Kabola

Formation Développeur Java SE

Manuel de formation

Ce manuel n'est pas un cours complet

Contact
+ 242 06 602 22 22
+ 242 05 383 45 45

Module 1
Section 2

Kabola

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Concept de classe

L'idée de base de la programmation orientée objet est de rassembler dans une même entité appelée objet les données et les traitements qui s'y appliquent.

Une classe est le support de l'encapsulation : c'est un ensemble de données et de fonctions regroupées dans une même entité. Une classe est une description abstraite d'un objet. Entre classe et objet il y a, en quelque sorte, le même rapport qu'entre type et variable.

Java est un langage orienté objet : tout appartient à une classe sauf les variables de types primitives. Pour accéder à une classe il faut en déclarer une instance de classe ou objet.

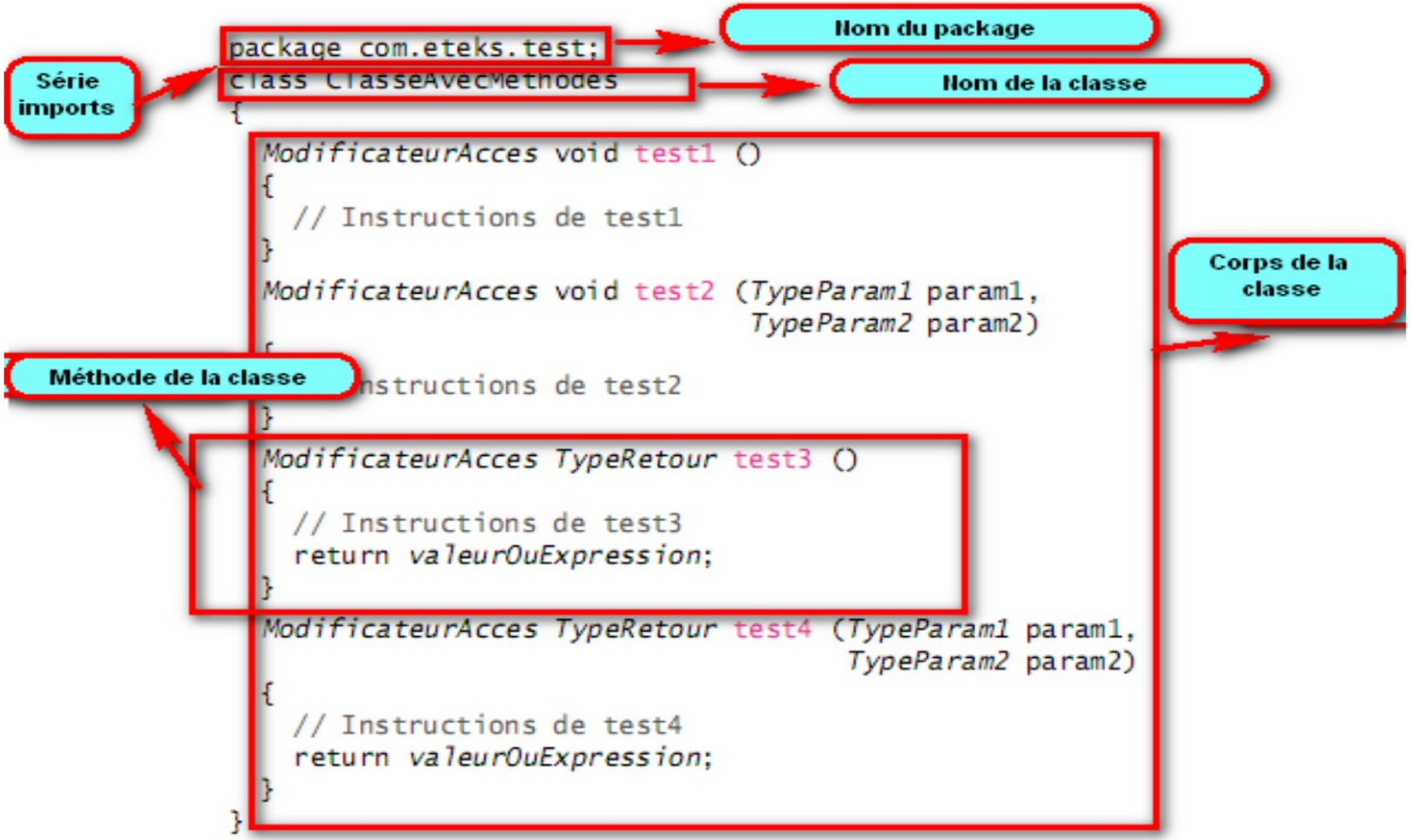
Une classe se compose de deux parties : un en-tête et un corps. Le corps peut être divisé en 2 sections : la déclaration des données et des constantes et la définition des méthodes.

Kabola

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Structure d'une classe



Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Syntaxe de déclaration d'une classe

La syntaxe de déclaration d'une classe est la suivante :

```
modificateurs class nom_de_classe [extends classe_mere] [implements interfaces]
{
    ...

}
```

Modificateur	Rôle
abstract	la classe contient une ou des méthodes abstraites, qui n'ont pas de définition explicite. Une classe déclarée abstract ne peut pas être instanciée
final	la classe ne peut pas être modifiée, sa redéfinition grâce à l'héritage est interdite. Les classes déclarées final ne peuvent donc pas avoir de classes filles.
private	la classe n'est accessible qu'à partir du fichier où elle est définie
public	La classe est accessible partout

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Concept d'objets

Les objets contiennent des attributs et des méthodes. Les attributs sont des variables ou des objets nécessaires au fonctionnement de l'objet.

En Java, une application est un objet. La classe est la description d'un objet. Un objet est une instance d'une classe. Pour chaque instance d'une classe, le code est le même, seules les données sont différentes à chaque objet.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Cycle de vie d'un objet

Les objets ne sont pas des éléments statiques et leur durée de vie ne correspond pas forcément à la durée d'exécution du programme.

Le cycle de vie d'un objet passe par trois étapes :

- la déclaration de l'objet et l'instanciation grâce à l'opérateur new

  Exemple :

  ```
  nom_de_classe nom_d_objet = new nom_de_classe( ... );
  ```

- l'utilisation de l'objet en appelant ses méthodes
- la suppression de l'objet : elle est automatique en Java grâce à la machine virtuelle. La restitution de la mémoire inutilisée est prise en charge par le récupérateur de mémoire (garbage collector). Il n'existe pas d'instruction delete comme en C++.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Objet null

L'objet null est utilisable partout. Le fait d'initialiser à null une variable référençant un objet permet au ramasse miette de libérer la mémoire allouée à
l'objet.

La variable this

Cette variable sert à référencer dans une méthode l'instance de l'objet en cours d'utilisation. This est un objet qui est égal à l'instance de l'objet dans lequel il est utilisé.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Les modificateurs d'accès

Ils se placent avant ou après le type de l'objet mais la convention veut qu'ils soient placés avant.
Ils s'appliquent aux classes et/ou aux méthodes et/ou aux attributs.

Ils ne peuvent pas être utilisés pour qualifier des variables locales : seules les variables d'instances et de classes peuvent en profiter.
Ils assurent le contrôle des conditions d'héritage, d'accès aux éléments et de modification de données par les autres objets.

Modificateur	Rôle
public	Une variable, méthode ou classe déclarée public est visible par tous les autres objets. Depuis la version 1.0, une seule classe public est permise par fichier et son nom doit correspondre à celui du fichier. Dans la philosophie orientée objet aucune donnée d'une classe ne devrait être déclarée publique : il est préférable d'écrire des méthodes pour la consulter et la modifier
par défaut : package friendly	Il n'existe pas de mot clé pour définir ce niveau, qui est le niveau par défaut lorsqu'aucun modificateur n'est précisé. Cette déclaration permet à une entité (classe, méthode ou variable) d'être visible par toutes les classes se trouvant dans le même package.
protected	Si une méthode ou une variable est déclarée protected , seules les méthodes présentes dans le même package que cette classe ou ses sous classes pourront y accéder. On ne peut pas qualifier une classe avec protected.
private	C'est le niveau de protection le plus fort. Les composants ne sont visibles qu'à l'intérieur de la classe : ils ne peuvent être modifiés que par des méthodes définies dans la classe prévues à cet effet. Les méthodes déclarées private ne peuvent pas être en même temps déclarées abstract car elles ne peuvent pas être redéfinies dans les classes filles.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Le mot clé static

Le mot clé static s'applique aux variables et aux méthodes. Les variables d'instance sont des variables propres à un objet.

Il est possible de définir une variable de classe qui est partagée entre toutes les instances d'une même classe : elle n'existe donc qu'une seule fois en mémoire.
Une telle variable permet de stocker une constante ou une valeur modifiée tour à tour par les instances de la classe. Elle se définit avec le mot clé static.

Le mot clé final

Le mot clé final s'applique aux variables de classe ou d'instance ou locales, aux méthodes, aux paramètres d'une méthode et aux classes.

Il permet de rendre l'entité sur laquelle il s'applique non modifiable une fois qu'elle est déclarée pour une méthode ou une classe et initialisée pour une variable.
Une variable qualifiée de final signifie que la valeur de la variable ne peut plus être modifiée une fois que celle-ci est initialisée.

Le mot clé abstract

Le mot clé abstract s'applique aux méthodes et aux classes**.** Abstract indique que la classe ne pourra être instanciée telle quelle. De plus, toutes les méthodes de cette classe abstract ne sont pas implémentées et devront être redéfinies par des méthodes complètes dans ses sous classes.

Une classe est automatiquement abstraite dès qu'une de ses méthodes est déclarée abstraite. Il est possible de définir une classe abstraite sans méthodes abstraites.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Les propriétés ou attributs

Les données d'une classe sont contenues dans des variables nommées propriétés ou attributs. Ce sont des variables qui peuvent être des variables d'instances, des variables de classes ou des constantes.

- **Les variables d'instances :** Une variable d'instance nécessite simplement une déclaration de la variable dans le corps de la classe.

- **Les variables de classes :** Les variables de classes sont définies avec le mot clé static. Chaque instance de la classe partage la même variable.

- **Les constantes :** Les constantes sont définies avec le mot clé final : leur valeur ne peut pas être modifiée une fois qu'elles sont initialisées.

Kabola

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Les méthodes

Les méthodes sont des fonctions qui implémentent les traitements de la classe.
La syntaxe de la déclaration d'une méthode est :

- modificateurs type_retourné nom_méthode (arg1, ...) {... }

Les modificateurs de méthodes sont :

Modificateur	Rôle
public	la méthode est accessible aux méthodes des autres classes
private	l'usage de la méthode est réservé aux autres méthodes de la même classe
protected	la méthode ne peut être invoquée que par des méthodes de la classe ou de ses sous classes
final	la méthode ne peut être modifiée (redéfinition lors de l'héritage interdite)
static	la méthode appartient simultanément à tous les objets de la classe (comme une constante déclarée à l'intérieur de la classe). Il est inutile d'instancier la classe pour appeler la méthode mais la méthode ne peut pas manipuler de variable d'instance. Elle ne peut utiliser que des variables de classes.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Les constructeurs

Le constructeur suit la définition des autres méthodes excepté que son nom doit obligatoirement correspondre à celui de la classe et qu'il n'est pas typé, pas même void, donc il ne peut pas y avoir d'instruction return dans un constructeur. On peut surcharger un constructeur.

La définition d'un constructeur est facultative. Si aucun constructeur n'est explicitement défini dans la classe, le compilateur va créer un constructeur par défaut sans argument.

Dès qu'un constructeur est explicitement défini, le compilateur considère que le programmeur prend en charge la création des constructeurs et que le mécanisme par défaut, qui correspond à un constructeur sans paramètres, n'est pas mis en oeuvre. Si on souhaite maintenir ce mécanisme, il faut définir explicitement un constructeur sans paramètres en plus des autres constructeurs.

Module 1 : CONCEPTS DE BASES

Section 2 : Programmation Orienté Objet partie 1

Les accesseurs

L'encapsulation permet de sécuriser l'accès aux données d'une classe. Ainsi, les données déclarées private à l'intérieur d'une classe ne peuvent être accédées et modifiées que par des méthodes définies dans la même classe.

Si une autre classe veut accéder aux données de la classe, l'opération n'est possible que par l'intermédiaire d'une méthode de la classe prévue à cet effet. Ces appels de méthodes sont appelés « échanges de messages ».

Un accesseur est une méthode publique qui donne l'accès à une variable d'instance privée. Pour une variable d'instance, il peut ne pas y avoir d'accesseur, un seul accesseur en lecture ou un accesseur en lecture et un autre en écriture.

Par convention, les accesseurs en lecture commencent par **get** et les accesseurs en écriture commencent par **set**.

Pour un attribut de type booléen, il est possible de faire commencer l'accesseur en lecture par **is** au lieu de **get**