Informatique - Java

Nicolas Rousset

Java - les classes

Java étant particulièrement orienté objet, les classes sont omniprésentes.
Ainsi même si l'on veut définir des fonctions qui ne sont pas rattachées à une classe particulière, on est obligée de créer une classe et de les définir comme static liés à cette classe.

Qu'est-ce qu'une classe ?

On peut définir une classe de deux façons :

- comment étant la représentation d'un objet modélisé
- comme étant un rassemblement de variables (les attributs) et des méthodes pour les traiter

Par exemple si l'on souhaite modéliser un avion, on pourra avoir des classes Passagers, Moteurs, Pilote, Aile, etc ...

Toutefois nous verrons que les classes ne se rattachent pas toujours à un objet réel que l'on souhaite modéliser.

Qu'est-ce qu'une classe ? (2)

Si l'on prend un exemple simple, une personne, on la représentera par une classe Personne. Celle-ci possèdera des attributs, par exemple :

- un nom sous forme de String
- un prénom sous forme de String
- un numéro de sécurité sociale sous forme de Long

On est donc obliger pour représenter un objet réel de déterminer l'ensemble des données qui le qualifie. On en aura donc une représentation limitée.

Les attributs d'une classe peuvent très bien être une autre classe.

Classe et instance de classe

Il faut distincter deux élements :

- la classe elle-même
- les instances de cette classes

Par exemple si l'on définit une classe Voiture, celle-ci représente le concept de général de voiture, et chaque voiture sera représentée par une instance.

Définition d'une classe

On définit une classe ainsi :

```
public class Personne
{
    private String prenom;
    private String nom;
    private int age;
}
```

Il s'agit d'une classe possédant 3 attributs, un prénom sous forme de champ texte, un nom sous forme de champ texte, un age.
Les attributs sont toujours **private**, sauf exception.

Définition d'une classe (2) - les modifiers

Lorsque l'on déclare une classe, on utilisera un certain nombre de qualificatifs sur les attributs et les méthodes. Les principaux sont :

Les modifiers d'accès :

- public : la méthode est accessible depuis n'importe quel endroit du code (normalement on ne l'utilise pas pour un attribut)
- protected : la méthode ou l'attribut est accessible à l'intérieur de la classe et des classes dérivées (voir plus loin)
- private : la méthode ou l'attribut est accessible uniquement à l'intérieur de la classe

Définition d'une classe (3) - les modifiers

Au delà des modifiers d'accès, il existe :

- final (attributs) : indique qu'une valeur ne peut être modifiée une fois initialisée. La valeur doit donc être initialisée dans le constructeur
- static (attributs ou méthode) : indique que l'attribut ou la méthode est rattachée à la classe. Pour un attribut il est donc commun à toutes les instances de la classe. Pour une méthode, cela signifie qu'elle ne peut accéder qu'à des méthodes et des attributs statiques.

Définition d'une classe (4) - getters and setters

private => par défaut on n'expose pas les attributs (pas d'accès ou de modification directe)

Du coup on définit des méthodes associées pour accéder et modifier les valeurs. Un attribut peut n'avoir ni l'un ou l'autre.

```
public class Personne
{
    // Convention de nommage, les noms commencent par une minuscule
    private String prenom;
    private String nom;
    private int age;

    // convention de nommage, get + le nom de l'attribut avec une majuscule au début
    public String getNom()
    {
        return nom;
    }

    // même chose, set + le nom de l'attribut avec une majuscule au début
    public void setNom(String nom)
    {
        this.nom = nom;
    }
}
```

Définition d'une classe - les constructeurs

Le constructeur est, comme son nom l'indique, la méthode qui est appelée pour construire la classe, pour l'initialiser.
Il se déclare comme une fonction avec le même nom que la classe, et en omettant le type de retour :

```
public class Personne
{
    private String prenom;
    private String nom;
    private int age;

    public Personne()
    {
        return nom;
    }

    // même chose, set + le nom de l'attribut avec une majuscule au début
    public Personne(String prenom, String nom, int age)
    {
        // Ici le this sert à distinguer le paramêtre
        // prénom de l'attribut prénom
        this.prenom = prenom;
        this.nom = nom;
        this.age = age;
    }
}
```

Comme toute méthode, un constructeur peut être surchargé : il peut y avoir plusieurs constructeurs, forcément de même nom, pourvu que les attributs soient différents. Le compilateur choisira le bon.

On appelle le constructeur avec l'opérateur **new** :

```
Personne personne = new Personne( "Jean", "Dupont", 18 );
Personne johnDoe = new Personne();
```