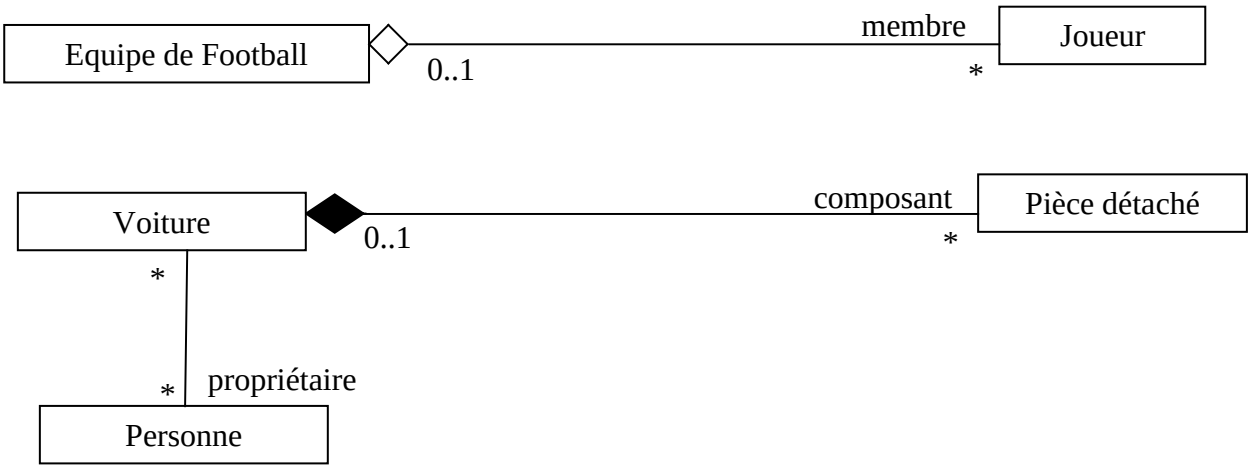
TD6 : Diagramme de classes dans le cadre d'une analyse détaillée (suite)

Diagramme de classes – éléments supplémentaires

Agrégation et composition

1. Indiquez la différence entre les 3 éléments suivants : une association, une association « agrégation », une association « composition ».
2. Précisez le type d'association dans les diagrammes de classes suivants.

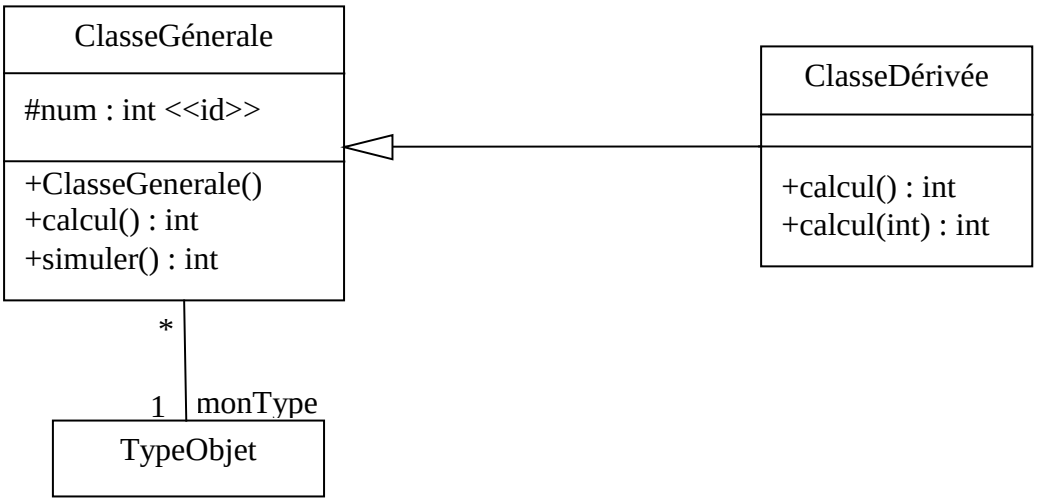


Généralisation/Spécialisation : héritage

La généralisation décrit une relation entre une classe générale (classe de base ou classe parent) et une classe spécialisée (sous-classe, classe dérivée ou classe enfant).
Un objet de la classe spécialisée peut être utilisé partout où un objet de la classe de base est autorisé. Cette relation de généralisation se traduit par le concept d'héritage. On parle également de relation d'héritage.
Les propriétés principales de l'héritage sont :

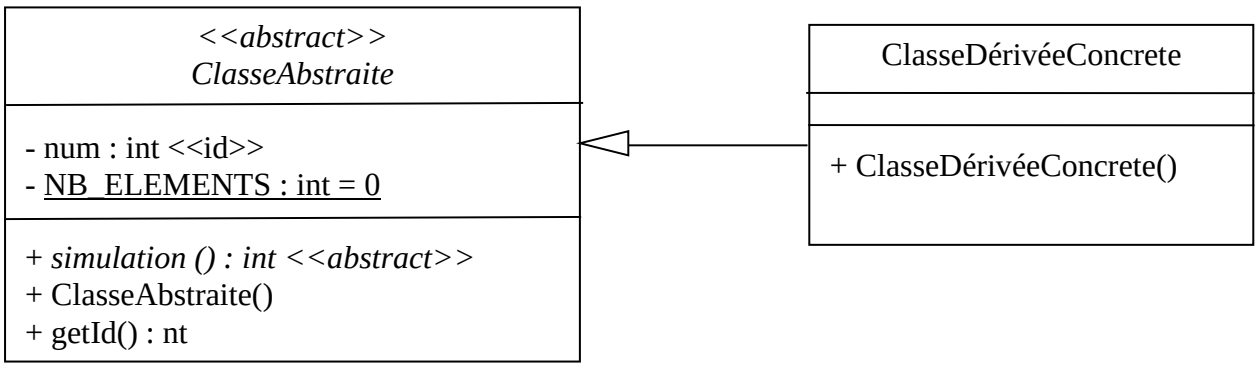
- la classe spécialisée possède toutes les caractéristiques (attributs, associations ou méthodes) de sa classe parent, mais elle ne peut accéder aux caractéristiques privées de cette dernière ;
- toutes les associations de la classe générale s'appliquent aux classes dérivées ;
- un objet d'une classe peut être utilisée partout où une instance de sa classe parent est attendue ;
- *redéfinition* : une classe enfant peut redéfinir une ou plusieurs méthodes de la classe parent (avec la même signature). Sauf indication contraire, un objet utilise les opérations les plus spécialisées dans la hiérarchie des classes.

La *surcharge* d'opérations (même nom, mais signatures des opérations différentes) est possible dans toutes les classes.
Lors de la conception ne pas confondre la notion « d'héritage » avec la notion être « du type ».



Classe abstraite

Une opération est dite *abstraite* lorsqu'on a défini sa signature mais pas son implémentation (c'est à dire « son code »).
Une classe est dite *abstraite* lorsqu'elle a au moins une méthode abstraite.
Une classe abstraite ne peut pas être instanciée.
Elle peut avoir un ou des constructeur(s).



Interface dans la notation UML

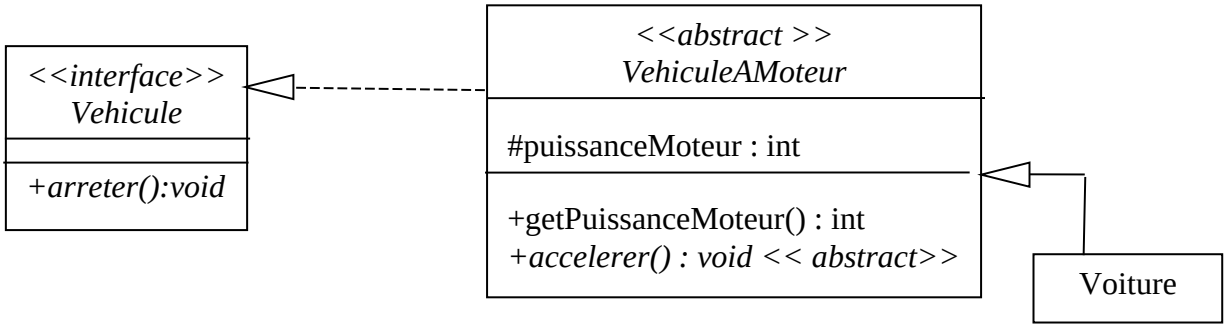
Ref : https://www.ibm.com/support/knowledgecenter/fr/SS5JSH_9.5.0/com.ibm.xtools.modeler.doc/topics/cinterfc.html

Les interfaces masquent et protègent le code en déclarant publiquement une liste de services (méthodes). Les classes réalisent les interfaces en implémentant ce comportement.

Le développement d'applications est simplifié car les développeurs de logiciels qui écrivent le code client ont uniquement besoin de connaître les interfaces, mais pas les détails de l'implémentation.

Si vous remplacez, les classes qui implémentent des interfaces, vous n'avez pas besoin de refaire la conception de l'application client car les nouvelles classes implémentent les mêmes interfaces.

Une interface est une classe contenant uniquement des attributs et méthodes de classes ainsi que des méthodes abstraites.
Une interface ne peut pas être instanciée et n'a pas de constructeur.



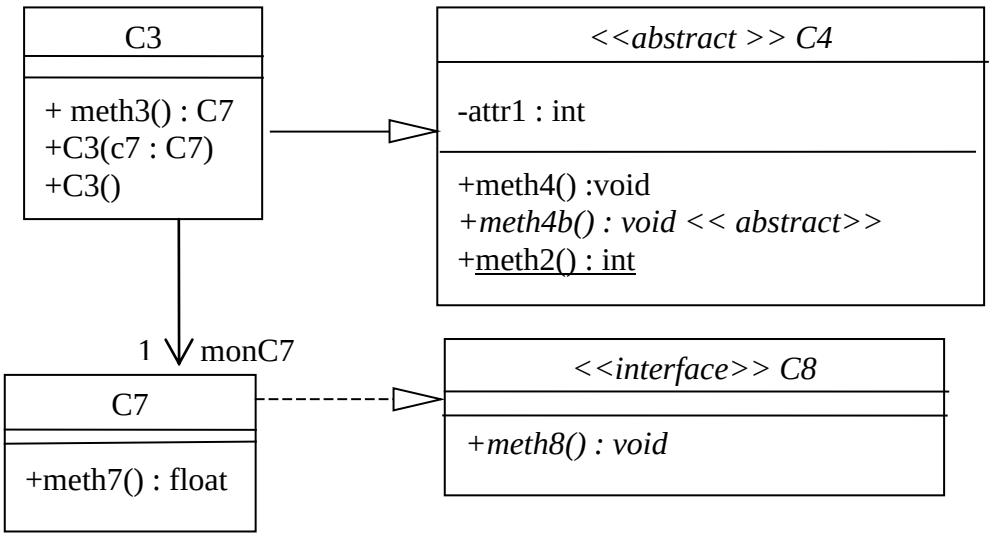

```
public interface Vehicule {
      public void arreter () ;
}

public abstract class VehiculeAMoteur implements Vehicule {
     protected int puissanceMoteur;
     abstract public void accelerer();
     public int getPuissanceMoteur() {
           return puissanceMoteur;
    }
}

public class Voiture extends VehiculeAMoteur {
  public void accelerer() {}
  public void arreter() {}

}
```

Diagramme de Classes C3478



Questions

1. Ecrivez le code Java correspondant au diagramme de classes C3478.
2. Commentez les lignes de code en Java suivantes en fonction du diagramme de Classes C3478. Ce code a été réalisé par un programmeur <u>débutant</u> qui peut avoir fait des erreurs.

```
public class MainC3478 {
  public static void main(String[] args) {
        C8 o8 ;
        o8 = new C8() ;
        o8 = new C7();
        System.out.println(C4.meth2());
        C4 o4 ;
        o4 = new C4() ;
        o4 = new C3() ;
        o4.meth4b();
        o4.meth3();
        C7 o7 ;
        o7 = new C7() ;
        o7.meth8() ;
        System.out.println(o7.meth7());
        C3  o3 = new C3(c7) ;
        o3.meth4b();
        o3.meth4() ;
        C3.meth3() ;
        o3.meth3().meth8() ;
        System.out.println(o3.meth3().meth7());
}}
```