# Méthodes et variables statiques

## 1 Les composantes statiques des classes

Nous avons vu qu'une classe peut contenir un constructeur, des variables et méthodes d'instance. Mais une classe peut aussi contenir de variables et méthodes qui ne sont pas d'instance, mais *statiques* ou *de classe*. Les déclarations de ces variables et méthodes contiennent le mot-clé `static`.

Nous utilisons régulièrement une méthode statique : le `main` qui constitue le programme.

Une méthode statique existe dans la classe indépendamment des objets instances de la classe. Alors que les méthodes d'instance n'existent que dans les objets créés en instanciant la classe avec un `new`. Si l'on reprend l'exemple des comptes bancaires, la méthode `get_solde` qui est une méthode d'instance ne peut être utilisée que s'il existe déjà un objet de type `Compte`. Elle peut alors être appelée et elle renvoie le solde de ce compte-là.

Une méthode statique comme `main` existe dans la classe et peut être utilisée sans qu'aucun objet instance de la classe ne soit créé. On peut l'utiliser en l'appelant non pas sur un objet mais sur la classe.

## 2 Exemples de composants statiques prédéfinis

Outre les méthodes `main` que nous avons écrites, il existe des méthodes statiques prédéfinies. C'est le cas de diverses fonctions mathématiques de la classe `Math`. Par exemple, la méthode `random` permet d'obtenir un nombre tiré au sort compris entre 0 et 1. On appelle cette méthode non pas sur un objet instance de la classe `Math` mais au moyen du nom de la classe : `Math.random()` est une expression qui renvoie un nombre tiré au sort.

Il y a d'autres méthodes dans `Math` : `min, max, sin, cos,` etc. Une autre méthode assez fréquement employée est la méthode `parseInt` de la classe `Integer` qui sert à convertir une chaîne de caractère en un entier. Par exemple `Integer.parseInt("123")` est une expression qui renvoie l'entier 123 et `Integer.parseInt("abc123")` lève l'exception `NumberFormatException`.

Il existe également des variables statiques auxquelles on accède en utilisant le nom de la classe. La classe `Math` contient une variable `PI` qui est une valeur de type `double` approchant le nombre $\pi$ utilisé pour les calculs d'aires de cercles et la trigonométrie. Deux autres variables statiques sont également d'usage courant : les méthodes `in` et `out` de la classe `System`. Ces variables contiennent des objets Java représentant au sein du programme respectivement le clavier et l'écran de l'ordinateur.

## 3 Définition des composants statiques

Pour définir une méthode ou une variable statique dans une classe que l'on écrit, il faut ajouter le mot-clé `static` après le mot-clé `public` ou le mot-clé `private`.

Il y a une contrainte sur les méthodes statiques : ces méthodes ne peuvent pas utiliser les variables et méthodes d'instance. Le message d'erreur du compilateur si l'on viole cette contrainte est le suivant :

```
non-static variable solde cannot be referenced from a static context
```

La contrainte est parfaitement logique. Imaginez qu'on ajoute une méthode statique dans la classe `Compte` et que celle-ci veuille accéder à la variable solde. D'une part la méthode peut être appelée avant qu'existe un objet `Compte` contenant un solde et donc il n'y a pas de variable `solde` à ce moment-là, mais elle peut également être appelée alors que plusieurs objets `Compte` ont été créés qui ont chacun leur variable `solde`. Quel solde doit alors utiliser la méthode ?

# 4 A quoi servent les composantes statiques ?

Les méthodes statiques servent à écrire des méthodes qui ne dépendent pas des données spécifiques à un individu. Par exemple la fonction cosinus ne dépend pas d'autre chose que de son paramètre. Quant aux variables, elles servent pour les valeurs qui n'existent qu'en un exemplaire et sont les mêmes pour tous les individus, toutes les instances d'une classe.

Prenons des exemples : si l'on veut compter le nombre d'objets instance d'une classe au moyen d'un compteur incrémenté à chaque appel du constructeur, il y a bien un seul nombre d'objets et non pas un nombre différent dans chaque objet.

Autre exemple : si dans la classe `Compte`, on veut attribuer des numéros de compte différents à chaque compte bancaire, il faut une variable qui contient soit le dernier numéro attribué, soit le prochain à attribuer. Cette information doit exister en un seul exemplaire dans tout le programme, faute de quoi on ne peut pas assurer que le même numéro ne sera pas utilisé deux fois.

Voici un extrait du code de la classe Compte modifiée pour attribuer des numéros de compte uniques. Cette version utilise une variable statique contenant le prochain numéro à attribuer dans une variable statique.

```
public class Compte{
    private String titulaire;
    private int numero;
    private double solde;
    private static int prochain_numero = 1;
    public Compte(String tit){
        titulaire = tit;
        numero = Compte.prochain_numero;
        Compte.prochain_numero = Compte.prochain_numero+1;
        solde = 0.0;
    }
    ...
}
```

Si l'on exécute la méthode `main` suivante, on obtient le résultat transcrit ci-dessous.

```
    public static void main(String[] args){
        Compte c1 = new Compte("riri");
        Compte c2 = new Compte("fifi");
        Compte c3 = new Compte("loulou");
        System.out.println(c1);
        System.out.println(c2);
```

```
        System.out.println(c3);
    }
```

```
> java Compte
Titulaire: riri numéro: 1 solde: 0.0
Titulaire: fifi numéro: 2 solde: 0.0
Titulaire: loulou numéro: 3 solde: 0.0
```

# 5 Créer un objet avec une méthode statique

Il est tout à fait déconseillé de faire des entrées-sorties écran/clavier dans le constructeur d'une classe. Si l'on souhaite construire un objet de la classe en utilisant des données saisies au clavier, on conseille de séparer le travail en deux parties : le constructeur reçoit les valeurs via les paramètres. La saisie des valeurs au clavier et l'appel du constructeur peuvent être faits par une méthode statique qui renvoie le nouvel objet créé.

```
import java.util.Scanner;
public class Compte{
    private String titulaire;
    private int numero;
    private double solde;
    private static int prochain_numero = 1;
    private static Scanner scan;
    ...
    public static Compte creerCompte(){
        String nom;
        double apport;
        Compte resultat;
        if (Compte.scan == null){
            Compte.scan = new Scanner(System.in);
        }
        System.out.print("Entrez␣le␣nom␣du␣titulaire:␣");
        nom = Compte.scan.nextLine();
        System.out.print("Entrez␣le␣versement␣initial:␣");
        apport = Compte.scan.nextDouble();
        Compte.scan.nextLine();
        resultat = new Compte(nom);
        resultat.deposer(apport);
        return resultat;
    }
    public static void main(String[] args){
        Compte c1 = creerCompte();
        Compte c2 = creerCompte();
        System.out.println(c1);
        System.out.println(c2);
    }
}
```

Notez également comment l'on utilise une variable statique pour éviter de recréer un scanner à chaque appel à la méthode : un scanner est créé la première fois que la méthode est appelée, parce

que la variable `scan` contient la valeur `null`. Ensuite, ce scanner est réutilisé à chauqe appel de la méthode.

L'exécution de ce programme produit l'affichage suivant :

```
> java Compte
Entrez le nom du titulaire: riri
Entrez le versement initial: 100
Entrez le nom du titulaire: fifi
Entrez le versement initial: 555
Titulaire: riri numéro: 1 solde: 100.0
Titulaire: fifi numéro: 2 solde: 555.0
```

# 6 Les variables statiques dans la mémoire

Les objets sont stockés dans le tas et contiennent les variables d'instance. En revanche les variables statiques sont stockées dans la mémoire des classes. Elles existent et sont initialisées au chargement de la classe qui les contient dans la mémoire. Les méthodes qu'elles soient statiques ou d'instance sont également dans la mémoire des classes.

Conceptuellement, les méthodes d'instance pourraient être stockées dans les objets, mais comme elles sont identiques pour tous les objets d'une classe, il est inutile de les dupliquer. Les objets contiennent donc le type d'instance qui est le nom de la classe utilisée pour les créer et c'est dans cette classe, dans la mémoire des classes, que les méthodes d'instance sont recherchées.