POO

en Java 8

Package, visibilité et Encapsulation

JDW(34)

Modularité

La modularité est une **propriété fondamentale** de tous les langages orientés objets. Un programme devient un ensemble de petites entités informatiques qui interagissent et communiquent par messages.

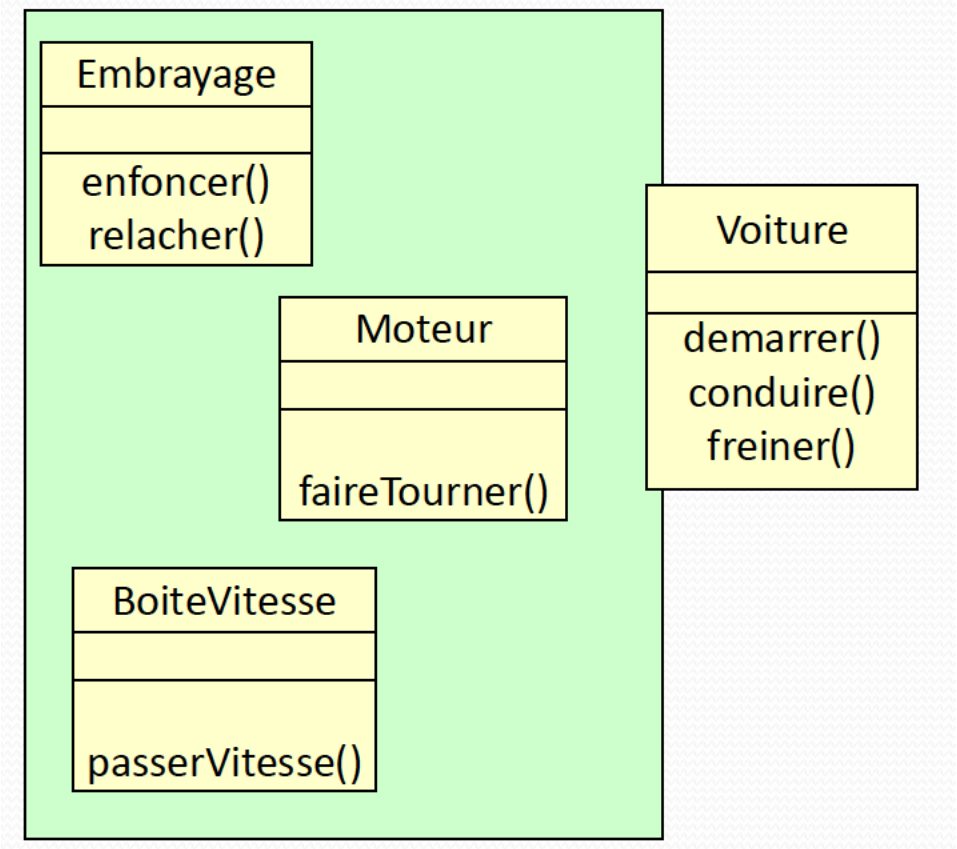
- **Le découpage en module se réfléchit au moment de la conception**
- Le but est d'assurer **le minimum de couplage** entre les modules

Un des moyens pour programmer de façon modulaire est de **découper le programme en packages**.

Les packages

Bien identifier la vue du projet (ce qui est visible dans l'IDE)
Les packages peuvent être imbriqués
De l'extérieur, les classes publiques sont visibles sous le nom package.classe
Exemple avec la classe Voiture

Dans le package vehicule :




```
package vehicule;
class Voiture {
	Embrayage embrayage;
	Moteur moteur;
	BoiteVitesse boite;
	public void demarrer() {
		embrayage.enfoncer();
		moteur.faireTourner(.);
		boite.passerVitesse(1);
		embrayage.relacher();
	}
}
```

Les packages

Pour pouvoir directement accéder aux classes d'un package sans avoir à les préfixer, utilisez la syntaxe : **import NomPackage1. NomPackage2.*;**
NB : * ne concerne qu'un seul niveau d'imbrication
Seules les classes visibles pourront être importées.
Attention aux **conflits** éventuels : Deux classes de même noms dans deux packages différents.
Dans ce cas : importer que le contenu d'un seul package. Le second sera accédé par la syntaxe *import NomPackage1.classe* dans le programme.

```
import package1.Classe1;
import package2.Classe1;
Classe1 maVar = new
Classe1(); // ?????
```

```
import package1.Classe1;
Classe1 maVar = new Classe1();
Package2.Classe1 maVar2 = new
package2.Classe1();
```

Les packages imbriqués

Il existe une priorité de la classe nommée sur l'*.

En général, les packages sont imbriqués les uns dans les autres :
package1.Classe11
package1.Classe12
package1.package2.Classe21
package1.package2.Classe22

Attention :
import package1.*; // importe les classes de package1 mais pas // celles de package2.
import package1.package2.*; // idem mais pour package2

Visibilité - abstract - final

Les classes

• public : classes accessibles sans restriction

• par défaut : classes visibles de la totalité du package, mais pas hors du package

Les propriétés ou méthodes

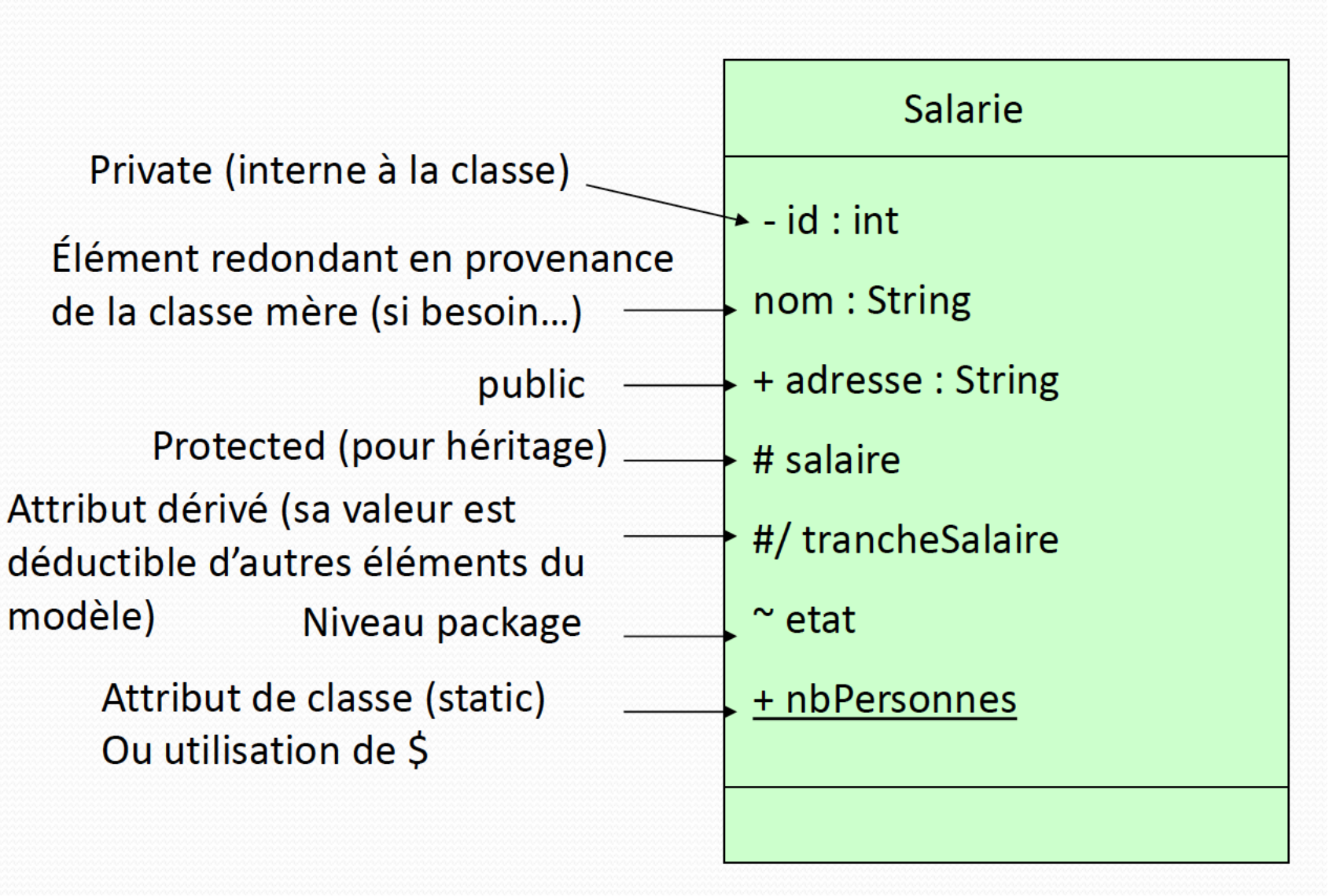
• public : éléments accessibles sans restriction

• private : éléments accessibles uniquement dans la classe locale. Membres non hérités

• protected : éléments accessibles au niveau package et dans ses classes dérivées (même si elles sont dans un autre package).

• par défaut : équivalent de public

Visibilité / UML



Encapsulation

Les méthodes peuvent être public, protected ou private.

 Les méthodes "public" constituent l'interface de l'objet avec l'extérieur.

 Les méthodes "private" sont invisibles de l’extérieur.

 Les méthodes "protected" sont accessibles dans la classe et ses classes dérivées ou dans le même package.

Voir Démo (Encapsulation)

Encapsulation

Principe :

 les attributs ne doivent pas être accessibles de l’extérieur.

 Seules les opérations sont habilitées à modifier les valeurs des attributs.

 Les méthodes accesseurs ou méthodes get ou méthodes **getters** permettent de "sortir" des informations.

 Les méthodes modifieurs ou méthodes set ou méthodes **setters** permettent "d'entrer" des informations.

Destructeur

Le *destructeur* est une méthode particulière qui permet, lorsqu'il est activé, de **nettoyer la mémoire** des objets qui ne sont plus utilisé (lorsque l'objet n'est plus référencé).
Le mécanisme utilisé est le ***garbage collector*** que l'on peut appeler par System.gc();

Lors de son action, gc recherche et exécute, si elle existe, la méthode **finalize()** qui doit être surchargée dans la classe.

```
// destructeur dans la classe
public void finalize() {
      System.out.println ("Objet détruit, Nombre de user : " + --nbUser);
}
// appel de gc dans main
public static void main(String[] args) throws Exception { ….
      System.gc();
}
```

Exercices

A vous pour les exercices