Etude comparative des mécanismes d'héritage dans les langages C#, C++, Eiffel et Java

BOUCHEREZ Kévin
SAUSSIER Pascal

A quoi sert l'héritage?

L'héritage permet de modéliser de manière simple le monde qui nous entoure

Deux types d'héritages

- **Héritage simple.**
- **Héritage multiple.**

L’héritage simple

Description de l'héritage simple

- **Une classe fille n'hérite que d'une seule classe mère.**

Exemple :

Une Voiture « EST_UN » Véhicule.

Comparatif de l'héritage simple

1) **L'héritage de classes et sa syntaxe**
2) **Les niveaux d'accessibilité**
3) **Les constructeurs**
4) **Le polymorphisme**
5) **Interface**

L'héritage de classes et sa syntaxe

- **En Java :**

public class Voiture extends Vehicule

- **En C# :**

public class Voiture : Vehicule

- **En C++ :**

class Voiture : public Vehicule

- **En Eiffel :**

class Voiture inherit Vehicule

Les niveaux d'accessibilité

- **En Java :**

public, private, protected

- **En C# :**

public, private, protected, internal, internal protected

- **En C++ :**

public, private, protected

- **En Eiffel :**

Par défaut tout est publique, il est possible de définir des membres de manière privée : feature {NONE}

Les constructeurs

Les quatre langages ont un constructeur sans paramètre par défaut si aucun constructeur n'est créé.

- **En Java, C# et C++ ce constructeur par défaut porte le même nom que la classe.**
- **En Eiffel, le constructeur par défaut n'a pas de nom. Si le programmeur en définit un il lui donne le nom qu'il veut.**

Le polymorphisme

- **En Java et C++, la redéfinition de méthodes se fait de manière transparente (les méthodes de classes ancêtres restant toutefois accessibles).**
- **En C#, le compilateur suggère au programmeur de faire précéder la méthode redéfinie par le mot clé *new.***
- **En Eiffel, il n'y a pas de surcharge mais on peut renommer, redéfinir ou éliminer une méthode héritée.**

Interface

Une interface exprime la relation : PEUT_ETRE_UTILISEE_COMME.

Une Voiture PEUT_ETRE_UTILISEE_COMME un Vehicule.

- **En Java :**

public interface Vehicule{…}

public class Voiture implements Vehicule{…}

Interface

- **En C# :**

interface IVehicule{…}

public class Voiture : IVehicule{…}

- **En C++ et en Eiffel :**

La notion d'interface n'est pas présente.

L’héritage multiple

Description de l'héritage multiple

- **Une classe fille hérite de plusieurs classes mères.**

Exemple :

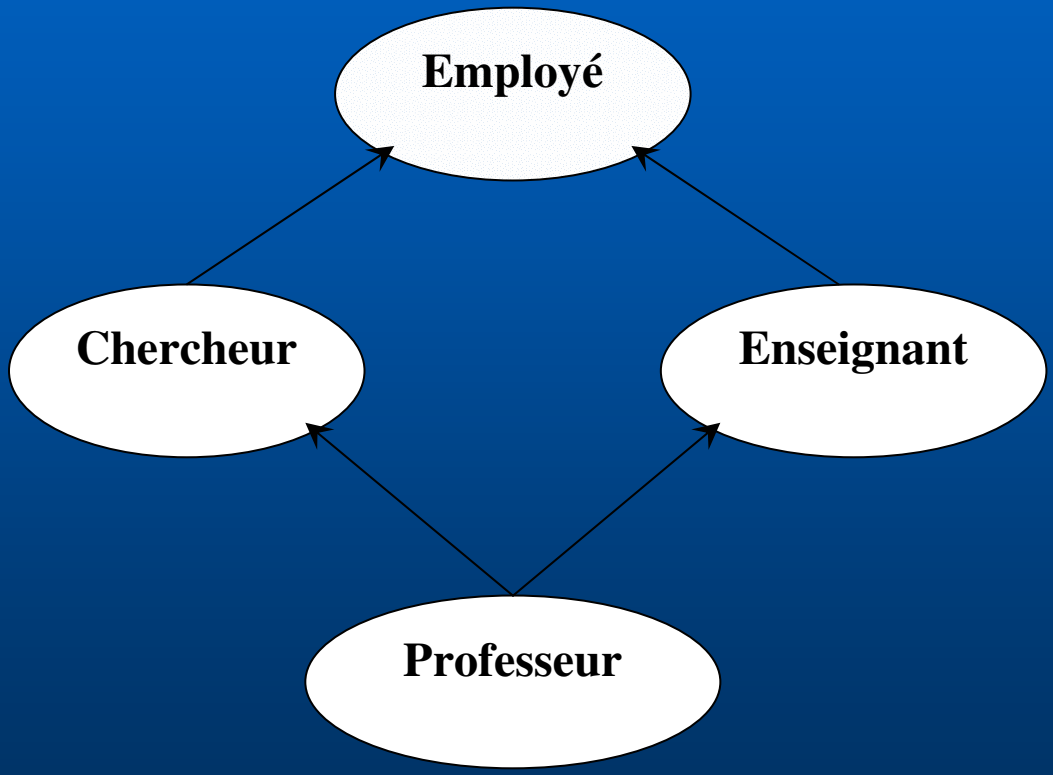
Un Professeur « EST_UN » Enseignant et « EST_UN » Chercheur.

Description de l'héritage multiple (suite)

Lors de l'héritage multiple d'une même classe mère les membres de cette classe peuvent être hérités de deux façons:

- **Héritage commun**
- **Héritage répété**

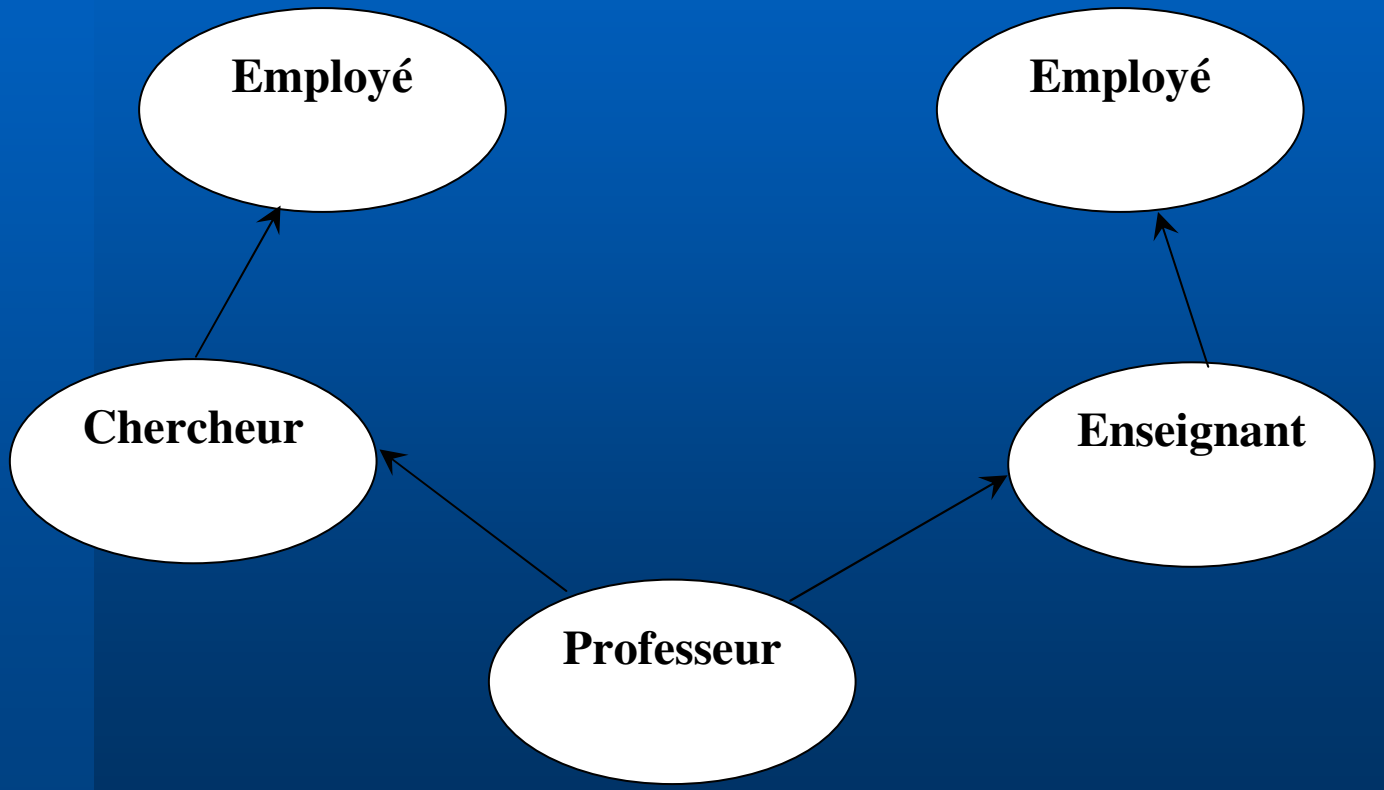
Héritage commun



Héritage commun (suite)

Dans un héritage commun les membres d'Employé n'apparaissent qu'une seule fois dans la représentation en mémoire

Héritage répété



Héritage répété (suite)

Dans un héritage répété les membres d'Employé apparaissent plus d'une fois dans la représentation en mémoire

Comparatif

En C++

Comparatif (suite) – C++

- **En cas d'héritage commun, les clauses d'héritage concernées doivent être préfixées du symbole *virtual*.**

Exemple **:**

```
class Chercheur : virtual public Employé{…}
class Enseignant : virtual public Employé{…}
class Professeur : public Chercheur, public Enseignant{…}
```

Comparatif (suite) – C++

- **L'héritage répété est le type d'héritage par défaut de C++, il n'y a pas de mot clé spécifique.**

Exemple **:**

```
class Chercheur : public Employé{…}
class Enseignant : public Employé{…}
class Professeur : Chercheur, public Enseignant{…}
```

Comparatif (suite) – C++

- **Pour un peu plus de clarté C++ permet de spécifier explicitement le membre que l'on désire atteindre.**

Exemple **:**

Dans la classe Professeur on peut dire :

Chercheur::getSalaire() ou
Enseignant::getSalaire() ou bien
Employé::getSalaire()

Comparatif

En Eiffel

Comparatif (suite) – Eiffel

- **Par défaut Eiffel fait de l'héritage commun.**
- **Pour pouvoir faire de l'héritage répété, il faut que les classes intermédiaires renomment les membres à hériter de manière répétée.**

Comparatif (suite) – Eiffel

Exemple :

```
class Employé
feature
    k: INTEGER
end;

class Enseignant
inherit
    Employé rename k as eek end
feature
…
end;
```

```
class Chercheur
inherit
    Employé rename k as eck end
feature
…
end;

class Professeur
inherit
    Enseignant; Chercheur
feature
…
end;
```

Conclusion

L'utilisation de l'héritage simple en Java a permis de séduire bon nombre de personnes de par sa simplicité, c'est donc pour cela que Java n'a pas d'héritage multiple, il souhaitait rester simple. C#, quand à lui n'innove pas beaucoup, il reprend les grands principes de Java tout apportant quelques améliorations.

Conclusion (suite)

- C++ n'est pas un langage pur objet, et bien qu'il permette l'héritage multiple sa mise en pratique peut parfois ne pas sembler évidente.
- Enfin, Eiffel semble beaucoup plus strict quand à l'utilisation de l'héritage multiple.

FIN