Année 2008-2009

esil

Méthodes et variables statiques

Nicolas Baudru
mél : nicolas.baudru@esil.univmed.fr
page web : nicolas.baudru.perso.esil.univmed.fr

Supposons que vous voulez écrire une méthode int abs(int x) dans une classe Math qui calcule la valeur absolue d'un nombre x. Le comportement de cette méthode ne dépend pas de la valeur des variables d'instance de la class Math.

Pourquoi devrait-on créer un objet de la classe Math pour utiliser cette méthode ? Ce serait perdre de la place sur le tas.

La classe Math existe en Java, ainsi que la méthode abs(). Et effectivement il n'est pas possible d'instancier cette classe.

```
Math objetMath = new Math();
```

Cette instruction renvoie l'erreur suivante à la compilation :

```
%javac TestMath
TestMath.java:3: Math() has private
access in java.lang.Math
...
1 error
```

Méthodes statiques

Bien que Java soit un langage objet, il existe des cas où une instance de classe est inutile. Le mot clé static permet alors à une méthode de s'exécuter sans avoir à instancier la classe qui la contient. L'appel à une méthode statique se fait alors en utilisant le nom de la classe, plutôt que le nom de l'objet.

```
public class MaClassMath {
   ...
   public static int min( int a, int b) {
      // retourne la plus petite valeur
   }
}

public TestMaClassMath {
   public static void main( String []args){
      int x = MaClassMath.min(21,4);
   }
}
```

Aucun objet n'est créé !

- En général, une classe possédant des méthodes statiques n'est pas conçue pour être instanciée.

  Donnez deux méthodes pour empêcher l'instanciation d'une classe ?

- Mais cela ne signifie pas qu'une classe possédant une méthode statique ne doit jamais être instanciée : si une méthode de la classe n'est pas statique, alors l'instanciation doit être possible.

- Une méthode statique ne peut pas utiliser des variables d'instance non statiques.

Essayez de deviner pourquoi ?

```
public class Chien {
  private int taille;

  public static void aboyer() {
    if (taille < 20) System.out.println("kai␣kai");
  }
}
```

« taille » ne peut pas être utilisée dans la méthode aboyer() car aboyer() est statique.

- Une méthode statique ne peut pas utiliser de méthodes non statiques.

Essayez de deviner pourquoi ?

```
public class Chien {
  private int taille;

  public int getTaille() { return taille};

  public static void aboyer() {
    if (taille < 20) System.out.println("kai␣kai");
  }
}
```

Ce code ne compile pas car getTaille() ne peut pas être utilisée ici (c'est une méthode non statique utilisée dans une méthode statique).

Peut-on appeler une méthode non statique A
à partir d'une méthode statique B si A n'utilise pas de variables d'instance ?

Le code suivant est-il correct ?

```
public class Chien {
  private int taille;

  public static void aboyer() {
    if (taille < 20) System.out.println("kai␣kai");
  }
}

// quelque part ailleurs :
Chien c = new Chien();
c.aboyer();
```

Une variable d'instance statique est partagée par toutes les classes.

Comparez les deux implémentations suivantes de la classe Chien :

```
public class Chien {
  private int nbChien;

  public Chien() {
    nbChien++;
  }
}
```

```
public class Chien {
  private static int nbChien;

  public Chien() {
    nbChien++;
  }
}
```

Initialisation d'une variable statique

Les variables d'instance statiques sont initialisées lorsque la classe est chargée.

C'est la JVM qui choisit le moment où la classe va être chargée. En général, le chargement intervient juste avant la création d'un objet de cette classe ou l'invocation d'une méthode statique.

Les règles utilisées pour l'initialisation des variables d'instance statiques sont les mêmes que pour les variables d'instance non statiques.

Variables statiques et finales

Rappel :

- Une variable déclarée finale ne change plus de valeur une fois initialisée.
- Si elle est statique, alors elle est utilisable sans créer d'instance de la classe.
- Enfin, si elle est aussi publique, la variable est utilisable partout.

Par conséquent, une variable publique, statique et finale est une constante. Par convention, elle est notée en majuscule, un blanc souligné séparant les mots.

```
class MesConstantes {
  public static final double PI_APPROX = 3.1415;
}

// ailleurs dans le programme
int i = 2 * MesConstantes.PI_APPROX;
```

Il existe deux manières d'initialiser des variables statiques et finales.

- Au moment de la déclaration :

```
class MesConstantes {
  public static final int MA_CONST = 32;
}
```

- Dans un initialiseur « static » :

```
class MesConstantes {
  public static final int MA_CONST;

  static {
    MA_CONST = 16 * 2;
    }
}
```

Le code contenu dans « static » est exécuté lors du chargement de la classe.

Rappels sur l'utilisation de final

Que signifie final pour une variable d'instance ?

Que signifie final pour une variable locale ?

Que signifie final pour un paramètre ?

Que signifie final pour une méthode ?

Que signifie final pour une classe ?