MOOC Init Prog Java
Tutoriels semaine 6

Les tutoriels sont des exercices qui reprennent des exemples du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même.

Ils sont conseillés comme un premier exercice sur un sujet que l'étudiant ne pense pas encore assez maîtriser pour aborder par lui-même un exercice «classique».

Semaine 6 : Encore des moyennes

Le but ici est de résoudre un petit exercice illustrant l'utilisation des méthodes (fonctions) .

Commencez comme d'habitude par ouvrir un nouveau fichier, par exemple ici `Moyenne.java` dans votre éditeur/IDE favori (par exemple Geany ou Eclipse), puis préparez la «coquille vide» de base de votre programme :

```
class Moyenne {
    public static void main(String[] args) {

    }
}
```

Écrivez maintenant dans la méthode `main`, les instructions nécessaires pour demander deux notes à l'utilisateur et afficher leur moyenne (pour commencer sans utiliser de méthode).

Essayez de le faire par vous même sans regarder la solution ci-dessous

(solution page suivante)

Solution :

La façon que nous avions jusqu'ici de réaliser un tel programme ressemble à peu près à ceci :

```
private static Scanner scanner = new Scanner(System.in);
public static void main(String[] args) {
    double note1, note2;
    System.out.println("Entrez vos deux notes :");
    note1 = scanner.nextDouble();
    note2 = scanner.nextDouble();
    System.out.println("Votre moyenne est : " + (note1 + note2)/2.0);
}
```

Vous avez peut-être utilisé des `int` au lieu des `double` ici. C'est très bien : cela va aussi.
Cela signifie juste que pour vous une note est un nombre entier, alors que pour moi elle peut être fractionnaire.

Notez que les parenthèses lors du calcul de la moyenne sont obligatoires puisque la division est prioritaire sur l'addition.

De plus, dans le cas où les notes sont des entiers, il est recommandé de diviser par 2.0 plutôt que par 2, pour assurer que le résultat soit un double et non pas un entier !
Revoyez le cours si vous ne comprenez pas cette remarque.

Continuons l'exercice pour maintenant introduire des méthodes auxiliaires qui nous aiderons à modulariser notre méthode `main`.

Imaginons que ce programme rudimentaire soit amené à grandir et que le calcul de la moyenne soit nécessaire à plusieurs endroits différents du programme.

Il peut alors être judicieux de sortir le calcul de la moyenne du corps de la méthode `main`, de manière à ce qu'il puisse être **réutilisé** par d'autres parties du programme.

Nous voulons donc définir une **méthode auxiliaire** qui prenne deux `doubles` comme paramètres, et retourne le résultat sous forme de `double`.

Quel est alors l'entête de cette méthode ?
(sans regarder `;-)`)

(solution page suivante)

Solution :

L'entête de cette méthode est le suivant :

```
static double moyenne(double nombre1, double nombre2) {
}
```

Ecrivons maintenant le corps de la méthode calcule la moyenne des paramètres, et retourne le résultat du calcul.

(solution page suivante)

Solution :

```
// solution 1
static double moyenne(double nombre1, double nombre2) {
    // declaration et initialisation d'une variable
    double moy = (nombre1 + nombre2) / 2.0;

    // la méthode renvoie la valeur de la variable moy
    return moy;
}

// solution 2
static double moyenne(double nombre1, double nombre2) {
    // la fonction renvoie directement la valeur
    // il n'est pas nécessaire d'utiliser une variable intermédiaire
    return (nombre1 + nombre2) / 2.0;
}
```

Bien que les deux solutions ci-dessous soit équivalentes, nous préférerons la deuxième qui est plus concise et élégante.

Il nous reste plus maintenant qu'à assembler les morceaux, c'est-à-dire faire **appel** à la fonction dans la méthode principale `main`.

(solution page suivante)

Solution :

Cela se fait simplement en mettant le nom de la fonction à appeler avec les paramètres voulus à l'endroit où l'on veut utiliser la valeur qu'elle retourne. Par exemple ici :

```
        System.out.println("Votre moyenne est : " + moyenne(note1, note2));
```

Notez que la définition de notre méthode auxilaire peut se trouver n'importe où (avant ou après la méthode ou les méthodes l'utilisant).

On arrive donc au résultat :

```
import java.util.Scanner;
class Moyenne {
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        double note1, note2;
        System.out.println("Entrez vos deux notes :");
        note1 = scanner.nextDouble();
        note2 = scanner.nextDouble();

        // appel
        System.out.println("Votre moyenne est : " + moyenne(note1, note2));
    }

    // définition
    static double moyenne(double nombre1, double nombre2) {
        return (nombre1 + nombre2) / 2.0;
    }
}
```

Semaine 6 : 2. Somme récursives

On veut écrire le programme `Somme.java` qui calcule de façon récursive la somme des *n* premiers entiers positifs.

«*De façon récursive*» signifie que le calcul de la somme pour *n* fait appel au calcul de la somme pour des nombres plus petits que *n*. Dans ce cas précis (somme), on utilisera le fait que la somme des *n* premiers entiers vaut celles des *n-1* premiers entiers, à laquelle on ajoute *n*. Mathématiquement, on noterait : $S_n = S_{n-1} + n$

Cet exercice reprend pas à pas les différentes étapes pour y parvenir.

1. Ouvrez un fichier vide `Somme.java` dans votre éditeur favori.
2. Préparez la «coquille vide» de base accueillant votre programme :

```
class Somme{
    public static void main(String[] args) {

    }
}
```

3. Définissez la méthode récursive `somme`, auxiliaire de `main`.

   Pour formuler notre méthode récursive, il suffit de constater que la somme des n premier nombres (de 1 à n), vaut `n` plus la somme des n-1 premiers nombre(de 1 à n-1)

   (solution page suivante)

Solution :

```
// définition de la méthode somme
static int somme(int n) {
    return (n + somme(n-1));
}
```

Evidemment, formulé de cette manière notre code a de forte chance de partir à la conquête de l'espace (ou plus précisément de ne s'arrêter que lorsqu'il aura atteint la limite de récursion de la machine virtuelle Java).

Il est donc impératif de lui indiquer quand il doit s'arrêter. Cette situation correspond généralement au cas de résolution trivial. Essayez de le formuler par vous-même.

```
// définition de la méthode somme
static int somme(int n) {
    if (n == 0) { // Condition d'arrêt
        return 0;
    } else {
        return (n + somme(n-1));
    }
}
```

---

4. la méthode `main` doit ensuite être complétée de façon appropriée pour faire apppel à notre méthode.

   (solution page suivante)

```java
import java.util.Scanner;
class Somme {
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args){
        int n;
        do {
            System.out.println("Donnez un entier positif:");
            n = scanner.nextInt();
        } while(n < 0);
        System.out.println(" La somme est " + somme(n));
    }

    static int somme(int n){
        if (n == 0) {
            return n;
        } else {
            return (n+somme(n-1));
        }
    }
}
```
