Le langage Java

Approche orientée objet – Héritage et redéfinition de méthodes

Chapitre 2
L'héritage

Problématique et solution

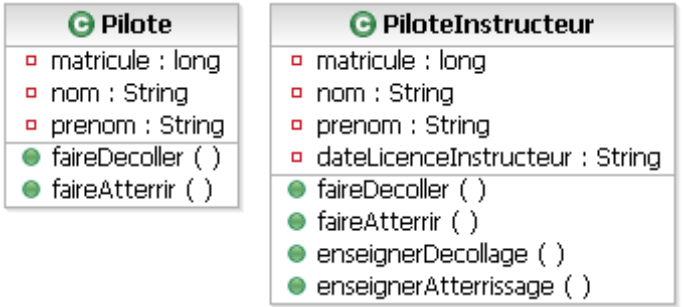
Problématique

Exemple

Un **pilote** a : un nom, un prénom un matricule.
Il sait piloter un avion (le faire décoller et le faire atterrir).

Un **pilote instructeur** a : un nom, un prénom, un matricule, **une licence d'instructeur**.
Il sait piloter un avion (le faire décoller et atterrir);
il peut également enseigner à des apprentis.

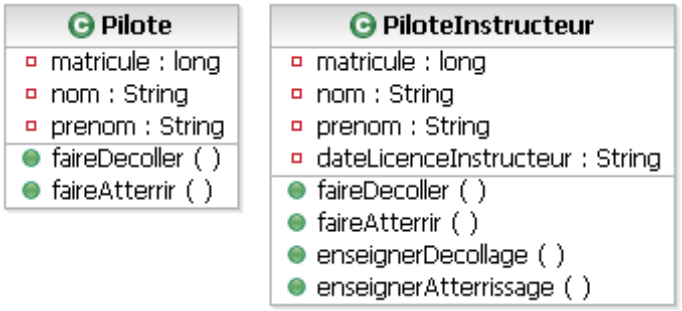
Modèle sans héritage



Duplication de code difficile à maintenir

Problématique

Modèle sans héritage



Duplication de code difficile à maintenir

Solution : l'héritage

Modèle avec héritage

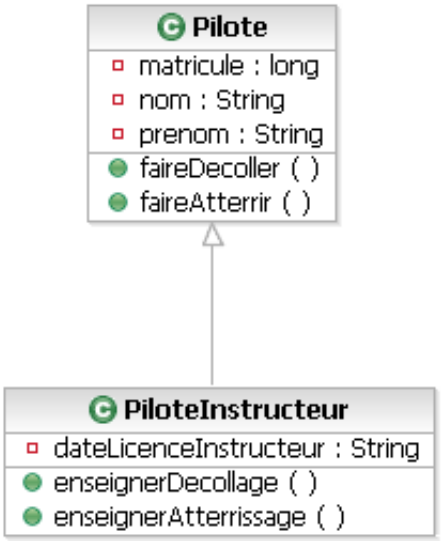
un pilote instructeur **est un type particulier de pilote** (spécialisation)

Relation d'héritage

Pilote définit les comportements et attributs communs

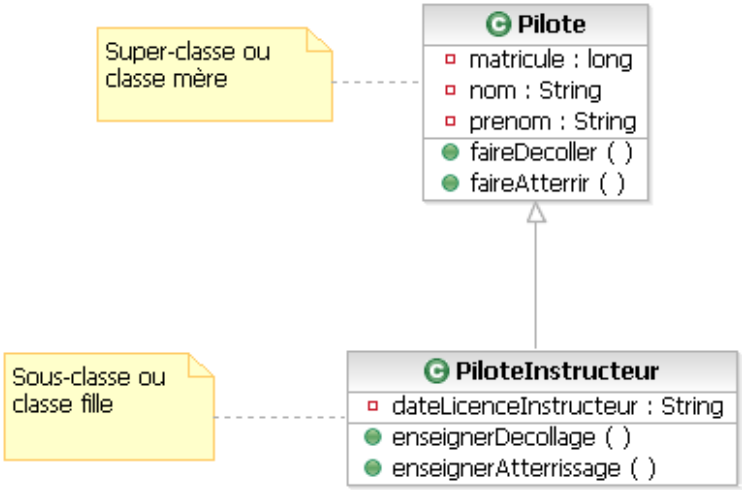
PiloteInstructeur ne définit que ce qui lui est propre

Pas de duplication de code



Relation d'héritage en Java

Mot-clé **extends** dans la signature de la sous-classe.




```java
public class PiloteInstructeur extends Pilote {
  private String dateLicenceInstructeur;

  public void enseignerDecollage(){
    // code pour enseigner le décollage
  }

  public void enseignerAtterrissage(){
    // code pour enseigner le décollage
  }
}
```

Comportement (exécution)

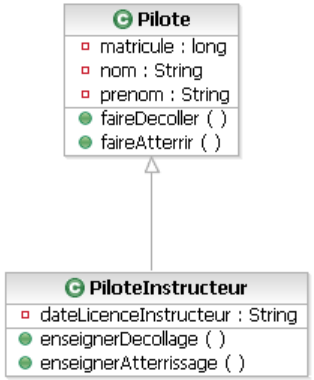
Création

```
PiloteInstructeur instructeur = new PiloteInstructeur();
```

L'objet instructeur a 4 attributs en mémoire.

Appels de méthode

Sur l'instance de classe fille, on peut appeler des méthodes de la classe fille ou de la classe mère (à condition que la visibilité le permette).



Hiérarchie des classes

Arborescence

Une classe ne peut avoir qu'une seule classe mère

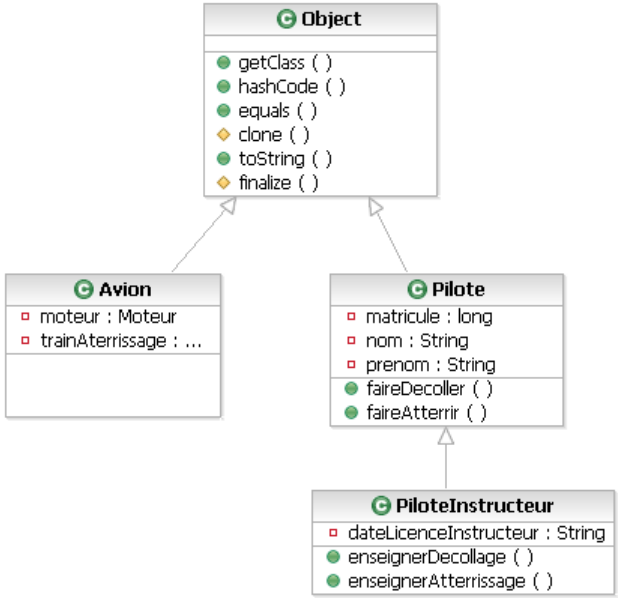
Classe mère implicite : **java.lang.Object**

Méthodes communes à tous les objets

`String toString()` : *retourne la forme textuelle de l'objet, ici le nom long de la classe + un identifiant.*

`boolean equals(Object)` : *indique si un objet est égal à un autre, ici si ce sont strictement le même objet en mémoire*

…



Redéfinition de méthode

Définir dans une classe fille une méthode avec la même signature que dans la classe mère

Dans ce cas, la méthode de la classe mère est ignorée

Redéfinition de méthodes

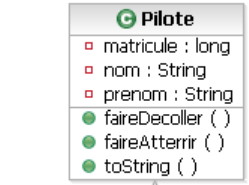
Contraintes

1) Le type de retour doit être strictement identique
2) Les paramètres sont strictement identiques
3) Visibilité doit être égale ou supérieure

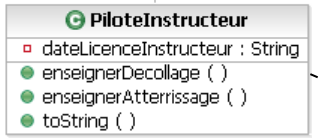
Exemple : une méthode définie comme publique dans la super-classe ne peut pas être redéfinie comme privée dans la sous-classe

Redéfinition de méthodes

Exemple de redéfinition




```
public String toString(){
    return prenom + " " + nom;
}
```

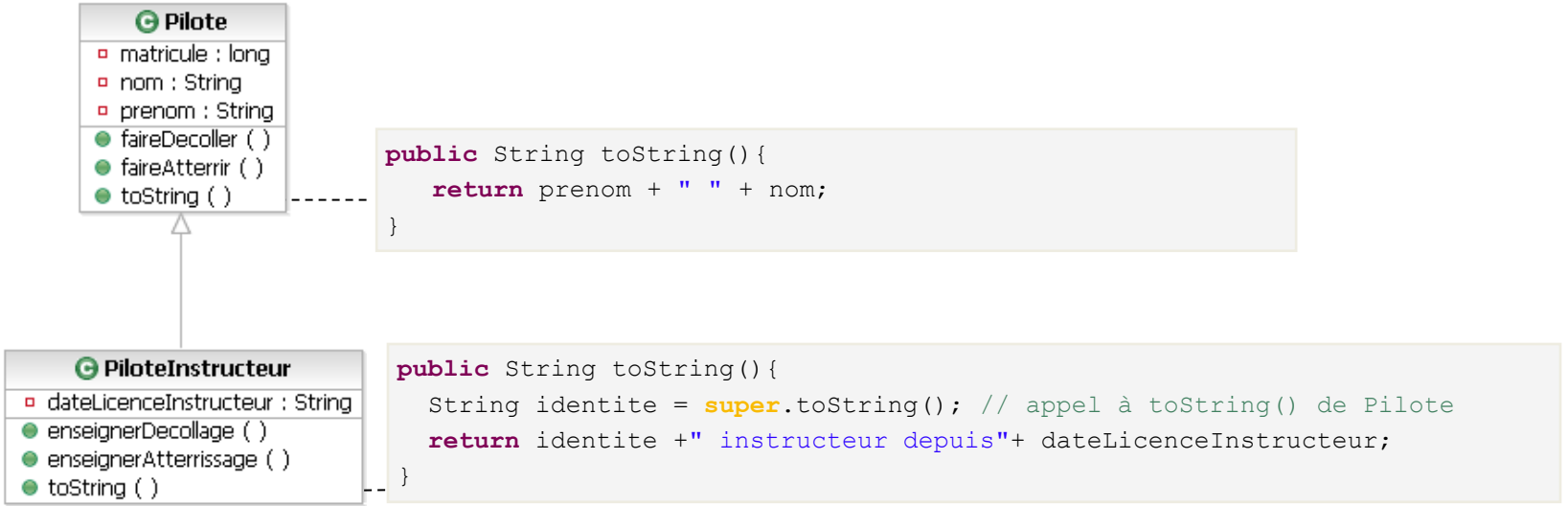



```
public String toString(){
  return "instructeur depuis " + dateLicenceInstructeur;
}
```

Redéfinition de méthodes

Exemple de redéfinition avec mot clé **super**

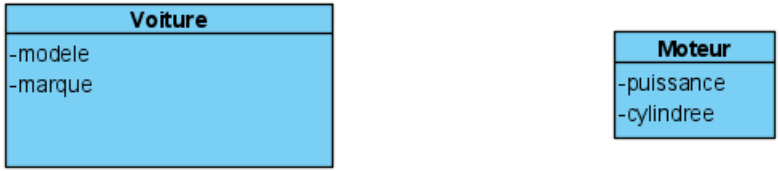
Le mot clé super permet de réutiliser la méthode située dans la classe mère.



Héritage vs Association

Comment sait-on si on doit hériter d'une autre classe où si on doit les associer ?

Exemple :

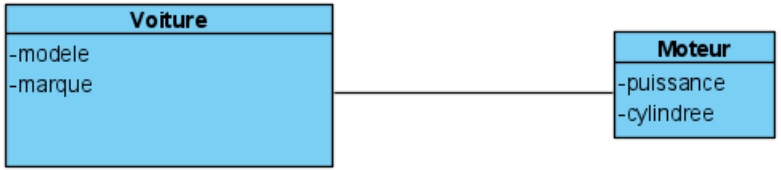


-Est-ce que Voiture hérite de Moteur ?
-Est-ce que Voiture a un attribut de type Moteur ?

Héritage vs Association

Si on peut utiliser le verbe être : héritage
Si on peut utiliser le verbe avoir : association

Une Voiture a un moteur. Une Voiture n'est pas un type particulier de Moteur.



-Est-ce que Voiture hérite de Moteur ? NON
-Est-ce que Voiture a un attribut de type Moteur ? OUI

Atelier (TP)

OBJECTIFS : redéfinir la méthode toString

DESCRIPTION : Dans le TP suivant vous allez devoir redéfinir la méthode toString().