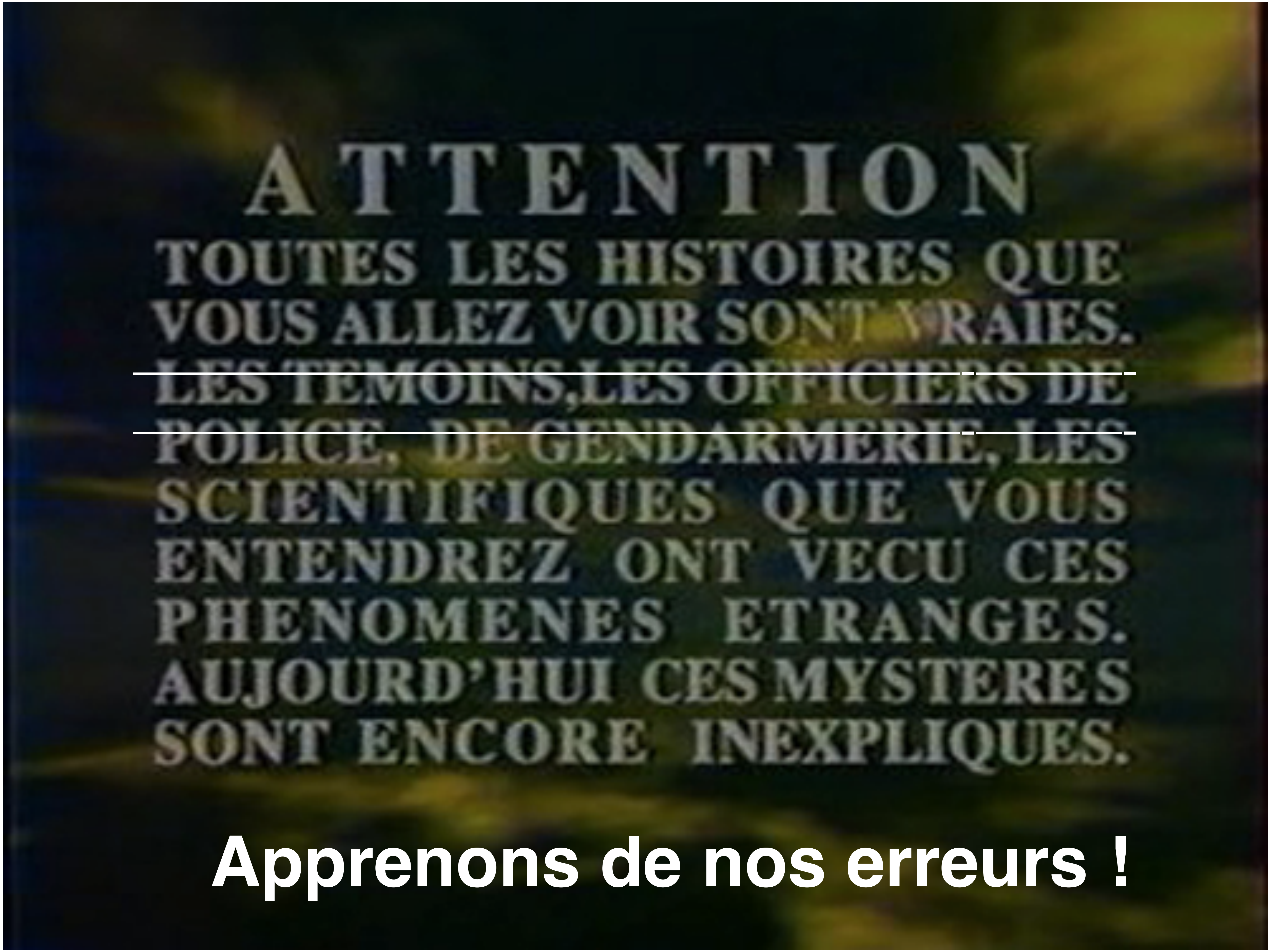


ATTENTION

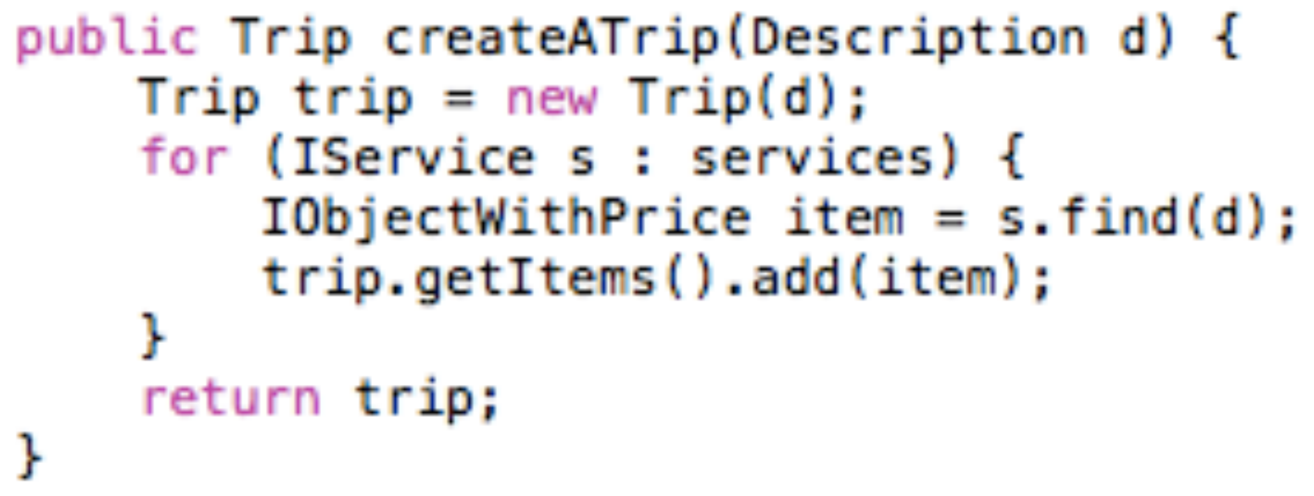
TOUTES LES HISTOIRES QUE VOUS ALLEZ VOIR SONT VRAIES. ~~LES TEMOINS,LES OFFICIERS DE POLICE, DE GENDARMERIE, LES~~ SCIENTIFIQUES QUE VOUS ENTENDREZ ONT VECU CES PHENOMENES ETRANGES. AUJOURD'HUI CES MYSTERES SONT ENCORE INEXPLIQUES.

Apprenons de nos erreurs !

A quoi cela correspond?

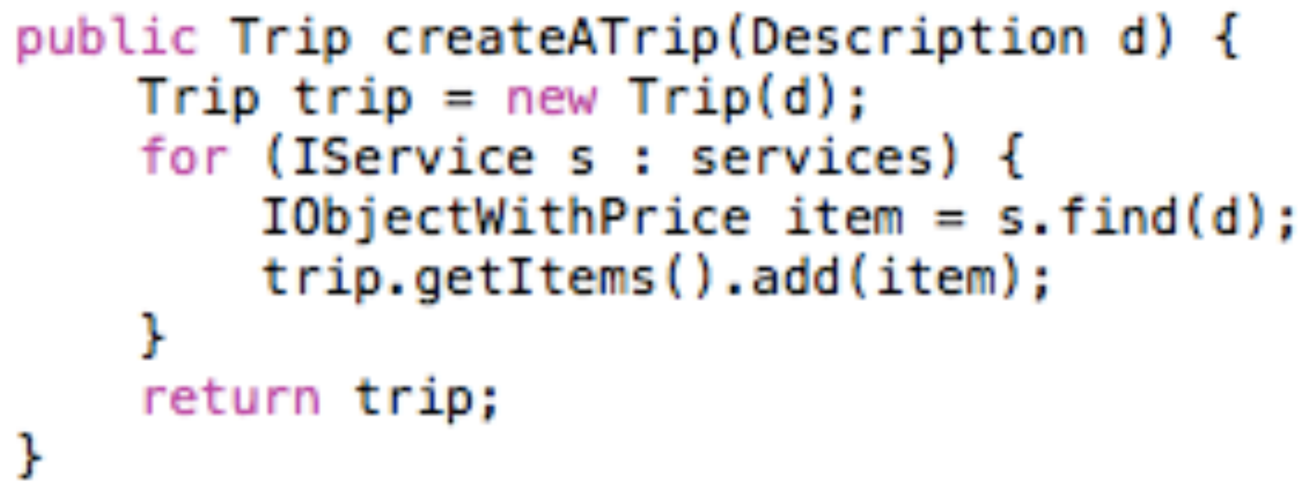
```
public class Vehicules
```

Responsabilités?



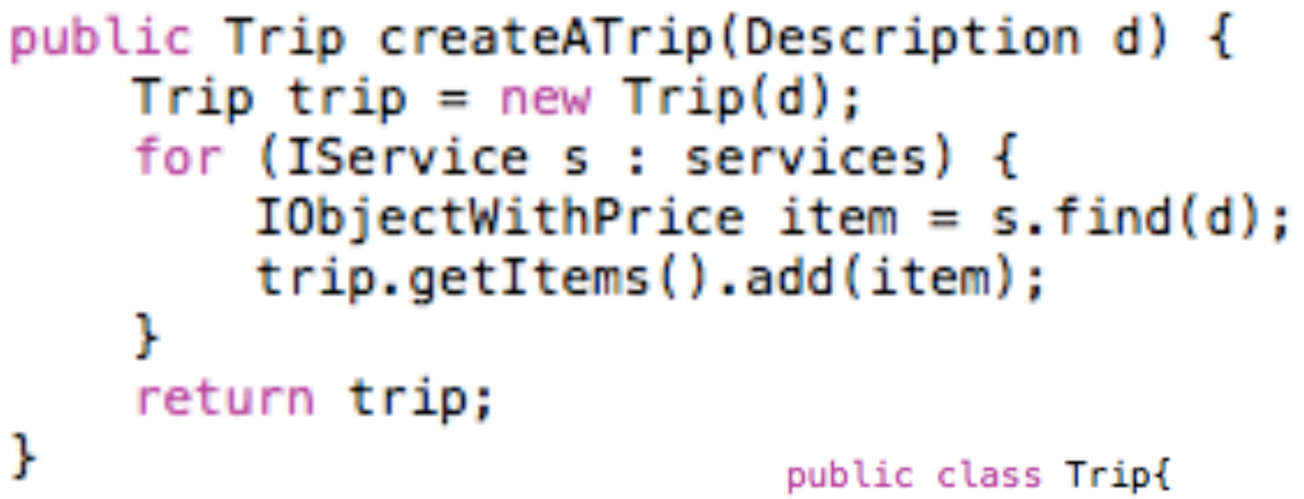
```java
public Trip createATrip(Description d) {
    Trip trip = new Trip(d);
    for (IService s : services) {
        IObjectWithPrice item = s.find(d);
        trip.getItems().add(item);
    }
    return trip;
}
```

Responsabilités?

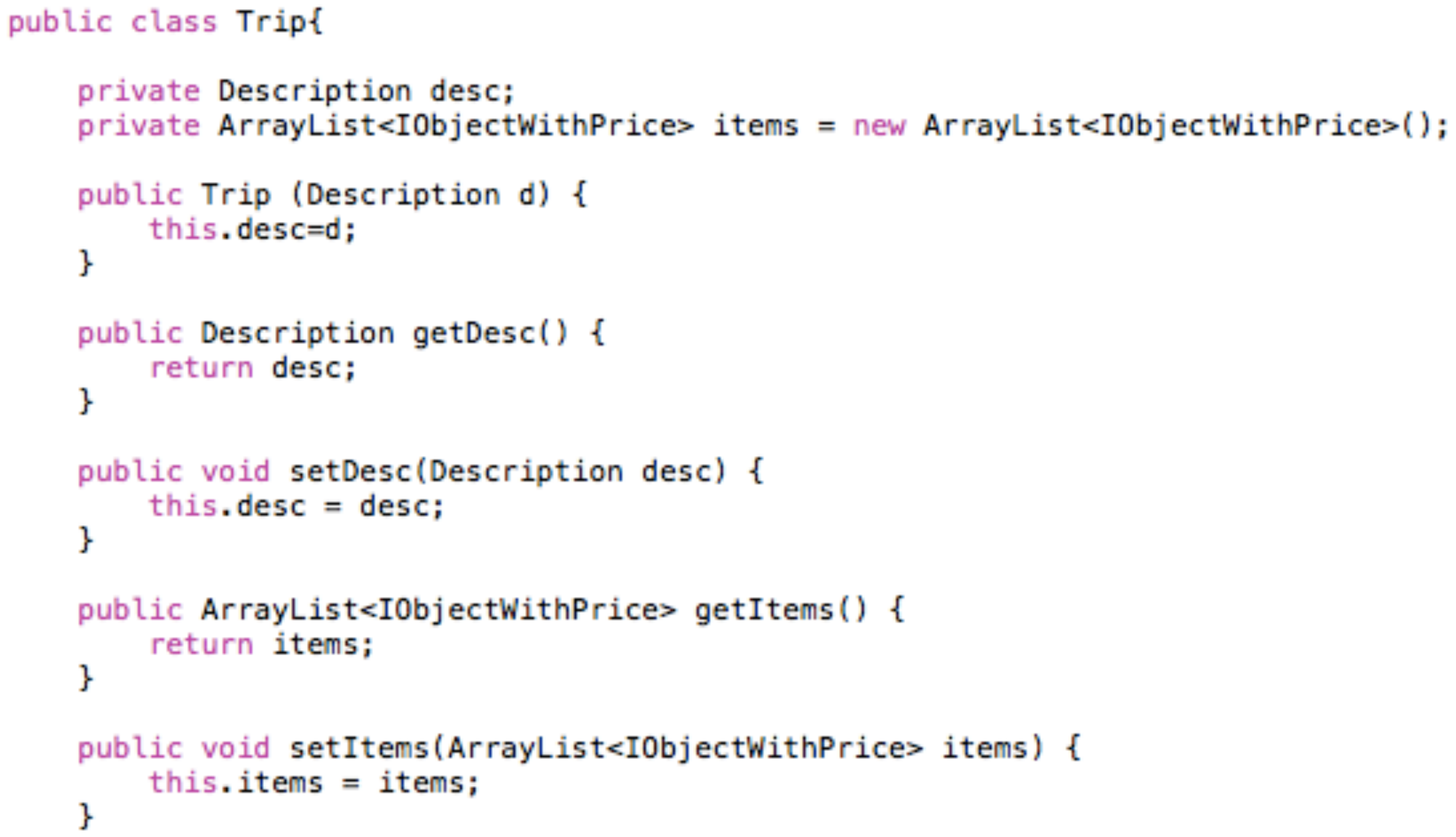


```
public Trip createATrip(Description d) {
    Trip trip = new Trip(d);
    for (IService s : services) {
        IObjectWithPrice item = s.find(d);
        trip.getItems().add(item);
    }
    return trip;
}
```

Responsabilités?



```
public Trip createATrip(Description d) {
    Trip trip = new Trip(d);
    for (IService s : services) {
        IObjectWithPrice item = s.find(d);
        trip.getItems().add(item);
    }
    return trip;
}
```



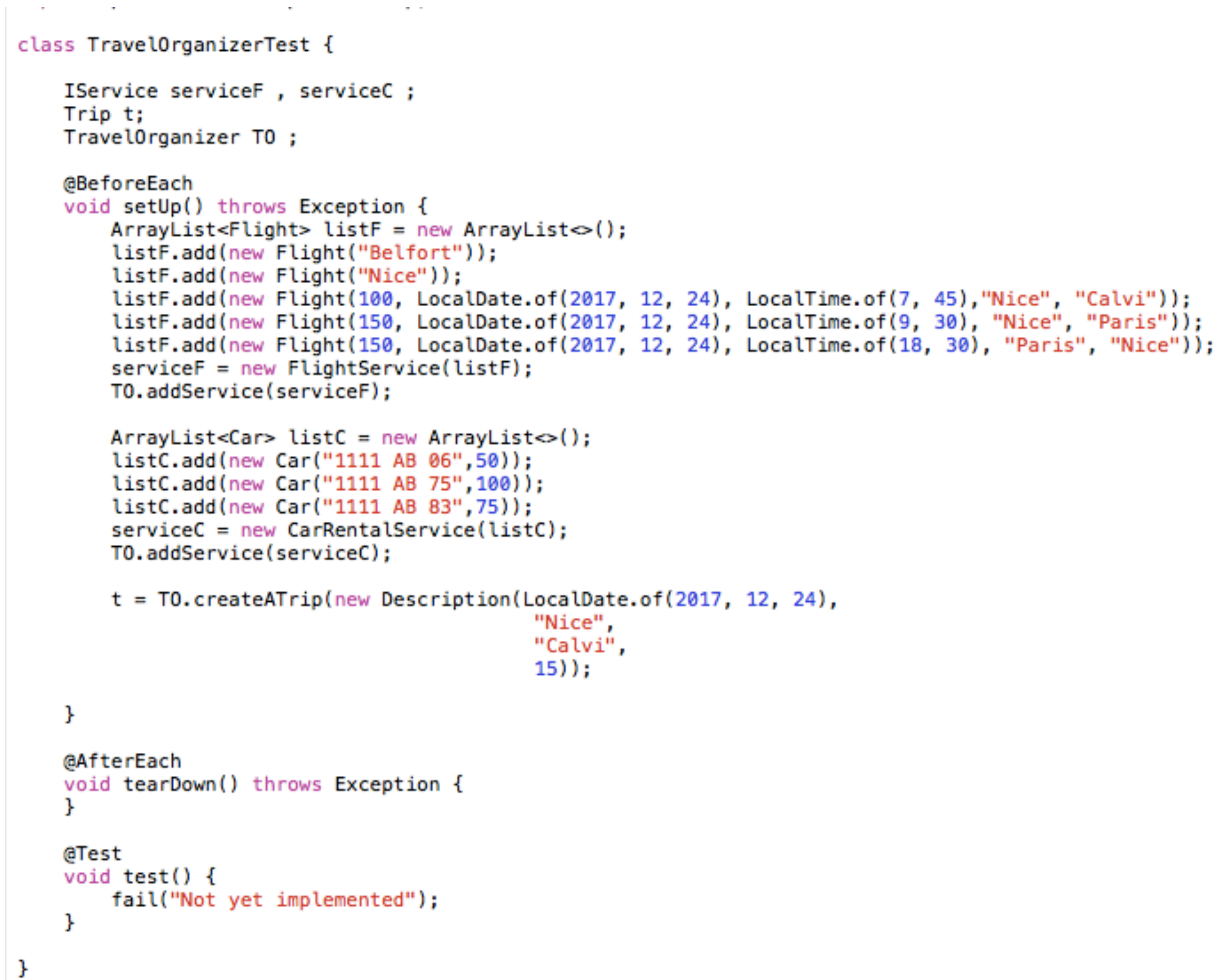
```
public class Trip{

    private Description desc;
    private ArrayList<IObjectWithPrice> items = new ArrayList<IObjectWithPrice>();

    public Trip (Description d) {
        this.desc=d;
    }

    public Description getDesc() {
        return desc;
    }

    public void setDesc(Description desc) {
        this.desc = desc;
    }

    public ArrayList<IObjectWithPrice> getItems() {
        return items;
    }

    public void setItems(ArrayList<IObjectWithPrice> items) {
        this.items = items;
    }
```

Responsabilités?

```java
public Trip createATrip(Description d) {
    Trip trip = new Trip(d);
    for (IService s : services) {
        IObjectWithPrice item = s.find(d);
        trip.getItems().add(item);
    }
    return trip;
}
```

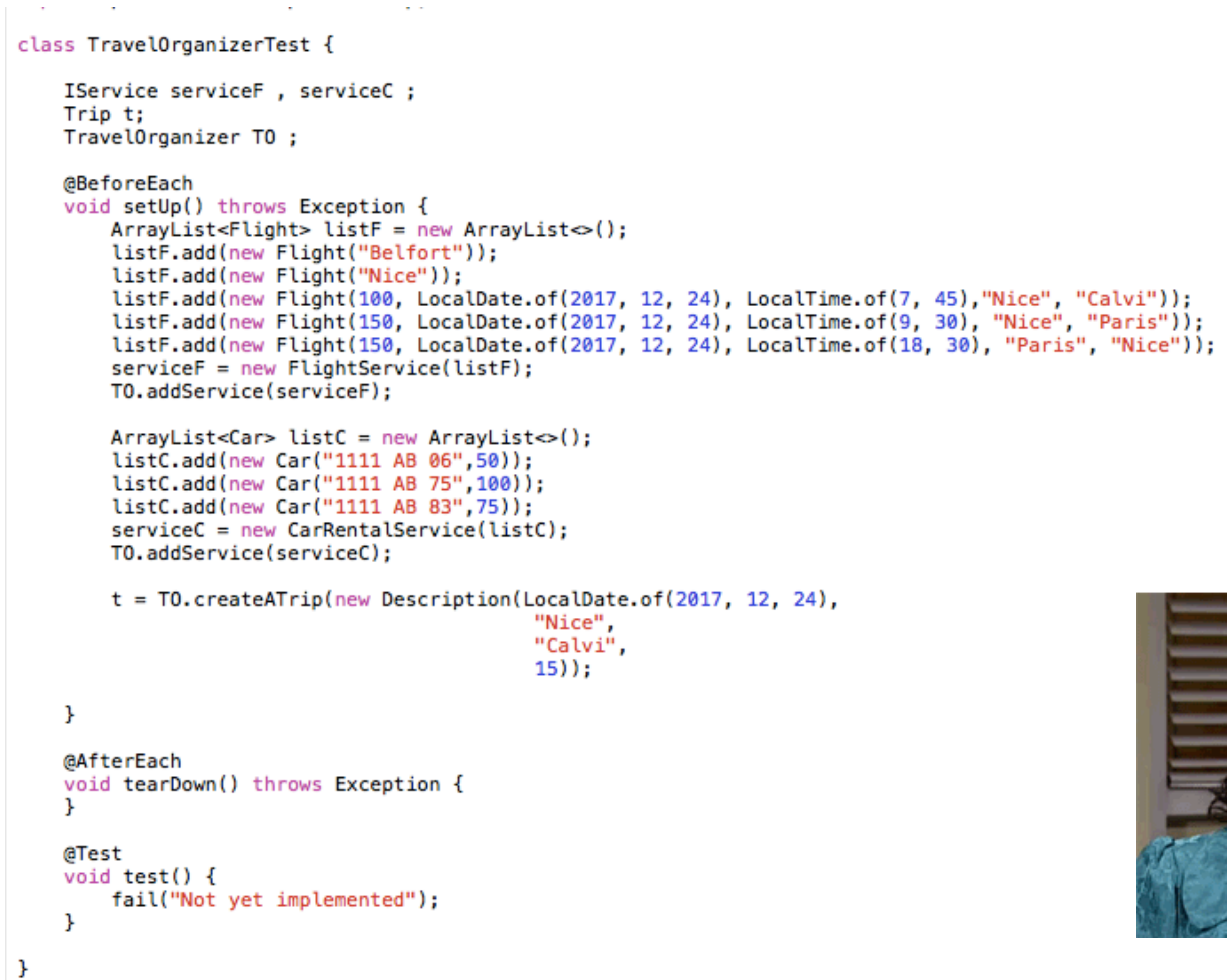
```java
public class Trip{

    private Description desc;
    private ArrayList<IObjectWithPrice> items = new ArrayList<IObjectWithPrice>();

    public Trip (Description d) {
        this.desc=d;
    }

    public Description getDesc() {
        return desc;
    }

    public void setDesc(Description desc) {
        this.desc = desc;
    }

    public ArrayList<IObjectWithPrice> getItems() {
        return items;
    }

    public void setItems(ArrayList<IObjectWithPrice> items) {
        this.items = items;
    }
```

Vous avez dit Test?



```java
class TravelOrganizerTest {

    IService serviceF , serviceC ;
    Trip t;
    TravelOrganizer TO ;

    @BeforeEach
    void setUp() throws Exception {
        ArrayList<Flight> listF = new ArrayList<>();
        listF.add(new Flight("Belfort"));
        listF.add(new Flight("Nice"));
        listF.add(new Flight(100, LocalDate.of(2017, 12, 24), LocalTime.of(7, 45),"Nice", "Calvi"));
        listF.add(new Flight(150, LocalDate.of(2017, 12, 24), LocalTime.of(9, 30), "Nice", "Paris"));
        listF.add(new Flight(150, LocalDate.of(2017, 12, 24), LocalTime.of(18, 30), "Paris", "Nice"));
        serviceF = new FlightService(listF);
        TO.addService(serviceF);

        ArrayList<Car> listC = new ArrayList<>();
        listC.add(new Car("1111 AB 06",50));
        listC.add(new Car("1111 AB 75",100));
        listC.add(new Car("1111 AB 83",75));
        serviceC = new CarRentalService(listC);
        TO.addService(serviceC);

        t = TO.createATrip(new Description(LocalDate.of(2017, 12, 24),
                                    "Nice",
                                    "Calvi",
                                    15));

    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

Vous avez dit Test?



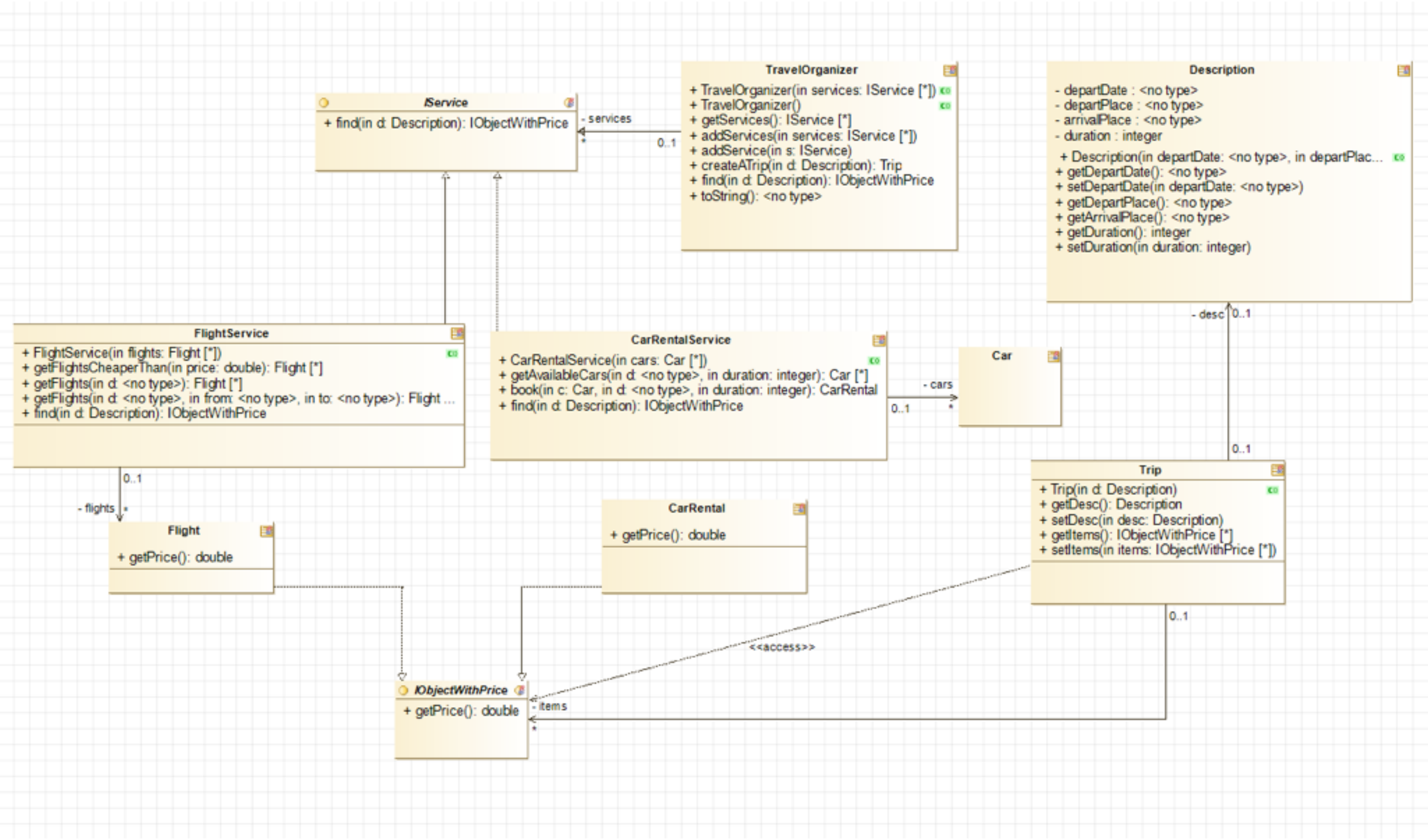
```
class TravelOrganizerTest {

    IService serviceF , serviceC ;
    Trip t;
    TravelOrganizer TO ;

    @BeforeEach
    void setUp() throws Exception {
        ArrayList<Flight> listF = new ArrayList<>();
        listF.add(new Flight("Belfort"));
        listF.add(new Flight("Nice"));
        listF.add(new Flight(100, LocalDate.of(2017, 12, 24), LocalTime.of(7, 45),"Nice", "Calvi"));
        listF.add(new Flight(150, LocalDate.of(2017, 12, 24), LocalTime.of(9, 30), "Nice", "Paris"));
        listF.add(new Flight(150, LocalDate.of(2017, 12, 24), LocalTime.of(18, 30), "Paris", "Nice"));
        serviceF = new FlightService(listF);
        TO.addService(serviceF);

        ArrayList<Car> listC = new ArrayList<>();
        listC.add(new Car("1111 AB 06",50));
        listC.add(new Car("1111 AB 75",100));
        listC.add(new Car("1111 AB 83",75));
        serviceC = new CarRentalService(listC);
        TO.addService(serviceC);

        t = TO.createATrip(new Description(LocalDate.of(2017, 12, 24),
                                    "Nice",
                                    "Calvi",
                                    15));

    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    void test() {
        fail("Not yet implemented");
    }

}
```

Duplicated Code…

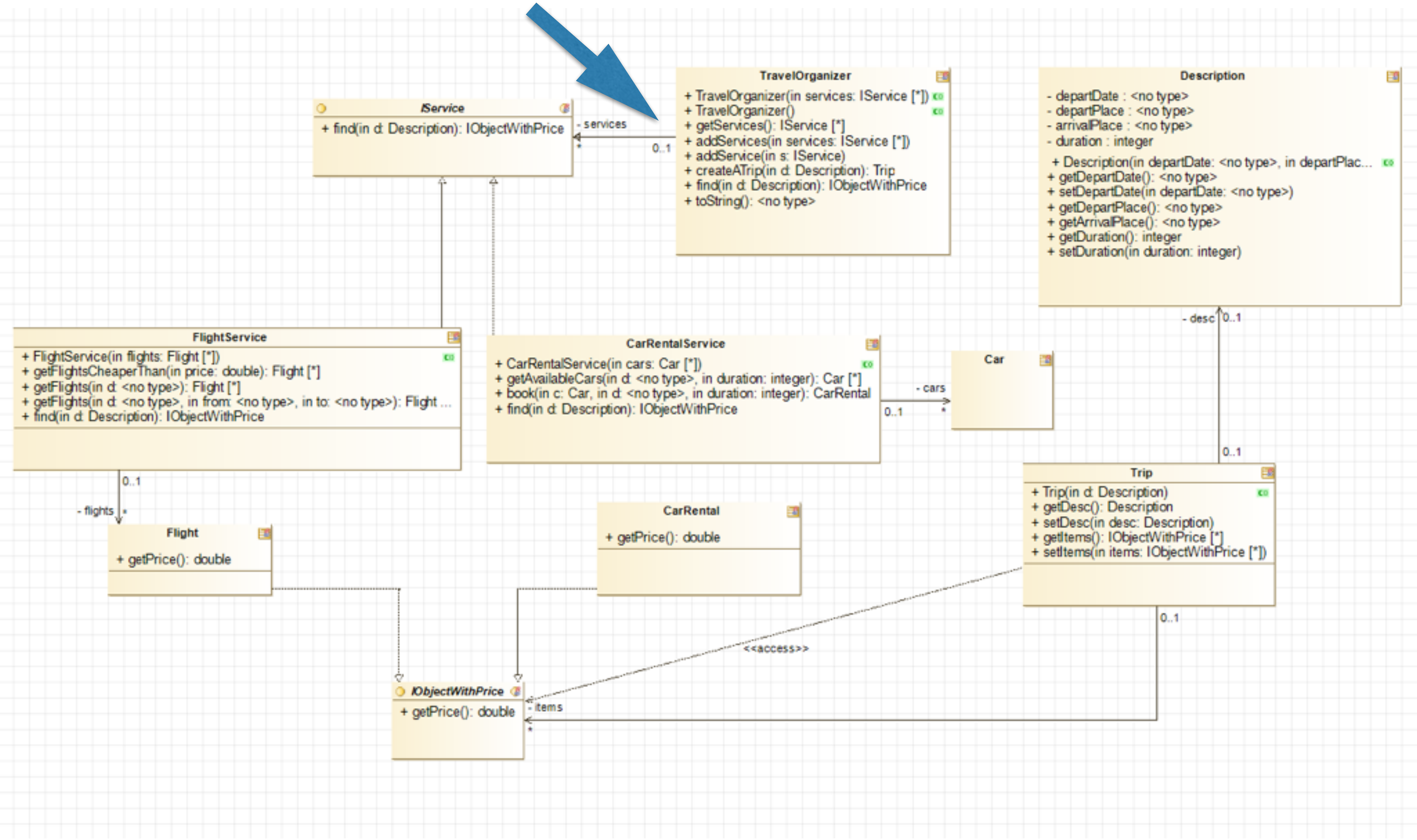
```
public abstract class Vehicule implements Machine{
   protected String immatriculation;
   protected int poidsVide;
   protected int charge;
   protected int horsePower = 15;

   public int getHorsePower() {
    return horsePower;
   }

   public abstract int calculVitesse();
   public abstract int chargeMax();
   public abstract int vitesseMaxMin();
   public int getCharge() {
    return charge;
   }

   public void setCharge(int charge) throws Exception {
    if ( charge > chargeMax() || charge < 0) throw new
Exception("La charge doit tre comprise entre 0 et " +
chargeMax());
    this.charge = charge;
   }
   @Override
   public String toString() {
    return "Vehicule [immatriculation=" +
immatriculation + ", poidsVide=" + poidsVide + ",
charge=" + charge
            + ", Vitesse=" + calculVitesse() + ",
chargeMax()=" + chargeMax() + "]";
   }

   public int getPoids(){
    return poidsVide + charge;
   }


}
```

```
public class VoitureSansPermis extends Vehicule {
   public VoitureSansPermis(String s){
    poidsVide = 1;
    charge = 0;
    immatriculation = s;
   }
   public int chargeMax(){
    return 0;
   }

   public int calculVitesse(){
    return  50;
   }

   @Override
   public int vitesseMaxMin() {
    return 50;
   }


public class CamionCiterne  extends Vehicule {

   public CamionCiterne(String s){
    poidsVide = 3;
    charge=0;
    immatriculation = s ;
   }
   public CamionCiterne(String s, int charge)
throws Exception{
    if ( charge > chargeMax() || charge < 0) throw
new Exception("La charge doit Ítre comprise entre 0
et " + chargeMax());
```

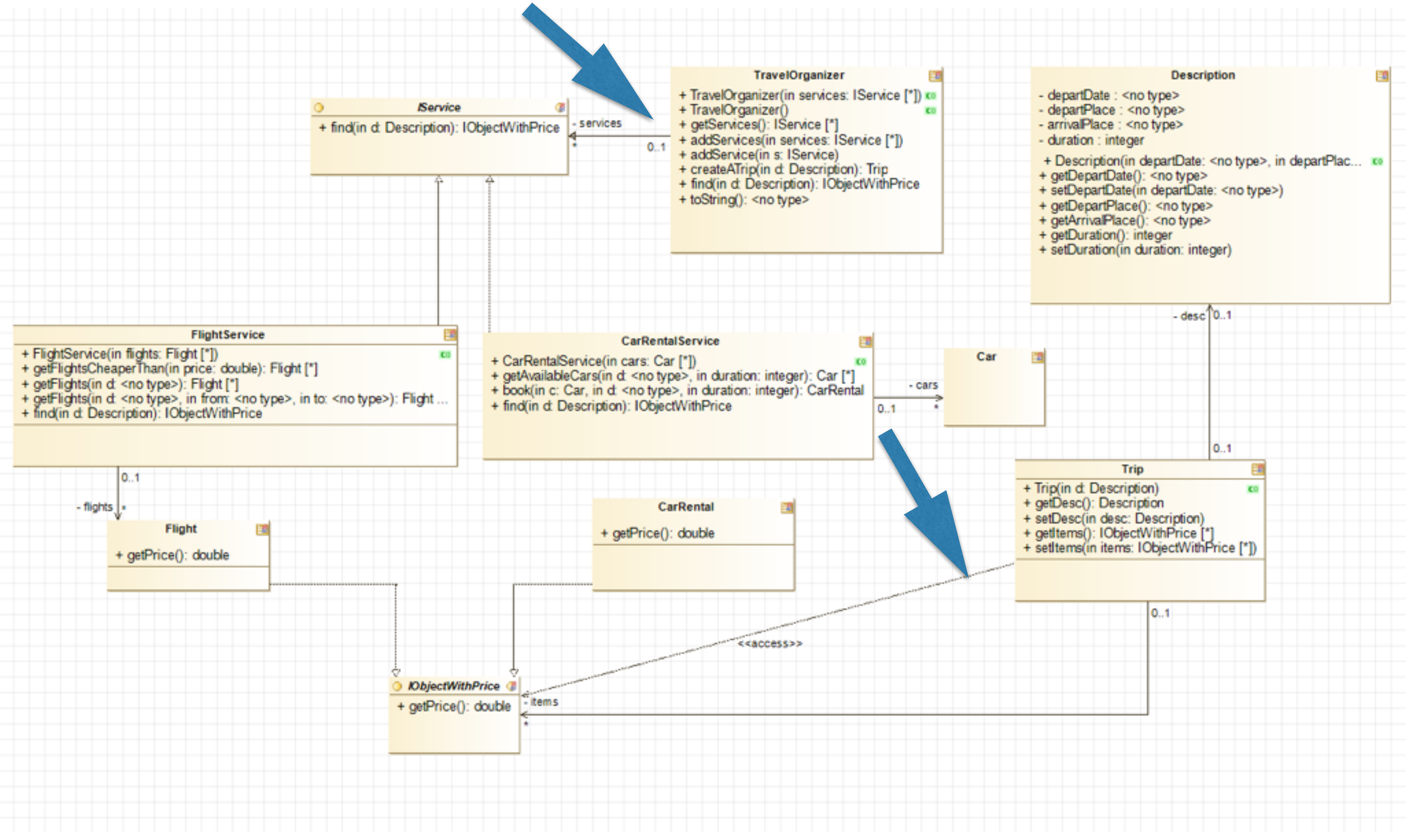
Implémenter ce modèle..


IService
+ find(in d: Description): IObjectWithPrice
- services
0..1
*
TravelOrganizer
+ TravelOrganizer(in services: IService [*])
+ TravelOrganizer()
+ getServices(): IService [*]
+ addServices(in services: IService [*])
+ addService(in s: IService)
+ createATrip(in d: Description): Trip
+ find(in d: Description): IObjectWithPrice
+ toString(): <no type>
Description
- departDate : <no type>
- departPlace : <no type>
- arrivalPlace : <no type>
- duration : integer
+ Description(in departDate: <no type>, in departPlac...
+ getDepartDate(): <no type>
+ setDepartDate(in departDate: <no type>)
+ getDepartPlace(): <no type>
+ getArrivalPlace(): <no type>
+ getDuration(): integer
+ setDuration(in duration: integer)
- desc
0..1
FlightService
+ FlightService(in flights: Flight [*])
+ getFlightsCheaperThan(in price: double): Flight [*]
+ getFlights(in d: <no type>): Flight [*]
+ getFlights(in d: <no type>, in from: <no type>, in to: <no type>): Flight ...
+ find(in d: Description): IObjectWithPrice
CarRentalService
+ CarRentalService(in cars: Car [*])
+ getAvailableCars(in d: <no type>, in duration: integer): Car [*]
+ book(in c: Car, in d: <no type>, in duration: integer): CarRental
+ find(in d: Description): IObjectWithPrice
- cars
0..1
*
Car
0..1
Trip
+ Trip(in d: Description)
+ getDesc(): Description
+ setDesc(in desc: Description)
+ getItems(): IObjectWithPrice [*]
+ setItems(in items: IObjectWithPrice [*])
0..1
- flights
*
Flight
+ getPrice(): double
CarRental
+ getPrice(): double
<<access>>
0..1
IObjectWithPrice
+ getPrice(): double
- items
*

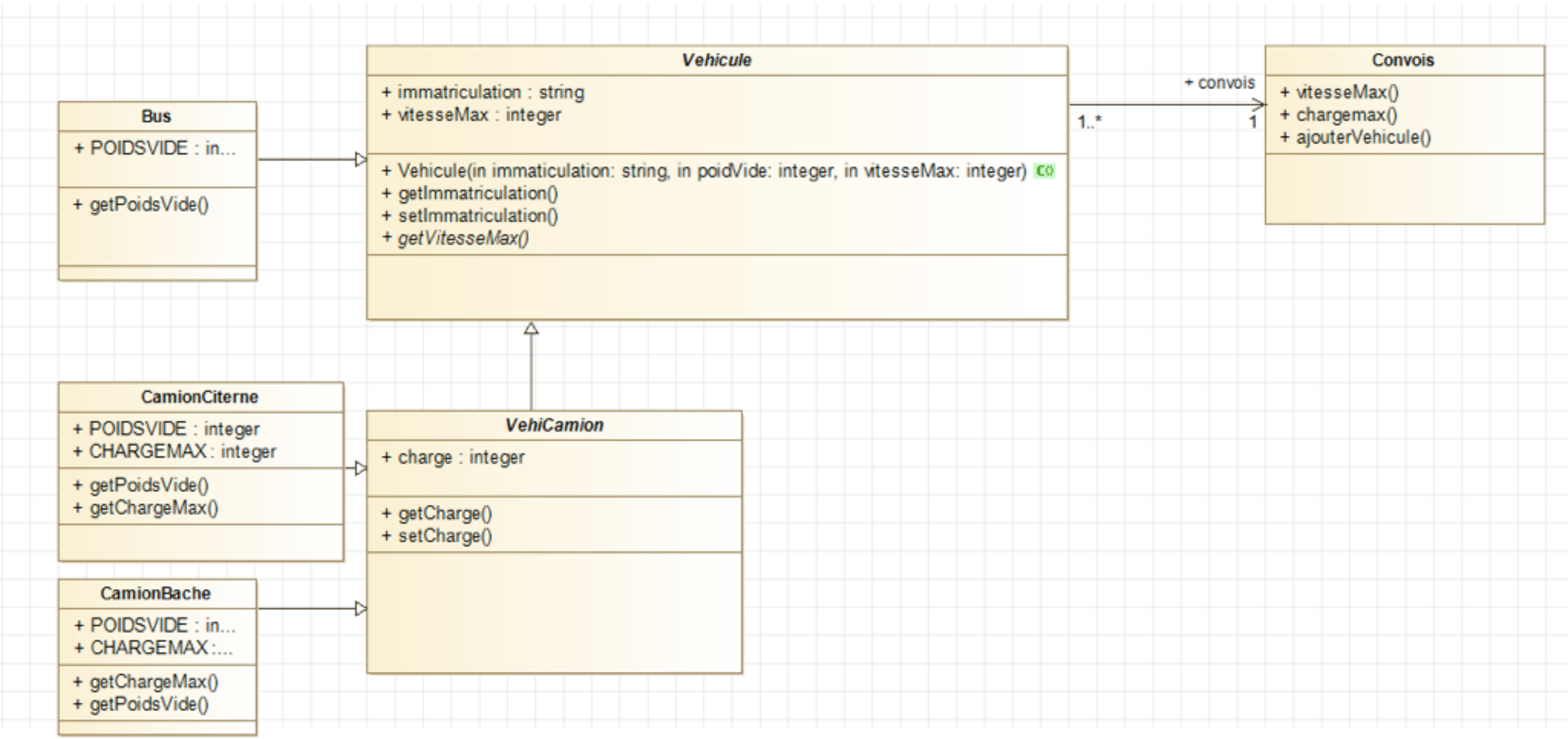
Implémenter ce modèle..


IService
+ find(in d: Description): IObjectWithPrice
- services
0..1
*
TravelOrganizer
+ TravelOrganizer(in services: IService [*])
+ TravelOrganizer()
+ getServices(): IService [*]
+ addServices(in services: IService [*])
+ addService(in s: IService)
+ createATrip(in d: Description): Trip
+ find(in d: Description): IObjectWithPrice
+ toString(): <no type>
Description
- departDate : <no type>
- departPlace : <no type>
- arrivalPlace : <no type>
- duration : integer
+ Description(in departDate: <no type>, in departPlac...
+ getDepartDate(): <no type>
+ setDepartDate(in departDate: <no type>)
+ getDepartPlace(): <no type>
+ getArrivalPlace(): <no type>
+ getDuration(): integer
+ setDuration(in duration: integer)
- desc
0..1
FlightService
+ FlightService(in flights: Flight [*])
+ getFlightsCheaperThan(in price: double): Flight [*]
+ getFlights(in d: <no type>): Flight [*]
+ getFlights(in d: <no type>, in from: <no type>, in to: <no type>): Flight ...
+ find(in d: Description): IObjectWithPrice
CarRentalService
+ CarRentalService(in cars: Car [*])
+ getAvailableCars(in d: <no type>, in duration: integer): Car [*]
+ book(in c: Car, in d: <no type>, in duration: integer): CarRental
+ find(in d: Description): IObjectWithPrice
- cars
0..1
*
Car
0..1
Trip
+ Trip(in d: Description)
+ getDesc(): Description
+ setDesc(in desc: Description)
+ getItems(): IObjectWithPrice [*]
+ setItems(in items: IObjectWithPrice [*])
0..1
- flights
*
Flight
+ getPrice(): double
CarRental
+ getPrice(): double
0..1
<<access>>
IObjectWithPrice
+ getPrice(): double
- items
*

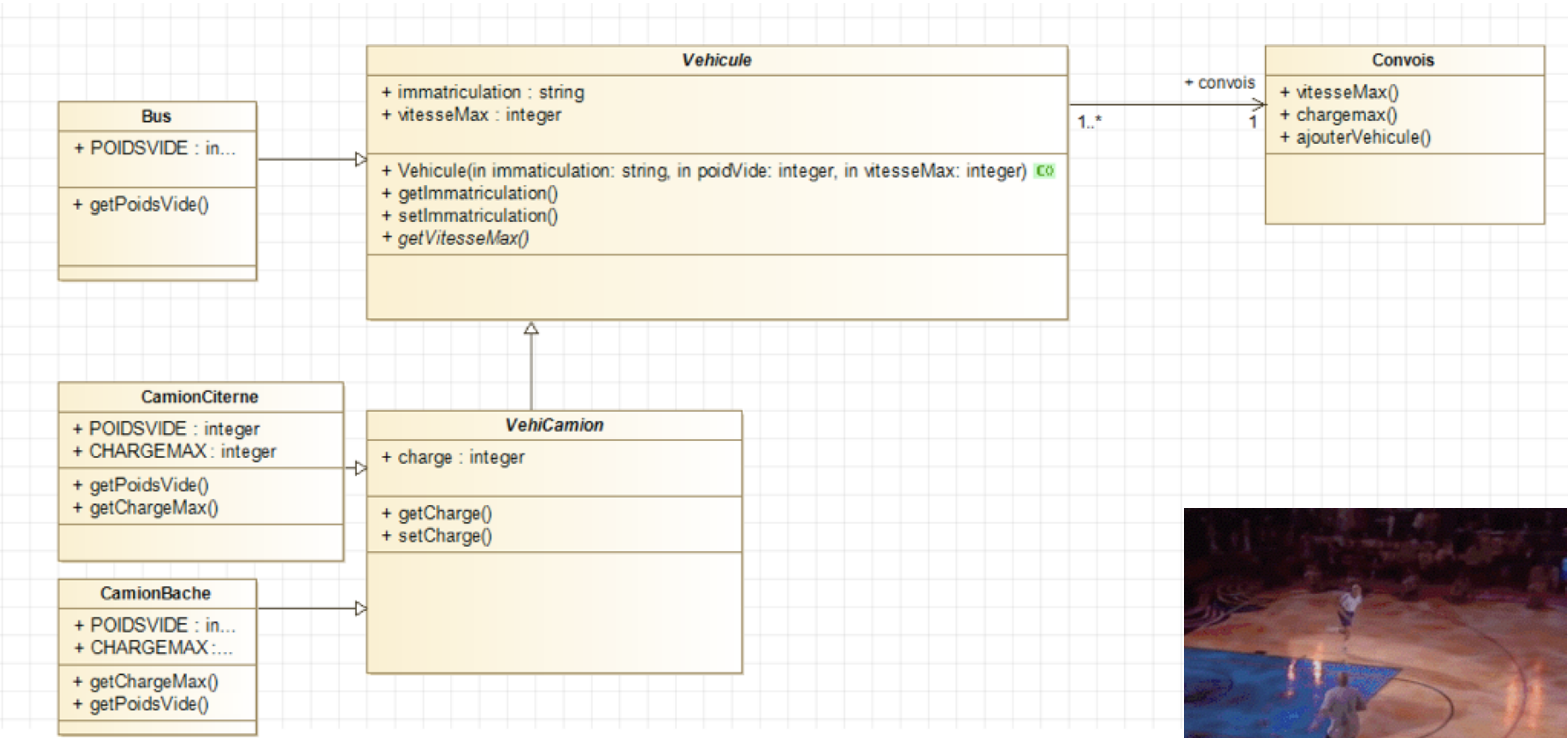
Implémenter ce modèle..


TravelOrganizer
+ TravelOrganizer(in services: IService [*])
+ TravelOrganizer()
+ getServices(): IService [*]
+ addServices(in services: IService [*])
+ addService(in s: IService)
+ createATrip(in d: Description): Trip
+ find(in d: Description): IObjectWithPrice
+ toString(): <no type>
Description
- departDate : <no type>
- departPlace : <no type>
- arrivalPlace : <no type>
- duration : integer
+ Description(in departDate: <no type>, in departPlac...
+ getDepartDate(): <no type>
+ setDepartDate(in departDate: <no type>)
+ getDepartPlace(): <no type>
+ getArrivalPlace(): <no type>
+ getDuration(): integer
+ setDuration(in duration: integer)
IService
+ find(in d: Description): IObjectWithPrice
- services
0..1
*
FlightService
+ FlightService(in flights: Flight [*])
+ getFlightsCheaperThan(in price: double): Flight [*]
+ getFlights(in d: <no type>): Flight [*]
+ getFlights(in d: <no type>, in from: <no type>, in to: <no type>): Flight ...
+ find(in d: Description): IObjectWithPrice
CarRentalService
+ CarRentalService(in cars: Car [*])
+ getAvailableCars(in d: <no type>, in duration: integer): Car [*]
+ book(in c: Car, in d: <no type>, in duration: integer): CarRental
+ find(in d: Description): IObjectWithPrice
Car
- cars
0..1
*
- desc
0..1
0..1
Trip
+ Trip(in d: Description)
+ getDesc(): Description
+ setDesc(in desc: Description)
+ getItems(): IObjectWithPrice [*]
+ setItems(in items: IObjectWithPrice [*])
0..1
- flights
*
Flight
+ getPrice(): double
CarRental
+ getPrice(): double
<<access>>
0..1
IObjectWithPrice
+ getPrice(): double
- items
*

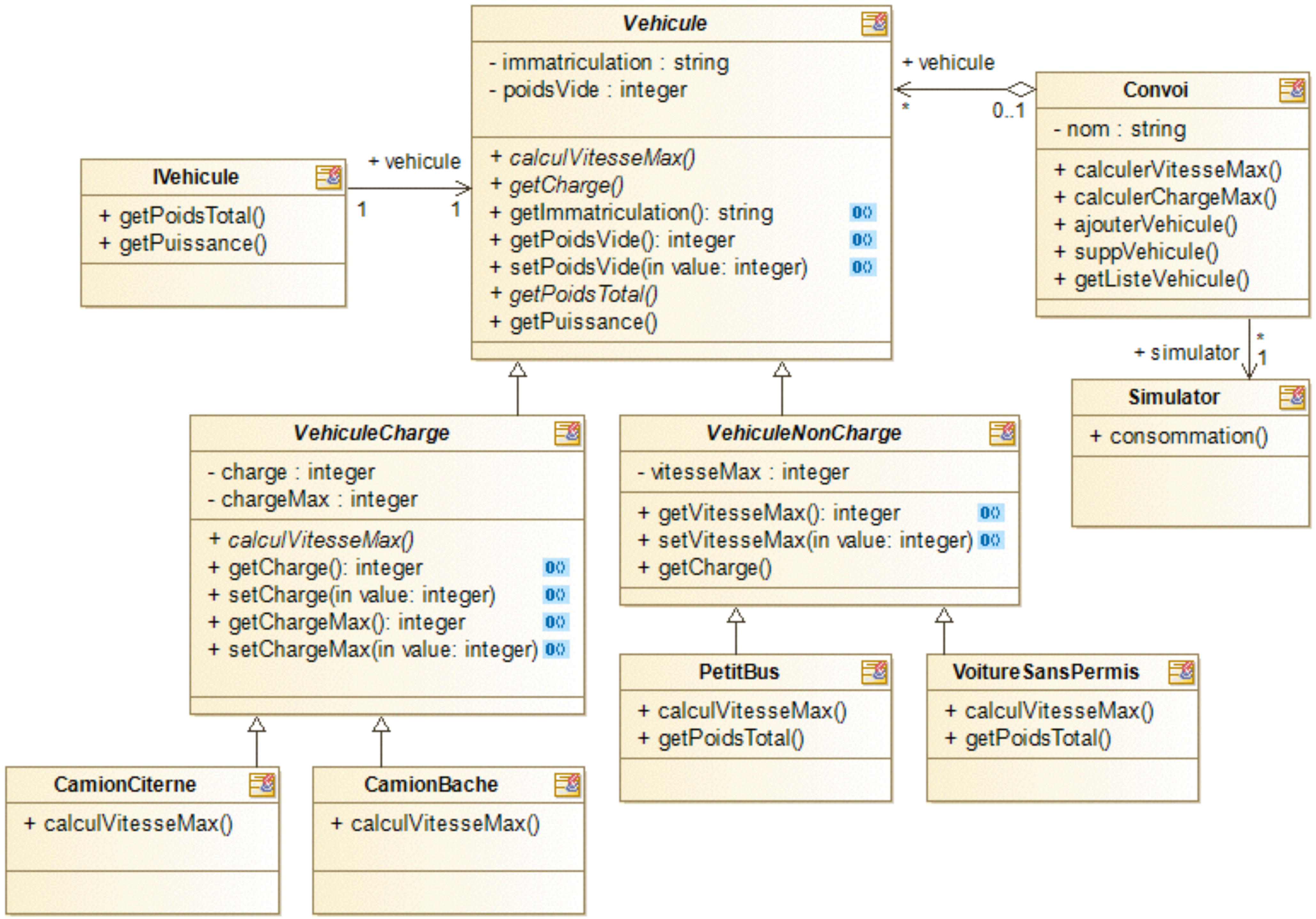
Implémenter « Convois »



Implémenter « Convois »



Que devez-vous implémenter pour (I)Vehicule?

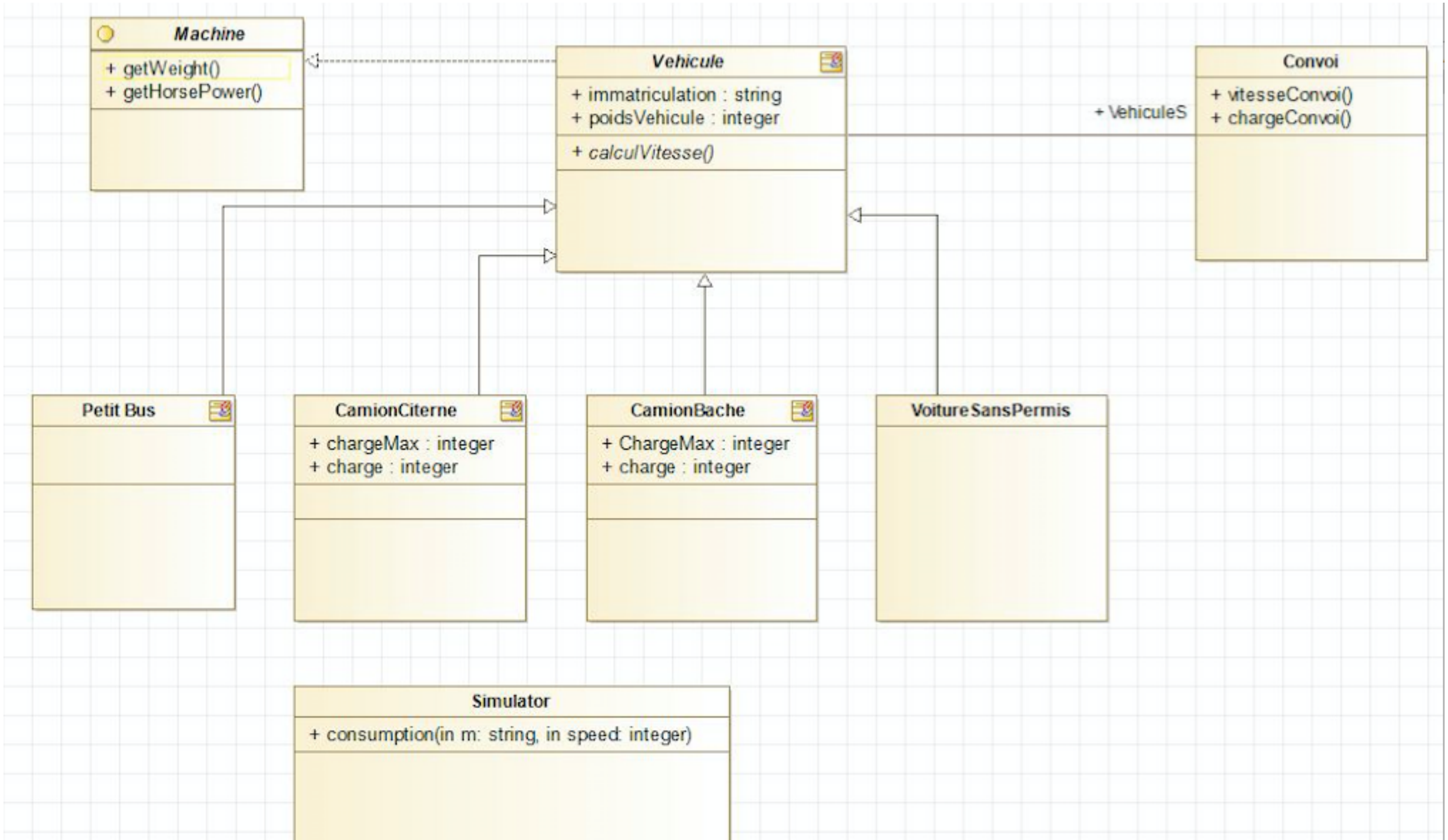


Correspondance ?

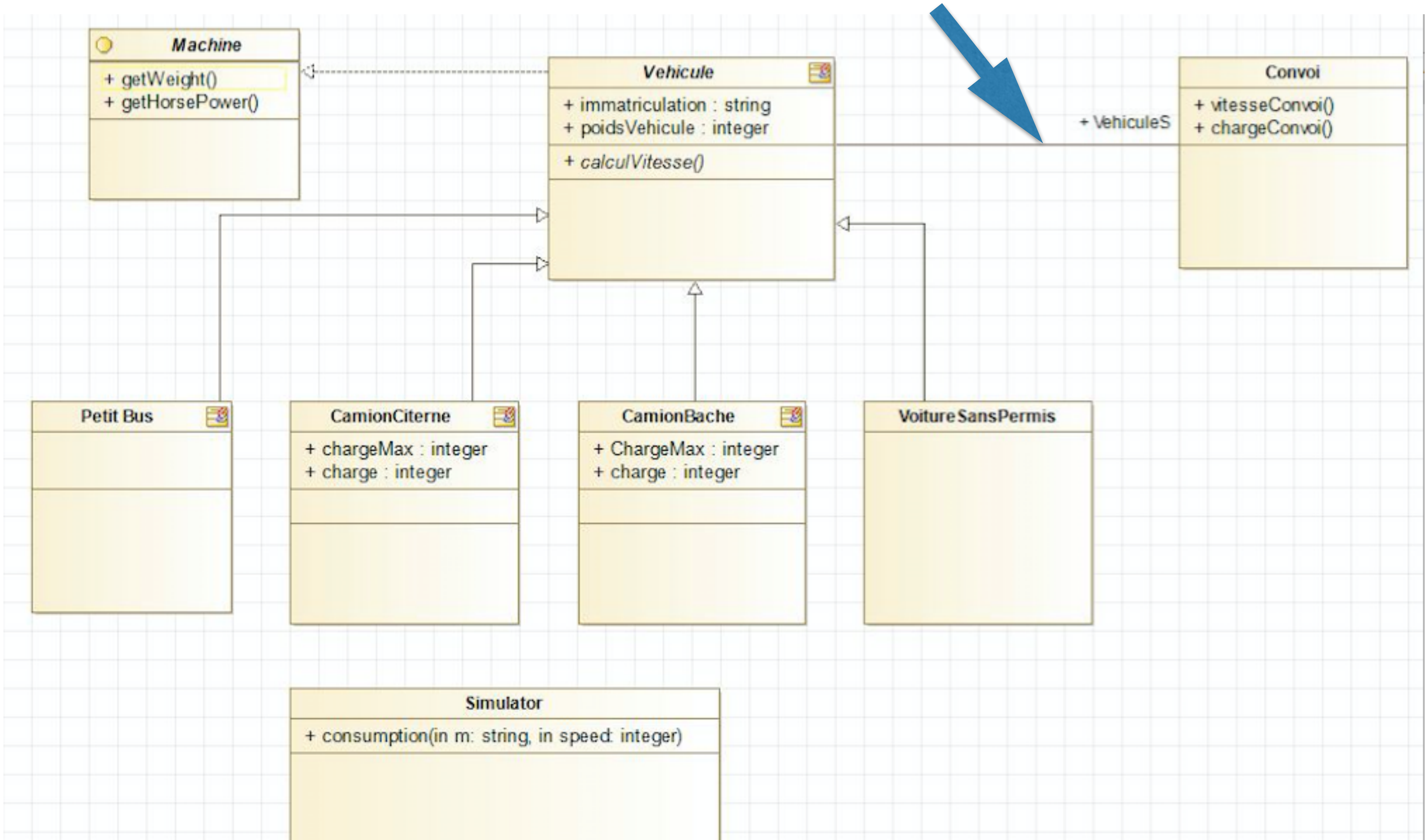
```java
public interface IVehicule {

   int getPoidsTotal();
   int getPuissance();
}
```

```java
Public abstract class  Vehicule implements IVehicule{
   private String immatriculation;
   private int poidsVide;

   public Vehicule(String imm) {
    this.immatriculation = imm;

   }

   public abstract int calculVitesseMax();
   public abstract int getCharge();
   public abstract int getPoidsTotal();

   public int getPuissance() {
    return 100;
   }

   public String getImmatriculation()
   {
    return this.immatriculation;
   }

   public int getPoidsVide()
   {
    return this.poidsVide;
   }

   public void setPoidsVide(int poids)
   {
    this.poidsVide = poids;
   }


}
```

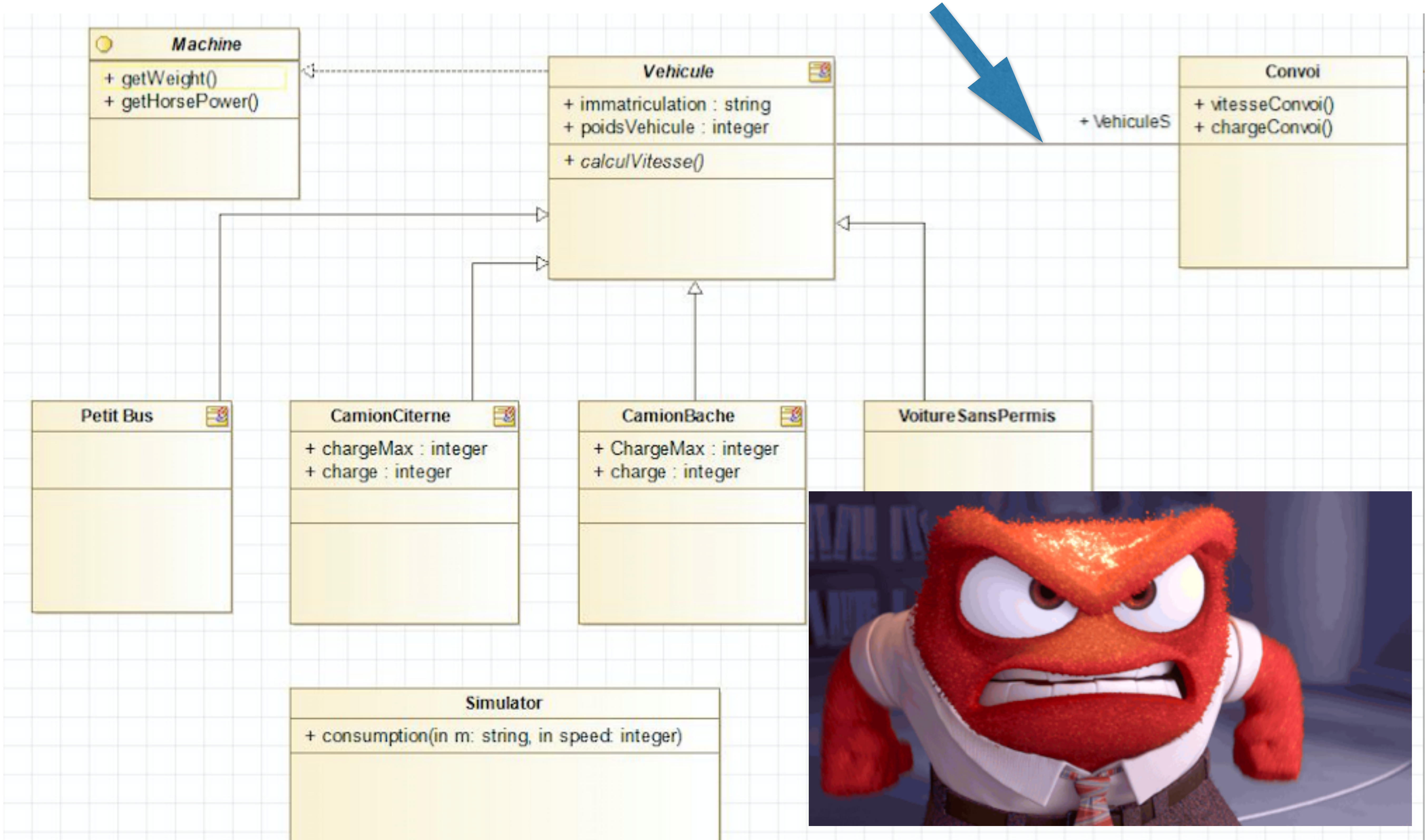
Ok, on implémente

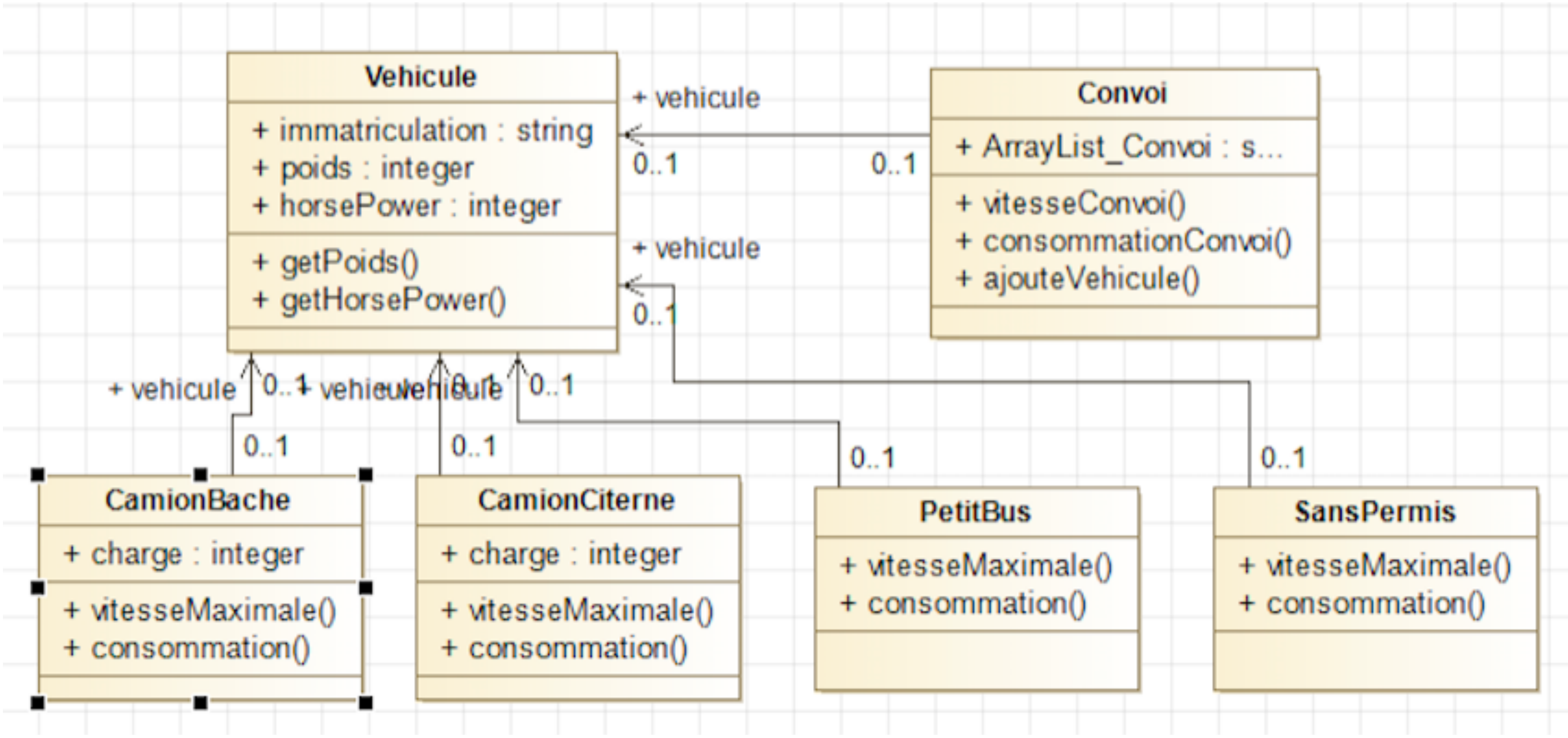


Ok, on implémente

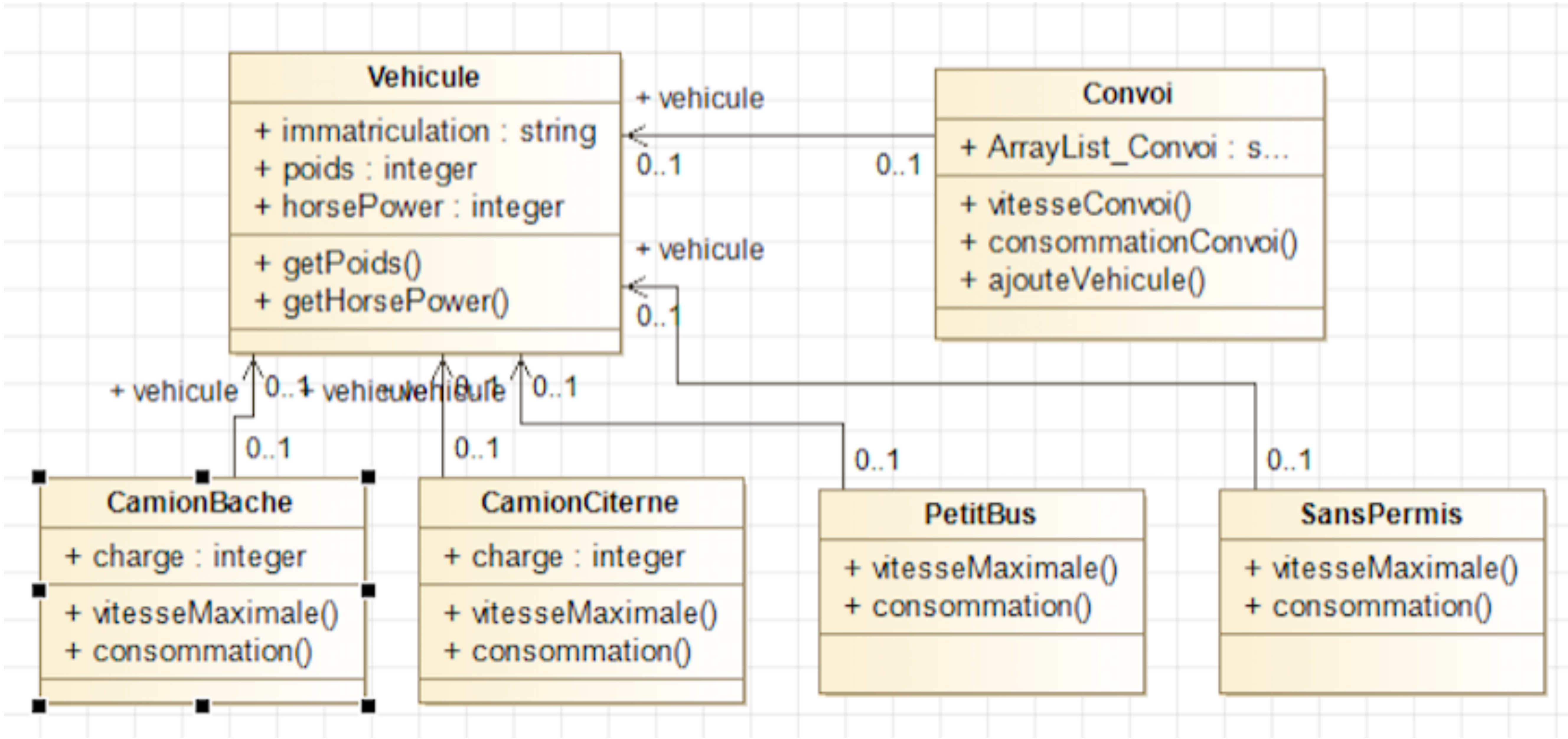


Ok, on implémente

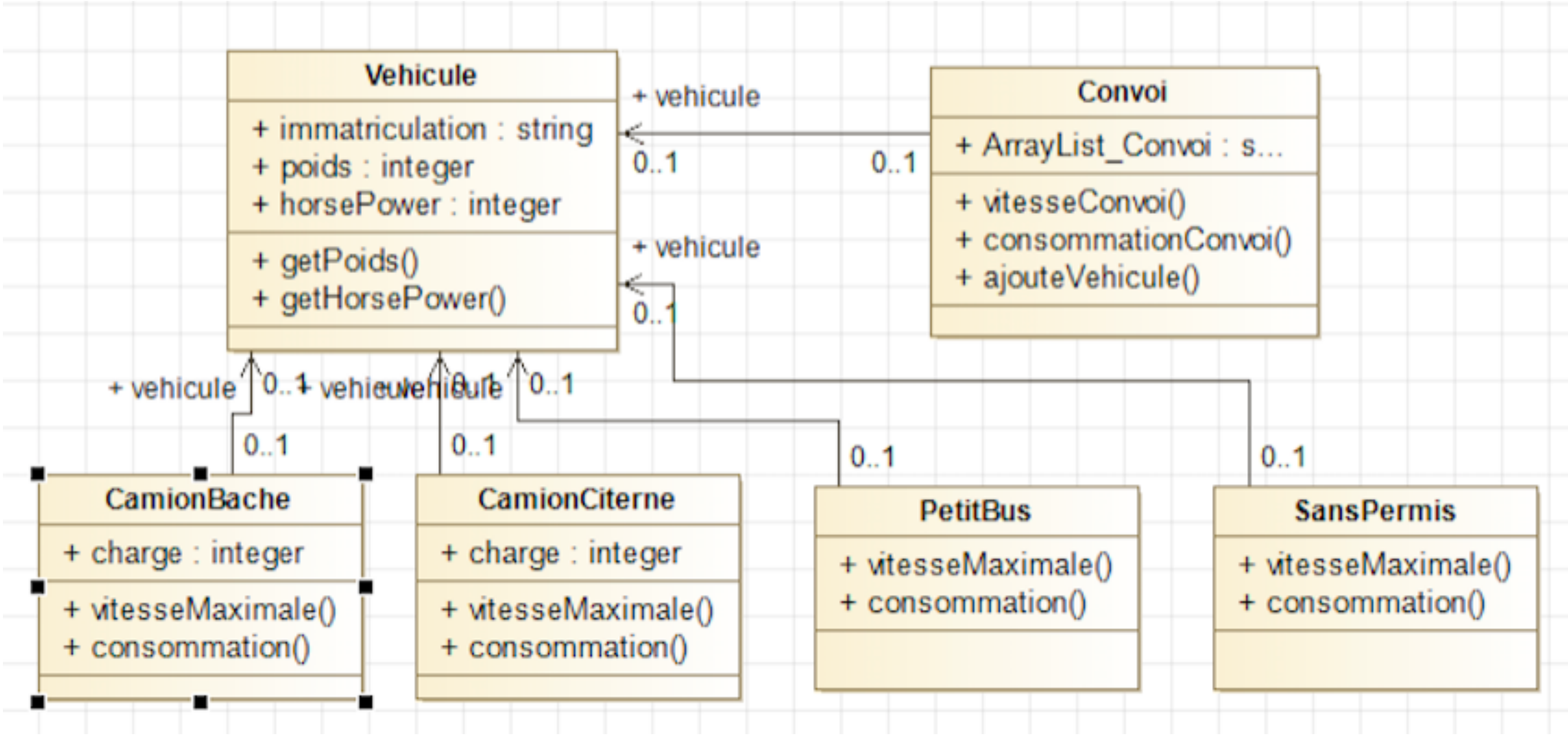




Vehicule
+ immatriculation : string
+ poids : integer
+ horsePower : integer
+ getPoids()
+ getHorsePower()
Convoi
+ ArrayList_Convoi : s...
+ vitesseConvoi()
+ consommationConvoi()
+ ajouteVehicule()
+ vehicule
0..1
CamionBache
+ charge : integer
+ vitesseMaximale()
+ consommation()
CamionCiterne
+ charge : integer
+ vitesseMaximale()
+ consommation()
PetitBus
+ vitesseMaximale()
+ consommation()
SansPermis
+ vitesseMaximale()
+ consommation()

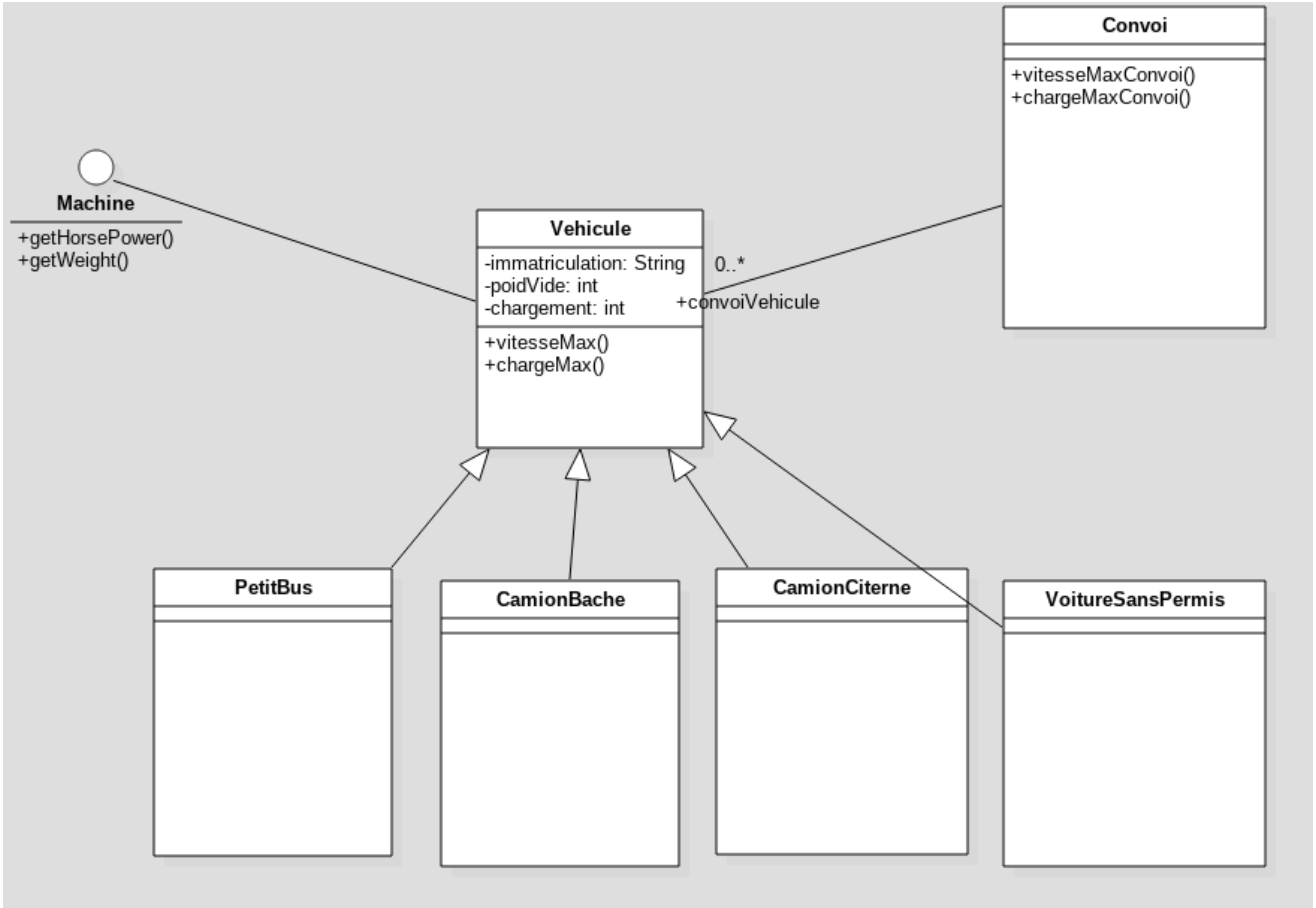


```
public class CamionBache **extends** Vehicules {
     private int charge;
     public CamionBache(String _immatriculation,int _charge,
int _horsePower){
     super(_immatriculation,5,_horsePower);
     charge = _charge;
```

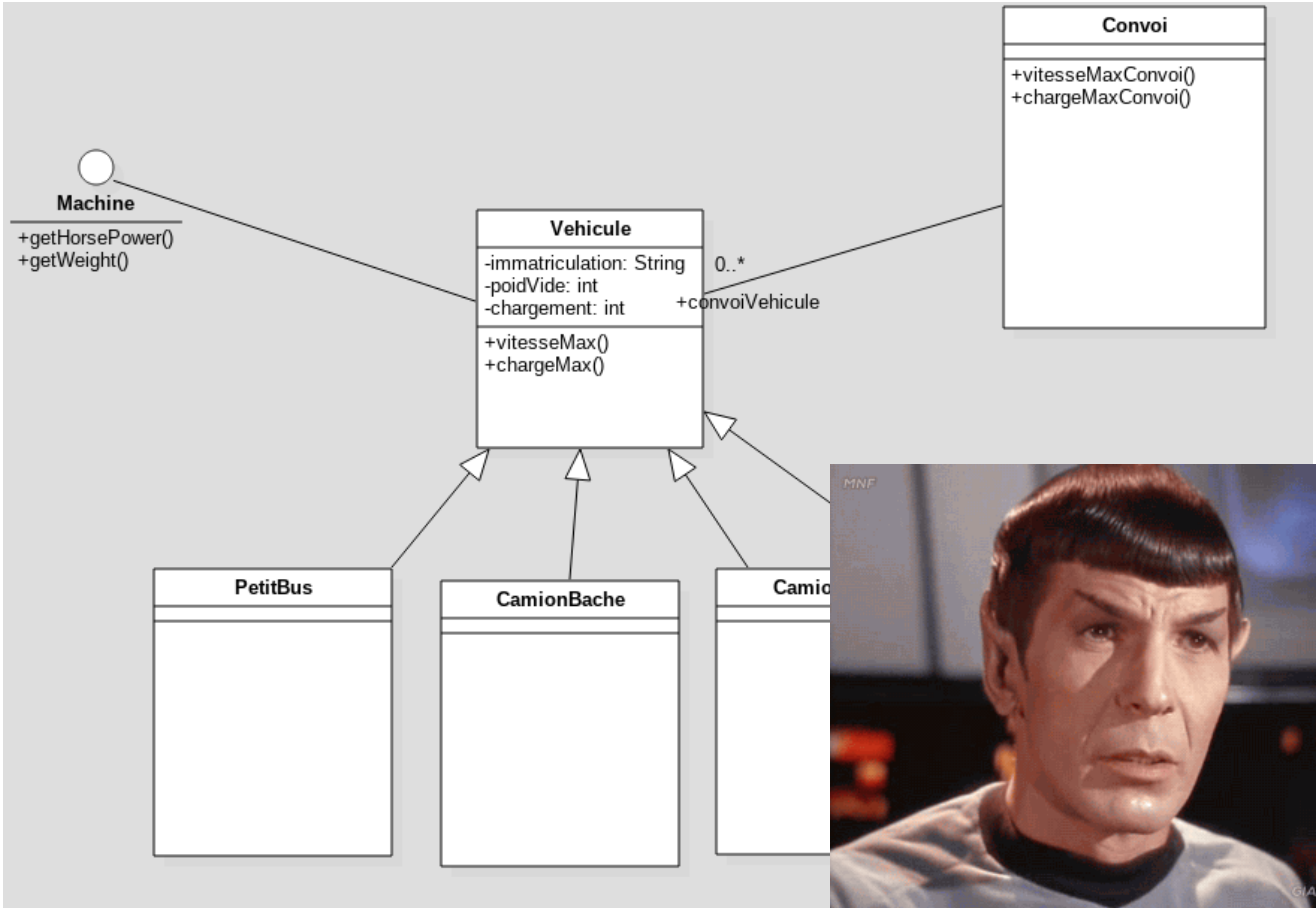


```
public class CamionBache extends Vehicules {
     private int charge;
     public CamionBache(String _imma
int _horsePower){
     super(_immatriculation,5,_horsePo
     charge = _charge;
```

Quelle relation entre Vehicule et convoi?



Quelle relation entre Vehicule et convoi?



Et si on parlait de complexité/optimisation

```java
public int calculerVitesseMax() {
     Vehicules vehiculeActuel = vehicules.get(0);
     for(int i=1; i<vehicules.size(); i++) {
         if(vehicules.get(i).vitesseMax() < vehiculeActuel.vitesseMax()) {
             vehiculeActuel = vehicules.get(i);
         }
     }
     return vehiculeActuel.vitesseMax();
}
```

Et si on parlait de complexité/optimisation

```
public int calculerVitesseMax() {
    Vehicules vehiculeActuel = vehicules.get(0);
    for(int i=1; i<vehicules.size(); i++) {
        if(vehicules.get(i).vitesseMax() < vehiculeActuel.vitesseMax()) {
            vehiculeActuel = vehicules.get(i);
        }
    }
    return vehiculeActuel.vitesseMax();
}
```

Combien de fois la méthode vitesseMax() est-elle invoquée ? est-ce indispensable ? Cohérent ?

Et si on parlait de complexité/optimisation

```
public int calculerVitesseMax() {
    Vehicules vehiculeActuel = vehicules.get(0);
    for(int i=1; i<vehicules.size(); i++) {
        if(vehicules.get(i).vitesseMax() < vehiculeActuel.vitesseMax()) {
            vehiculeActuel = vehicules.get(i);
        }
    }
    return vehiculeActuel.vitesseMax();
}
```

Combien de fois la méthode vitesseMax() est-elle invoquée ? est-ce indispensable ? Cohérent ?

```
package vehicules;
public abstract class VehiculeTransport extends Vehicule {
[redacted]

  public void set_Charge(int c) throws Throwable{
[redacted]
```

Ok… Try … v.se**t_C**harge.. Catch ?…

```
package vehicules;
public abstract class VehiculeTransport extends Vehicule {
```

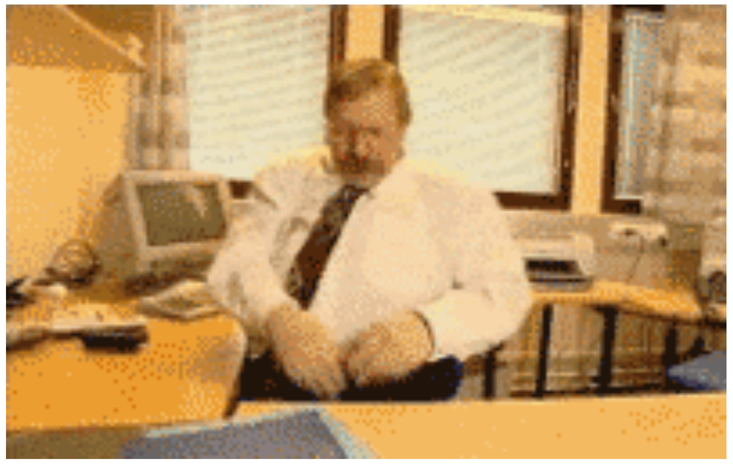
[redacted]

```
  public void set_Charge(int c) throws Throwable{
```

[redacted]

```java
package vehicules;
public abstract class VehiculeTransport extends Vehicule {

  private int charge;
  private int chargeMax;

  public VehiculeTransport(String imm, int p, int c) {
   set_Immatriculation(imm);
   set_PoidsVide(p);
   chargeMax = c;
  }

  public int get_Charge() {
   return charge;
  }

  public void set_Charge(int c) throws Throwable{
   if(c>=0 && c < chargeMax)
      charge = c;
   else throw new Throwable("Charge invalide\n");
  }


}
```

```java
package vehicules;
public abstract class VehiculeTransport extends Vehicule {

  private int charge;
  private int chargeMax;

  public VehiculeTransport(String imm, int p, int c) {
   set_Immatriculation(imm);
   set_PoidsVide(p);
   chargeMax = c;
  }

  public int get_Charge() {
   return charge;
  }

  public void set_Charge(int c) throws Throwable{
   if(c>=0 && c < chargeMax)
      charge = c;
   else throw new Throwable("Charge invalide\n");
  }


}
```



A ne pas faire !

- NE PAS Livrer les exécutables!
- Attentiion aux relations implements et hertage ce n'est pas pareil !!! Apprenez UML!
- N'oubliez pas d'ôter les commentaires "inutiles" tels que ceux générés surtout quand vous avez écrit le code.
- Ne laissez pas de Println …
- METTEZ LES ROLES!!!
- NOMMEZ MIEUX VOS VARIABLES CE SERA PLUS SIMPLE