DU ISN - Quelques éléments de récursivité

Guillaume Huard

Retour sur les méthodes

Méthode

Partie nommée d'un programme ($\simeq$ fonction), on peut

- l'appeler (exécuter son contenu)
- lui passer des paramètres (valeurs sur lesquelles travailler)
- récupérer sa valeur de retour (résultat)

Exemples

```
static void hello() {
        System.out.println("Hello world");
}

static int addition(int a, int b) {
        return a+b;
}
```

En l'absence d'objets, elle est toujours `static`

Cas particulier

La méthode `main` d'une classe est un cas particulier

- c'est par elle que démarre l'exécution
- on l'appelle aussi programme principal
- sa déclaration doit suivre un format imposé par le système

```
class Exemple {
      public static void main(String [] args) {
            hello();
      }
}
```

Nous reviendrons sur la notion de classe

Passage de paramètres

Lors de l'appel d'une méthode, java

- réserve assez d'espace en mémoire pour stocker tous les paramètres formels
- copie la valeur des paramètres effectifs passés lors de l'appel dans les paramètres formels

On appelle cela le passage de paramètres par valeur

Exemple d'une fonction qui ne fait rien

```
static void echange(int a, int b) {
        int tmp;
        tmp = a;
        a = b;
        b = tmp;
}
```

Récursivité

Méthode récursive

Méthode qui s'appelle elle-même

- directement ou via des appels en cascade
- tire parti du passage de paramètres par valeur : un appel travaille sur ses paramètres, n'a pas d'incidence sur l'appelant

```
static void afficheRec(int n) {
        if (n > 0) {
                afficheRec(n-1);
                System.out.print(" " + n);
        }
}
static void affiche() {
        System.out.print("Nombres :");
        afficheRec(10);
        System.out.println();
}
```

Pourquoi écrire des méthodes récursives ?

Forme le raisonnement

- pousse à décomposer le problème
- force à abstraire

Simplifie énormément l'écriture de certains programmes

- problèmes se décomposant naturellement en sous problèmes
- travail sur des structures de données comme les arbres

On peut souvent écrire un équivalent itératif

- plus compliqué
- en s'aidant de structures de données auxiliaires

Points importants

Décomposer le problème à résoudre

- l'exprimer sous forme de problèmes plus petits
- avoir foi dans le fait que les appels récursifs sont corrects

Ne pas oublier le cas de base

- les appels en cascade doivent s'arrêter
- une méthode récursive débute généralement par un test
- le sous problème le plus élémentaire s'appelle cas de base, il est résolu directement

En cas d'oubli, on tombe sur une erreur mémoire (trop d'appels)

Quelques fonctions

Factorielle

$$factorielle(0) = 1$$
$$factorielle(n) = n \times factorielle(n-1)$$

Fibonacci

$$fibonacci(0) = 0$$
$$fibonacci(1) = 1$$
$$fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)$$

Triangle de Pascal

$$pascal(n, 0) = 1$$
$$pascal(n, n) = 1$$
$$0 < j < i, pascal(i, j) = pascal(i-1, j-1) + pascal(i-1, j)$$

Quelques problèmes

- Plus grand diviseur commun
  Soit a et b deux entiers tels que $a \geq b$ (les échanger sinon)
  Le plus grand diviseur commun à a et b est
  - a si $b = 0$
  - le plus grand diviseur commun à $a - b$ et b sinon

- Tours de Hanoi
  N disques percés de diamètre croisant (de 1 à N) sont empilés sur l'un des trois piquets du jeu. Problème : déplacer tous les disques vers un autre piquet sachant que
  - on ne peut déplacer qu'un disque à la fois
  - interdits d'empiler un disque sur un autre de diamètre inférieur