Développement Logiciel Java

11/02/2025

L’Héritage

Hériter d’une classe revient à spécialiser certains de ses comportements et certaines de ses propriétés tout en profitant de ses services de base et, ainsi, éviter toute redondance de code.

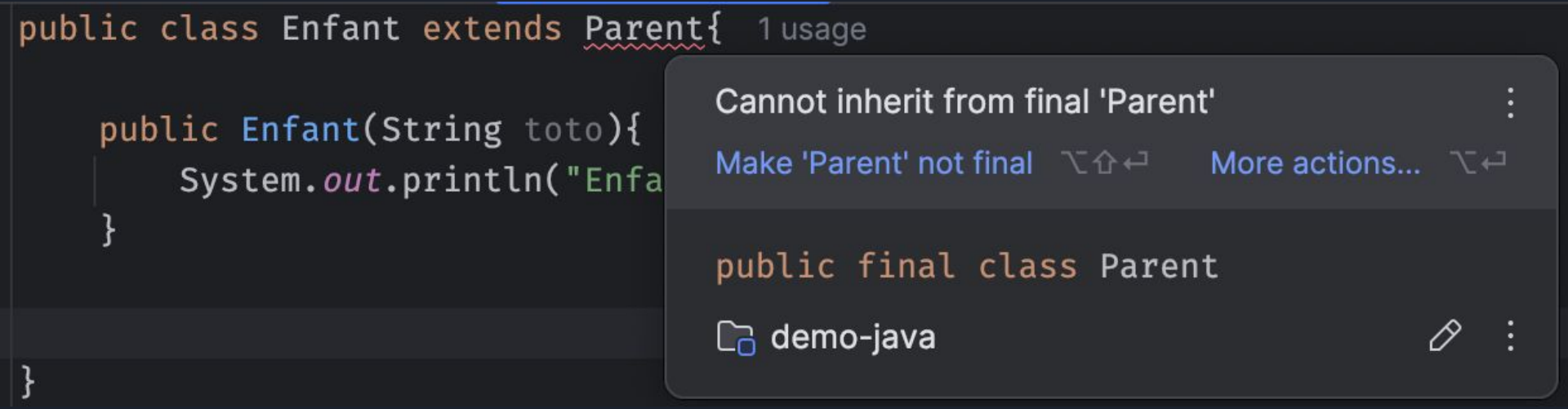
En Java, il y a un seul héritage par niveau mais une classe peut servir de parent à plusieurs classes (comme par exemple la classe java.lang.Object).

On utilise le mot-clé *extends* pour signaler l’héritage d’une classe.

Le Mot-Clé Final

Le mot-clé final dans la déclaration d'une classe permet d'en interdire l'héritage:

```
public final class Parent
```



Les constructeurs

Lorsqu'une classe héritière est instanciée, le constructeur de sa superclasse est appelé avant le sien.

Si la superclasse contient un constructeur vide (déclaré ou par défaut), ce mécanisme est automatique.

Dans le cas contraire, la classe héritière DOIT faire l'appel à un constructeur de la superclasse, sur la première ligne de ses constructeurs, grace au mot-clé *super*.

```
public class Enfant extends Parent{  1 usage

    public Enfant(String param){  1 usage
        super(param);
        System.out.println("Enfant");
    }
}
```

Le Mot-clé *super*

Le mot-clé *super* permet d'appeler le constructeur du parent mais également d'accéder aux membres de la classe de base.

Il est facultatif si le membre à atteindre porte un nom unique dans toute la hiérarchie.

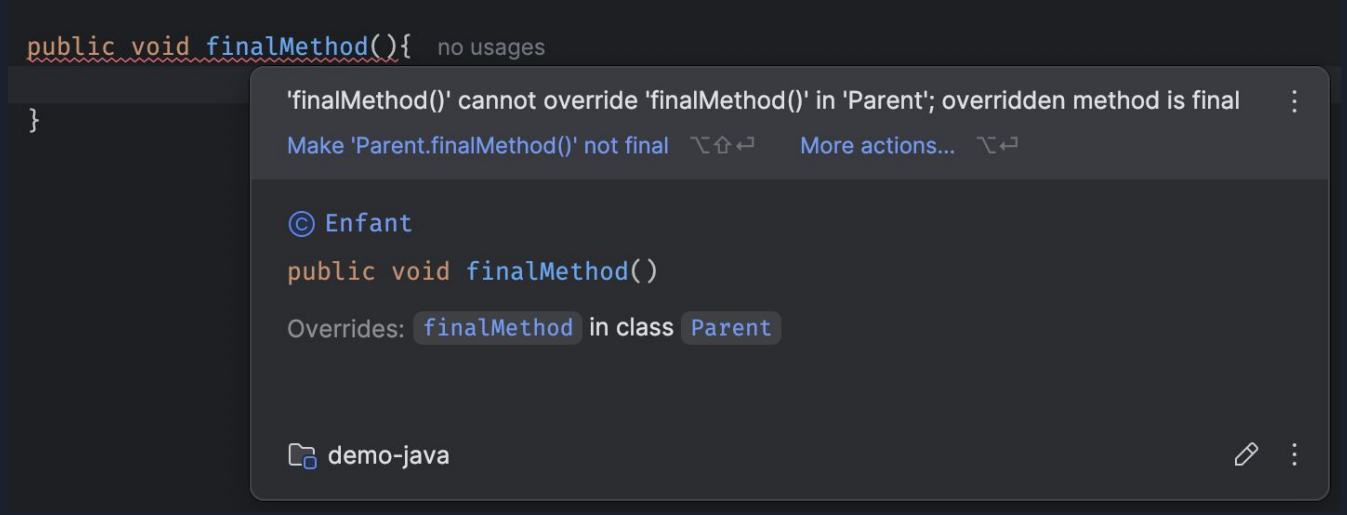
Redéfinition de méthodes

En spécialisant une classe de base, le développeur ajoute de nouveaux membres mais peut également remplacer certains membres existants de la classe de base pour aller vers des comportements spécifiques. On parle alors de méthodes virtuelles.

La déclaration d'une méthode dans la superclasse n'implique pas une substitution obligatoire du membre dans les classes héritières. Cette substitution se fera pour des besoins de spécialisation uniquement.

Méthodes de type final

De la même manière que l'on peut interdire l'héritage d'une classe spécifique, il est possible d'interdire la redéfinition d'une méthode par les descendants. C'est à nouveau le mot-clé *final* qui est employé.



Les classes abstraites

- Il peut arriver qu'une superclasse contienne des méthode qui ne peuvent pas être implémentées (elles dépendent de détails spécifiques à la spécialisation).
- On peut alors déclarer cette classe comme abstraite (*abstract*)
- Il est impossible d'instancier directement une classe abstraite.
- Une classe abstraite contient des méthodes abstraites, sans implémentation, à la manière des interfaces java.
- Une classe abstraite peut contenir également des méthodes classiques.
- A la différence des interfaces, une classe abstraites peut contenir des constructeurs, des blocs d'initialisation et des paramètres.

Le Polymorphisme

Le polymorphisme permet à une classe d'être considérée comme:

- Une de ses classes parentes
- L'une de ses interfaces.

Grâce à la virtualisation des méthodes, le traitement d'un appel de méthode est transféré à l'implémentation pertinente de la classe héritière.

Il est donc préférable de typer les traitements avec des interfaces, afin de réduire au maximum le couplage entre les modules.

instanceof

- L'opérateur instanceof permet de vérifier si un objet peut être considéré comme un instance d'un certain type (sa classe propre, ses classes parentes et ses interfaces).

```
Enfant p = new Enfant();
if (p instanceof Parent) {
    System.out.println("TRUE");
}
```

getClass

- La méthode *getClass()* renvoie la classe spécifique d'un objet.

```
Enfant enfant = new Enfant();
System.out.println(enfant.getClass()); // "class Enfant"
```

Le transtypage (casting)

- Après s'être assuré qu'un objet est du type attendu, il est possible de le caster pour pouvoir exploiter des membres spécifiques du type enfant.

```
Parent parent = new Enfant();

parent.methodeEnfant();

if (parent instanceof Enfant) {
    Enfant e = (Enfant)parent;
    e.methodeEnfant();
}
```

Les classes imbriquées

- Il est possible de déclarer une classe à l'intérieur d'une autre classe.
- En général il s'agit d'une classe privée
- La classe hôte a accès à toutes les propriétés (y compris privées) de la classe imbriquée, et inversement.
- Pour pouvoir instancier la classe imbriquée depuis l'extérieur, elle doit être déclarée static.

Les classes anonymes

- Une classe anonyme est un type de classe imbriquée.
- La classe est instanciée et déclarée en même temps.
- Elle peut être l'héritage d'une classe parent, ou l'instanciation d'une méthode.

```
public static void main(String[] args) {
    Iterator<Integer> myIterator = new Iterator<Integer>() {

        @Override
        public boolean hasNext() {
            return false;
        }

        @Override
        public Integer next() {
            return 0;
        }
    };
```

Les Interfaces fonctionnelles

Une interface fonctionnelle est une interface qui ne possède qu'une seule méthode abstraite.

C'est à partir des interfaces fonctionnelles qu'ont été conçues les expressions lambdas en Java.

Les Expressions Lambdas

Syntaxes d'une expression lambda

```
(paramètres optionnels) ->
{
   // corps de la fonction anonyme contenant des traitements
}
```

OU:

```
(paramètres optionnels) -> expression;
```

Si le traitement ne comporte qu'une ligne.

Les types des paramètres ne sont pas précisés car ils sont connus du compilateur. La valeur de retour également.