JAVA
FONDAMENTAUX

Présenté par :

Xavier TABUTEAU

L’héritage

C’est une notion indissociable du langage Java et de la Programmation Orientée Objet (POO). La question à se poser quand on parle d’héritage est :

Est-ce que ma classe est une sorte d’autre classe plus générique ?

Exemple :

Un Smartphone est une sorte de téléphone.

Cela induit donc une notion de parenté. La classe la plus générique sera appelée classe mère et la classe qui en héritera sera appelée classe fille. Les types génériques en Java héritent tous d’une classe nommée Object. Seuls les types de bases n’héritent pas de la classe Object. Par conséquent, la création d’une classe qui n’hérite de rien explicitement, hérite quand même de la classe Object. L’instruction « extends Object » n’est donc pas nécessaire car implicite.

Syntaxe de l’héritage :

```
class Smartphone extends Telephone {
        …
}
```

L’héritage

Avec l’héritage, on peut redéfinir les méthodes héritées. L’annotation « @Override » n’est pas obligatoire mais c’est une bonne pratique pour indiquer :

→ Aux développeurs qu’il s’agit d’une méthode redéfinie.

→ Au compilateur qu’il s’agit d’une méthode redéfinie pour qu’il vérifie son exactitude.

Le mot clé super : permet d’accéder aux attributs et méthodes de la classe mère si la visibilité de celle-ci le permet.

Le mot clé this : permet de référencer les attributs et méthodes de l’objet en cours et de l’objet dont il hérite si la visibilité de celui-ci le permet.

Il est possible de bloquer l’héritage en Java via le mot clé final sur la classe. En effet, cela provoque un blocage complet. Aucune classe ne peut hériter d’une classe définie avec final. On peut aussi limiter le blocage au niveau d’une méthode ou même simplement d’un attribut (devient une constante).

Si une méthode est définie dans la classe mère, et qu’elle n’est pas redéfinie dans la classe fille, alors cette méthode est accessible depuis un objet de la classe fille si la visibilité le permet.

Abstraction

Dans une classe, il est possible de déclarer des méthodes sans code associé.

→ parce qu'il n'est pas possible de fournir un code pour ce niveau d'abstraction.

→ parce qu'on veut mettre en œuvre le polymorphisme.

Exemple :

une classe Voiture avec une méthode rouler()

une classe Camion avec une méthode rouler()

une classe Vehicule avec une méthode abstraite rouler(), c'est à dire sans code car on ne sait pas dire comment le véhicule roule. C'est trop abstrait à ce niveau.

Remarque si on déclare une méthode abstraite, alors la classe aussi doit être abstraite.

Syntaxe de classe abstraite :

```
abstract class NomClasse {
        …
}
```

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 supérieure.

```
25 public class Garage{
26     public static void main(String[] args) {
27         Vehicule v1 = new Voiture(true);
28         Vehicule v2 = new Camion(false);
29         v1.rouler();
30         v2.rouler();
31     }
32 }
```

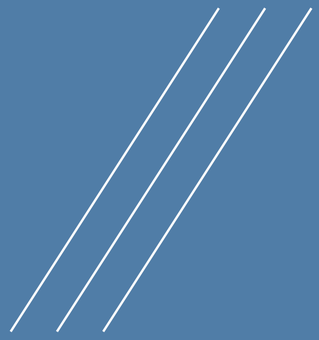
Création de 2 objets de type Vehicule :
1 compatible avec Voiture et
1 compatible avec Camion

Appel de la méthode rouler() possible car rouler() a été définie dans Vehicule

A l'exécution, c'est la méthode définie sur l'objet réel qui est appelée ! ➔ **Polymorphisme**

AbstractionPolymorphisme.java

JAVA FONDAMENTAUX



Présenté par

Xavier TABUTEAU