Compléments de POO en Java

L2-SINFL4A2

Année 2022/2023

Yann Mathet yann.mathet@unicaen.fr

Héritage, Polymorphisme

- Une classe peut servir de modèle à d'autres classes, par le mécanisme d'héritage
- On part du général (classe mère) vers le particulier (classe fille)
- Tout ce qui est déclaré et défini dans la classe mère est présent dans la classe fille
- Mais on peut faire des ajouts et des modifications dans la classe fille

Classe mère, classe fille

- Attention, le vocabulaire « mère » et « fille » est relatif à une relation d'héritage, et non une propriété figée
- Par exemple, la classe B peut être la classe mère de la classe C, et en même temps la classe fille de la classe A
- Vocabulaire : classe mère, super-classe, classe fille, sous-classe, classe dérivée.

Relation d'hériatage = « est un »

- La relation d'héritage correspond à la sémantique « est un » entre la classe fille et la classe mère : toute instance de la classe fille doit pouvoir être considéré comme un élément de la classe mère
- Exemple : Voiture « est un » Véhicule, Rectangle « est une » Forme
- Contre exemple : Cercle n'est pas un Point

Polymorphisme

- La relation « est un » est à la base du polymorphisme
- Le polymorphisme est le fait de pouvoir considérer une entité selon plusieurs types : son type natif, ou n'importe lequel de ses types
- On peut voir une Voiture comme une Voiture mais aussi comme un Véhicule (si Voiture extends Véhicule)

Classes, Instances, Références

- Le polymorphisme opère entre instances et références, à partir d'une relation d'héritage entre classes
- Une instance a TOUJOURS son type natif, quelle que soit la façon dont on y accède
- Mais des références de différents types peuvent pointer vers un objet d'un type donné.

Classes, Instances, Références

- Exemple :
  - Voiture voiture = new Voiture();
  - Véhicule véhicule = voiture; // OUI
  - véhicule.avancer(); // OUI
  - véhicule.ouvrirCoffre(); // NON
  - Véhicule véhicule2=new Voiture(); // OUI

Polymorphisme : utilité ?

- Offrir des traitements génériques
- Exemple :
  - Voiture extends Véhicule
  - Camion extends Véhicule
  - public void garer(Véhicule v) permet de garer des Voitures, des Camions, etc.
  - Sans polymorphisme, il faudrait une méthode par type

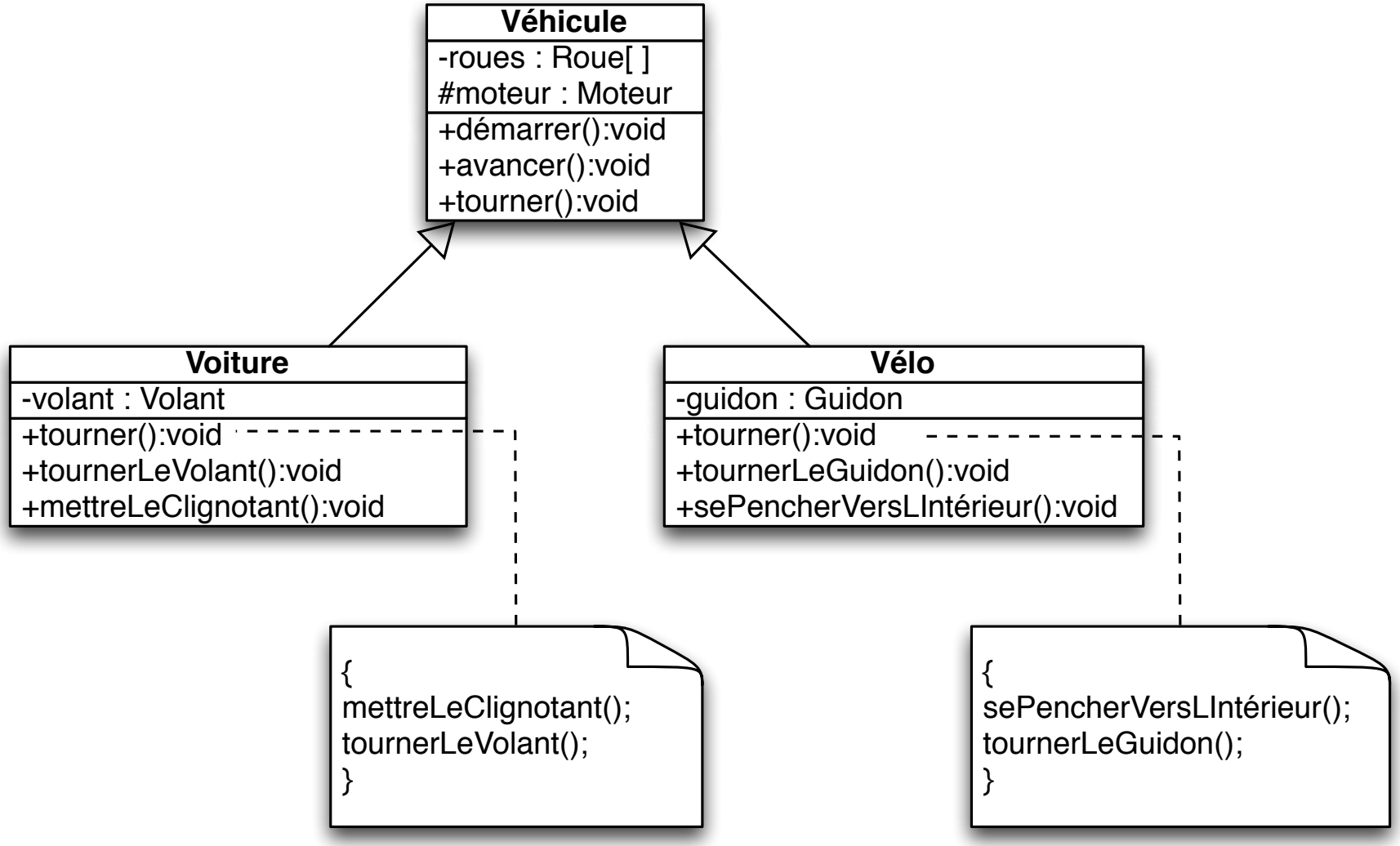
Polymorphisme : restrictions

- Par polymorphisme, on ne peut accéder qu'à une partie générique de l'objet.
- Toutes les méthodes plus spécifiques, bien qu'elles existent au sein de l'objet, ne sont pas accessibles par une référence plus générique
- Lors de la définition d'un traitement, il faut donc viser juste entre trop général (classe Object) et trop spécifique…

Méthodes virtuelles, liage dynamique

- En raison du polymorphisme, il n'est pas toujours possible de savoir quelle méthode va être réellement appliquée à l'exécution lors de la compilation
- En effet, cela dépend du type réel de l'objet, pas du type de la référence. Ce type n'est connu que lors de l'exécution
- On parle alors de méthode virtuelle, dont le code à exectuer est trouvé dynamiquement, i.e. à l'exécution)

Polymorphisme et méthodes virtuelles



Méthodes virtuelles (2)

Code de test :

```
Véhicule[ ] parcAuto=new Véhicule[2];
parcAuto[0]=new Voiture();
parcAuto[1]=new Vélo();
for (int i=0;i<parcAuto.length;i++)
   {
   parcAuto[i].démarrer();
   parcAuto[i].avancer();
   parcAuto[i].tourner();
   }
```

Ce qu'il faut retenir

- Lorsqu'une méthode est redéfinie dans une sous classe, et que l'on y accède par une référence plus générique, c'est malgré tout la méthode redéfinie qui est invoquée !
- Exemple :
  - Véhicule v = new Vélo();
  - v.tourner(); // c'est la méthode redéfinie dans Vélo qui est invoquée (par ex. tourner le guidon)

Polymorphisme : conclusion

- Concerne les références et non les objets
- Un objet possède toujours son type natif et ses méthodes éventuellement redéfinies
- Le polymorphisme permet d'accéder à différents types d'objets dans un même traitement (ex. une méthode), ou de stocker ces différents objets dans une même structure de données (ex. une liste)