HÉRITAGE

INTRODUCTION À L'APPROCHE ORIENTÉ OBJET

- Notion d'héritage
- Héritage multiple
- Héritage simple
- Implémentation UML
- Exercices

HÉRITAGE

NOTION D'HÉRITAGE

Toujours dans l'optique de la maintenabilité et la réutilisation, l'orienté objet utilise un deuxième concept fondamental : « L'héritage ».

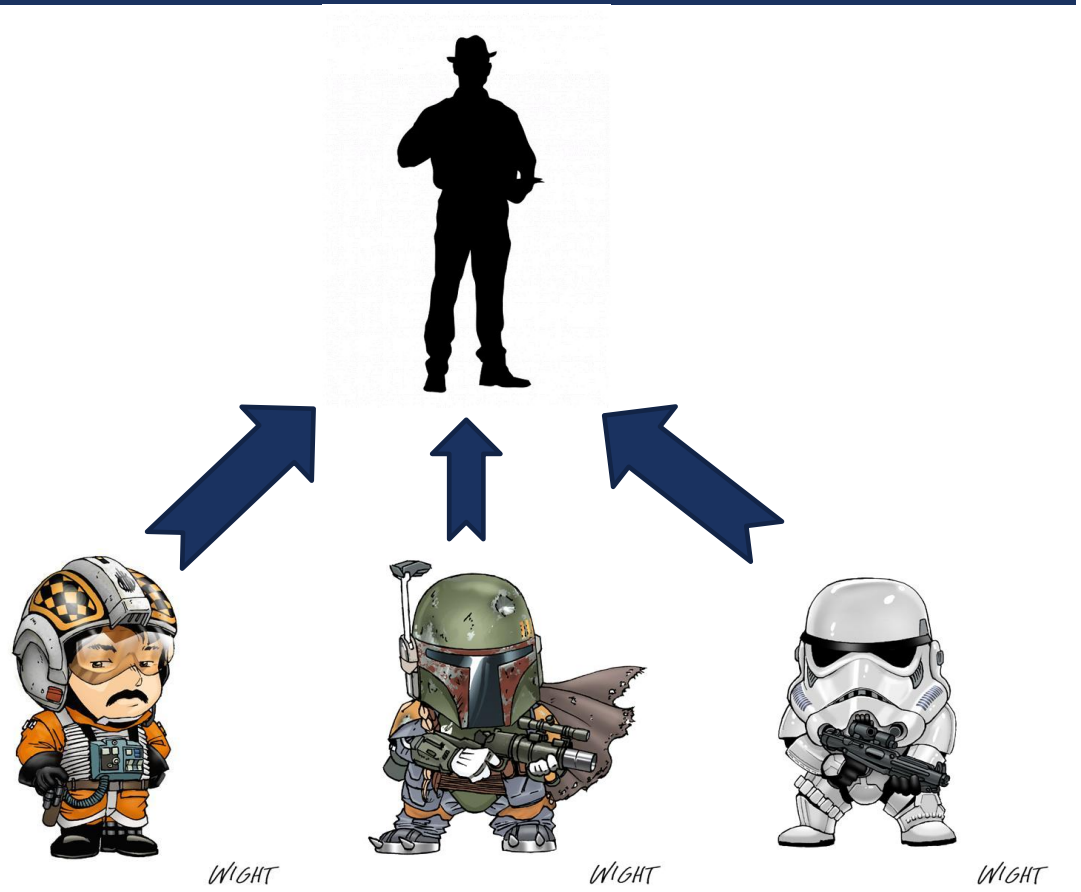
Lorsque plusieurs classes d'une même famille possèdent des fonctionnalités connexes, nous avons tout intérêt à utiliser l'héritage.

Le principe est de déclarer une classe, appelée classe parent, regroupant ces fonctionnalités et de spécifier que nos classes, qui deviennent des classes enfants, héritent de cette « superclasse ».

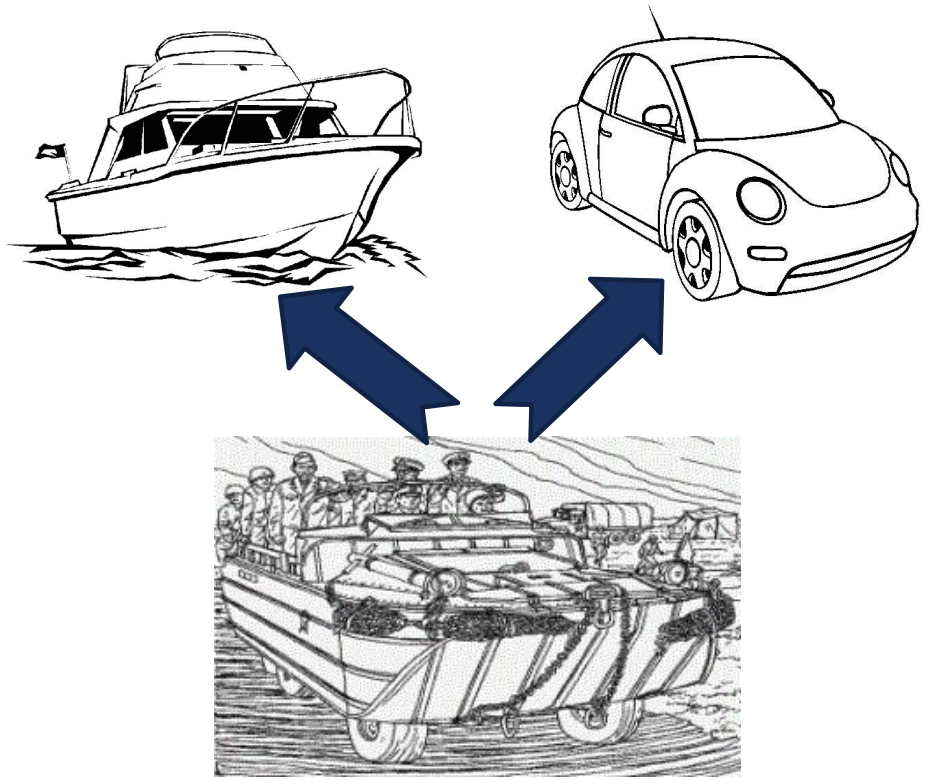
Cependant, il est important de conserver en tête les concepts d'encapsulation.

Par héritage on hérite de tout ce qui vient du parent, excepté les membres privés à celle-ci.

On peut résumer l'héritage par une association « est un(e) »



HÉRITAGE MULTIPLE



Dans le cadre de l'orienté objet, ils nous est permis de faire de l'héritage multiple. C'est-à-dire que nous avons la possibilité **d'hériter de plusieurs classes parents**.

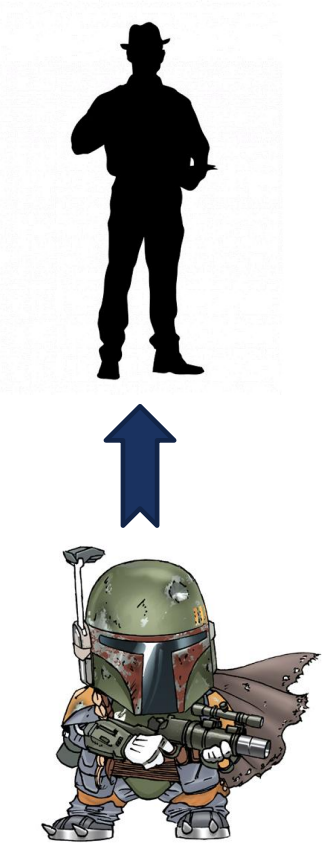
Cette technique permet de regrouper au sein d'une seule et même classe les attributs et méthodes de plusieurs classes parents.

HÉRITAGE SIMPLE

Cependant, développer en tenant compte de l'héritage multiple peut s'avérer rapidement être ardus.

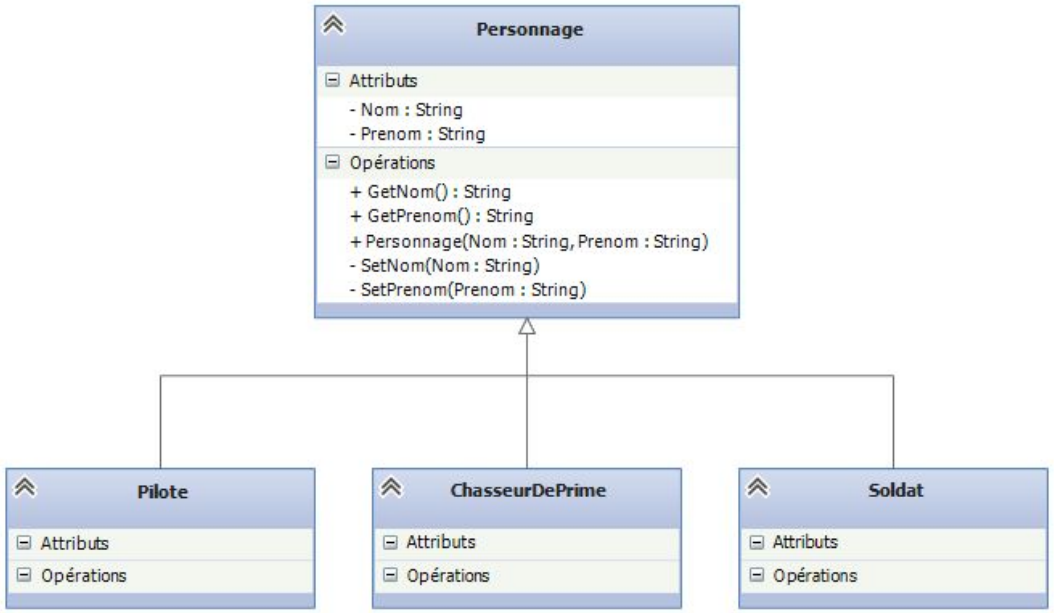
Par conséquent, les concepteurs de certains langages de programmation, comme le C# ou le Java, ont prit la décision de limiter l'héritage à une seule classe à la fois.

On parle donc d'héritage simple.

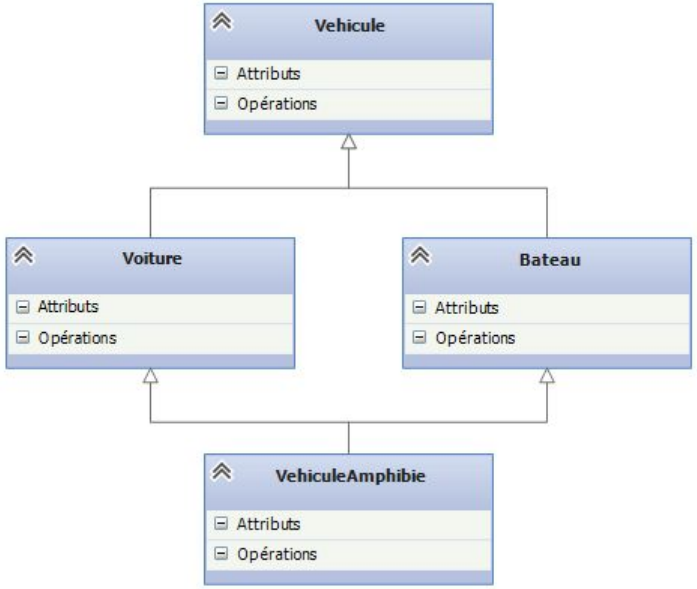


IMPLÉMENTATION UML

Héritage simple



Héritage multiple



EXERCICES

- Déterminer sur papier les attributs, les accesseurs, les méthodes, le(s) constructeur(s) et les modificateurs d'accès pour chacun d'entre eux afin de représenter un compte bancaire type livret d'épargne.
- Déterminer sur papier, sur base de l'exercice lié à l'encapsulation, s'il y a lieu de faire de l'héritage et l'implémenter
- Définir les classes à l'aide d'un diagramme de classe

POLYMORPHISME

INTRODUCTION À L'APPROCHE ORIENTÉ OBJET

- Définition
- Le polymorphisme paramétrique
- Le polymorphisme Ad hoc
- Le polymorphisme d'héritage

DÉFINITION

Polymorphisme impactant les membres
Le polymorphisme paramétrique
Le polymorphisme Ad Hoc

Polymorphisme impactant les types
Le polymorphisme d'héritage

Le mot polymorphisme est formé à partir du grec ancien πολύς (polús) qui signifie « nombreux » et μορφή (morphê) qui signifie « forme ».

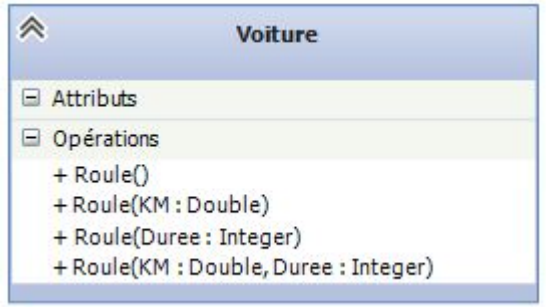
Dans le cadre de l'orienté objet, nous retrouvons trois types de polymorphismes dont deux qui impactent les membres et un qui concerne les types.

POLYMORPHISME PARAMÉTRIQUE

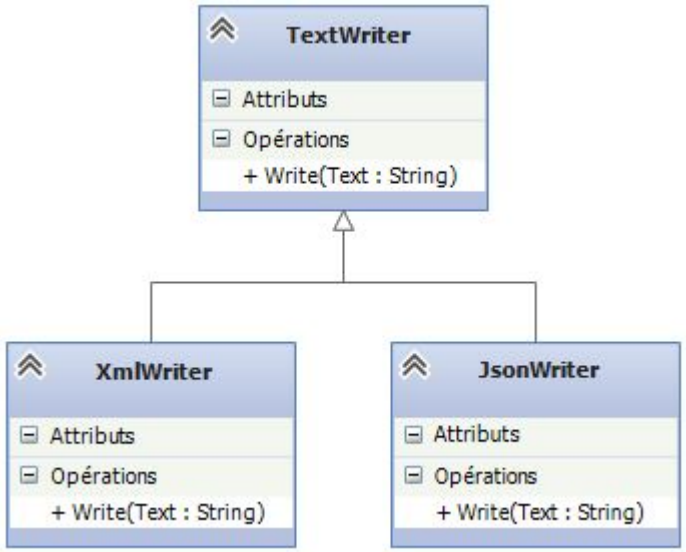
Appelé communément, **surcharge de méthodes**, ce type de polymorphisme s'applique sur la signature de méthodes.

La **signature d'une méthode** reprend le **nom** et les **paramètres** qu'elle reçoit en entrée.

Cela nous permet de déclarer plusieurs méthodes portant le même nom à condition qu'elles reçoivent des paramètres en nombre ou de types différents.



POLYMORPHISME AD HOC



Le polymorphisme « Ad hoc » quant à lui est souvent appelé **redéfinition de méthodes**.

Il permet de redéfinir le fonctionnement des méthodes des classes dérivées par rapport au fonctionnement de la classe parent.

Ces méthodes ont pour particularité d'avoir la **même signature**.

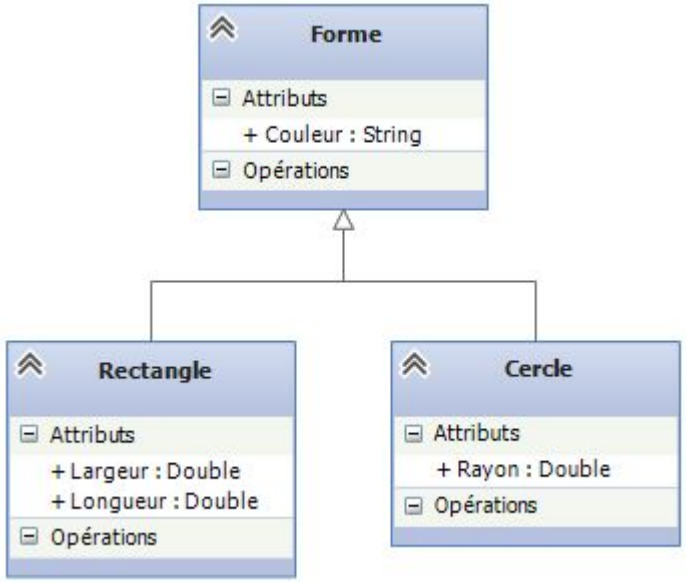
POLYMORPHISME D'HÉRITAGE

Le polymorphisme d'héritage utilise les types eux-mêmes.

En effet une valeur peut être utilisée avec son propre type ou tout type de son(es) parent(s).

Par exemple, nous pourrions déclarer une variable de type « Rectangle » ou « Cercle » et la stocker dans une variable de type « Forme ».

Il faut cependant garder en tête que la variable déclarée sera de type « Forme » et ne nous fournira, au final, que les fonctionnalités déclarées dans la classe « Forme ».



EXERCICES

- Définir le modificateur d'accès de la méthode SetSolde à private dans la classe Compte
- Déterminer les problèmes que cela pose
- Si besoin, adapter la solution pour que cela puisse fonctionner à nouveau tout en laissant la méthode SetSolde en private