L'HÉRITAGE

la programmation orientée objet

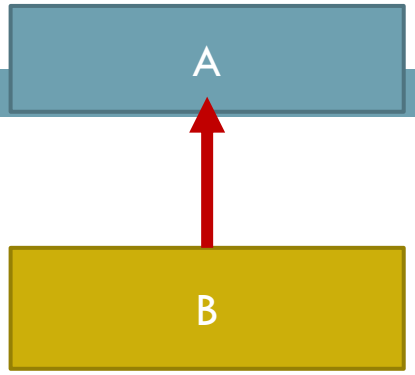
Dr. Jihène Tounsi
IHEC- Carthage
Email: tounsi.jihene@yahoo.fr

L'héritage fondement N°3

- L'héritage est un fondement de l'orienté objet.
- Le principe est de regrouper les mêmes caractéristiques dans une seule classe générique
  - Pas de redondance de code
  - Facilité lors de la modification

  →la réutilisation du code
- L'héritage multiple n'est pas supporté en Java.

Principe



- B hérite de la classe A càd:
  - A est la classe mère ou super-classe.
  - B est la classe fille ou dérivée.
- A peut avoir plusieurs classes dérivées
- B ne peut avoir qu'une seule super classe
- B peut aussi avoir une ou plusieurs classes dérivées
- Toutes les classes héritent implicitement de la classe « Object »

La classe dérivée

- On dérive une classe d'une autre à travers le mot clé : « extends »
- Une classe dérivée peut utiliser directement:
  - Les attributs de la classe mère sauf les attributs private.
  - Toutes les méthodes de la classe mère.
  - Le constructeur de la classe mère
- Une classe dérivée :
  - peut s'enrichir avec d'autres attributs et d'autres méthodes
  - Redéfinir des méthodes existantes dans la classe mère

La classe Object

- La classe Object est la classe de base de toutes les autres.
- C'est la seule classe de Java qui ne possède pas de classe mère.
- Tous les objets en Java, quelle que soit leur classe, sont du type Object. Cela implique que tous les objets possèdent déjà à leur naissance un certain nombre d'attributs et de méthodes dérivées d'Object.
- Dans la déclaration d'une classe, si la clause extends n'est pas présente, la super classe immédiatement supérieure est donc Object.

Le constructeur

- Le(s) constructeur(s) d'une classe fille peut utiliser le constructeur d'une classe mère à travers le mot clé super.
  - Cas1 : pas de constructeur dans la classe mère
  - Cas 2 : Un constructeur non paramétré dans la classe mère
  - Cas 3 : Un constructeur paramétré dans la classe mère.

```
Public Class A {
Public A(paramètre) {
//initialisation des attributs de la classe mère
….}}
```

ou

```
Public Class B extends A {
Public B(paramètre) { //initialisation des attributs de la classe mère
super(paramètre);
….}}
```

Instanciation dans le cadre de l'héritage

Un objet de la classe de base A	Un objet b de la classe dérivée B
allocation mémoire pour un objet du type A	allocation mémoire pour un objet du type B (donc A+B)
initialisation par défaut des champs	initialisation par défaut des champs (A+B)
initialisation explicite des champs	initialisation explicite des champs hérités de A
exécution des instructions du constructeur de A	exécution des instructions du constructeur de A
	initialisation explicite des champs hérités de B
	exécution des instructions du constructeur de B

Exception …. La classe Final

- Une classe avec le mot clé final ne peut être dérivée
- Une méthode avec le mot clé final ne peut être redéfinie dans la classe fille.
- Un attribut avec final veut dire

Que c'est une constante

```
Public final Class Exemple {
public final void Methode()
{ … }


}
```

La classe Abstraite

- Une classe abstraite est une classe qui ne peut pas être instanciée directement. Elle n'est utilisable qu'à travers sa descendance.
- Une classe abstraite doit toujours être dérivée pour pouvoir générer des objets.
- Une classe abstraite est précédée du mot clé abstract
  - Entête de la classe
    - Public abstract class nom_classe{...}

La classe Abstraite

- Dans une classe abstraite, on peut trouver des méthodes abstraites, mais pas forcément.
- Une méthode abstraite n'a pas d'implémentation dans la classe abstraite (pas de code).
- Déclaration de la méthode abstraite
  - Public abstract type_retour nom_méthode(..);

- La méthode abstraite doit obligatoirement être implémentée dans toutes les classes filles.
- Une classe fille qui n'implémente pas la méthode abstraite de la super-classe passe automatiquement à une classe abstraite.
- L'intérêt des méthodes abstraites vient du fait que l'on peut les appeler de la même façon pour tous les objets dérivés.

Une classe qui possède une (ou plusieurs) méthode abstraite est obligatoirement abstraite

Le polymorphisme

- Le polymorphisme dans java veut simplement dire qu'une classe peut prendre plusieurs formes et c'est d'autant plus vrai avec les classes qui hérite d'une classe mère.
- Il y a deux mécanismes de polymorphisme
  - Polymorphisme de méthodes
  - Polymorphisme d'objet ou le transtypage

Le polymorphisme de méthode

- Polymorphisme de méthode
  - C'est la redéfinition d'une méthode
  - Une méthode de la classe mère peut être implémentée différemment dans une classe fille: la méthode est dite redéfinie.
  - Il faut mettre l'annotation **@Override**

- La redéfinition d'une méthode dans une classe fille cache la méthode d'origine de la classe mère.
  - Pour utiliser la méthode redéfinie de la classe mère et non celle qui a été implémentée dans la classe fille, on utilise la mot-clé super suivi du nom de la méthode.

Le polymorphisme d'objet

- Polymorphisme d'objet
  - On peut transtyper (cast) un objet afin de pouvoir utiliser les méthodes d'une classe
    - Exemple : changer le type d'une instance d'une classe mère avec le type d'une classe fille afin de pouvoir utiliser ses méthodes.
- Importance de « instanceof » qui permet de vérifier si un objet est une instance d'une classe
  - Syntaxe : nom_obj instanceof nom_classe
  - Type de retour: boolean

Transtypage implicite

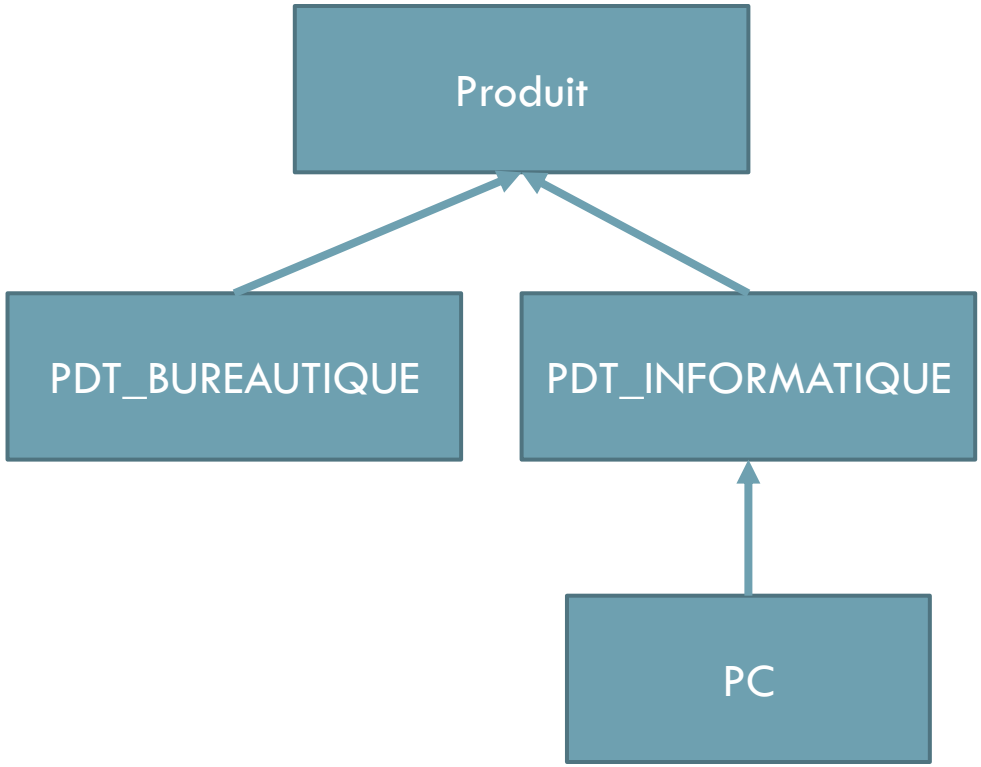
- Le transtypage implicite est traité automatiquement par le compilateur lors d'une affectation ou du passage d'un paramètre.
- Un transtypage peut être implicite si le type cible a un plus grand domaine que le type d'origine (gain de précision).
- Dans le transtypage d'objet implicite, une référence de la classe mère pour désigner un objet d'une classe fille.
- Exemple soit
  - OM : un objet de la classe mère
  - OF: la classe fille
  - OM=OF → c'est-à-dire que la référence de la classe fille modifie son type en utilisant celle de la mère, c'est-à-dire que OM n'a pas le droit d'utiliser les méthodes spécifiques de la classe fille.

Transtypage explicite

- Le transtypage explicite des références est utilisé pour convertir le type d'une référence dans un type dérivé.
- Syntaxe Obj de type hérité
  - ((type_dérivé)Obj).méthode_classe_dérivée()

Application 1

- Un magasin de vente de fournitures bureautiques possède deux familles de produits:
  - Les produits informatiques dont la TVA est de 13%
  - La fourniture de bureau dont la TVA est de 19%.
- Créer les classes JAVA du diagramme en respectant les principes de la POO.



Fin Du chapitre