Haute École de Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

2018 – 2019
DEV1

DEV1 – Laboratoires Java I

TD 6 – Méthodes

Ce TD introduit les notions de méthode, entête et déclaration, paramètre, et valeur de retour, l'instruction `return` ainsi que le type `boolean`.

Les codes sources et les solutions de ce TD se trouvent à l'adresse :

`https://git.esi-bru.be/dev1/labo-java/tree/master/td06-méthodes`

Table des matières

Version du document
3 septembre 2018

1 Introduction

Une méthode est une construction qui permet de décomposer une solution sous forme de petit module. Dans l'exemple ci-dessous la méthode `périmètre` de la classe `Cercle` calcule le périmètre d'un cercle étant donné son rayon.

```java
package esi.dev1.td6;

public class Cercle {

   static double périmètre(double rayon) {
      return 2 * 3.141596 * rayon ;
   }

   public static void main(String[] args) {
      System.out.println("Le périmètre d'un cercle de rayon 10 est: "+ périmètre(10));
   }
}
```

Cercle.java

L'entête[1] de cette méthode, `static double périmètre(double rayon)`, nous signale que :

- ▷ cette méthode s'appelle `périmètre` ;
- ▷ elle reçoit en paramètre un `double` qui sera manipulé par son nom `rayon` ;
- ▷ elle retourne une valeur de type `double`.

Nous verrons ultérieurement la signification du mot clef `static`, qui doit toujours se trouver au début de l'entête des méthodes que vous écrivez. Une méthode non-static a un tout autre sens que nous verrons en dev2.

Le *corps* de la méthode est ici très simple, il *retourne* (à l'aide de l'instruction `return`) la valeur reçue en paramètre (le rayon) multiplié par 2*3.141596. Le type de la valeur retournée doit être compatible avec le type déclaré dans l'entête, ici cela doit être un `double`.

La méthode principale *appelle* cette méthode avec la valeur 10 comme rayon.

Exercice 1 Périmètre et aire d'un cercle

Créer un package `g12345.dev1.td5` et, dans celui-ci, une classe `MathUtil`. Dans la classe `MathUtil` écrivez les 2 méthodes suivantes :

- ▷ `double périmètreCercle(double rayon)` qui reçoit le rayon (un double) en paramètre et retourne le périmètre du cercle.
- ▷ `double aireCercle(double rayon)` qui reçoit un rayon en paramètre et retourne l'aire du cercle.

Rappel : l'aire d'un cercle se calcule par la formule : πr^2 ou r est le rayon du cercle.

Attention : n'oubliez pas le mot-clef `static` au début de l'entête de vos méthodes.

Dans la méthode principale testez ces 2 méthodes comme cela est fait dans l'exemple ci-dessus avec la méthode `périmètre`.

1. aussi appelée *déclaration* de la méthode.

2 Appel d'une méthode

Dans l'exemple ci-dessus la méthode principale fait appel à la méthode périmètre en utilisant le nom et en fournissant une valeur pour le paramètre : `périmètre(10)`.

Il est possible de faire appel à une méthode depuis une *autre classe*.

```java
package esi.dev1.td6;

public class CercleApp {

   public static void main(String[] args) {
      System.out.println("Le périmètre d'un cercle de rayon 10 est: "+
                         Cercle.périmètre(10));
   }
}
```

CercleApp.java

La méthode principale de la classe `CercleApp` fait appel à la méthode `périmètre` de la classe `Cercle` en utilisant le nom de la classe suivi d'un point et du nom de la méthode : `Cercle.périmètre(10)`.

Exercice 2 **Périmètre et aire d'un cercle**

Dans une classe `CercleApp` écrivez un programme (et donc une méthode `main`) qui demande à l'utilisateur le rayon d'un cercle et affiche son périmètre et son aire.

Votre programme fera appel aux méthodes `périmètreCercle` et `aireCercle` de la classe `MathUtil` écrites précédemment.

3 Méthodes à plusieurs paramètres

```java
package esi.dev1.td6;

public class Maximum {

   static int max2(int a, int b) {
      int max = a;
      if(a < b) {
         max = b;
      }
      return max;
   }

   static int max3(int a, int b, int c) {
      int max = max2(a, b);
      max = max2(max, c);
      return max;
   }

   public static void main(String[] args) {
      System.out.println(max2(10, 6));
      System.out.println(max3(10, 6, 19));
   }
}
```

Maximum.java

Dans la classe `Maximum` ci-dessus sont définies 2 méthodes : `max2` et `max3`. Dans l'entête de la méthode `max2` on constate que cette méthode

- ▷ s'appelle `max2` ;

- ▷ reçoit *en paramètres* deux entiers, ces entiers seront manipulés grâce à leur nom `a` et `b` ;
- ▷ *retourne* un entier.

La méthode principale fait un appel à la méthode `max2` avec les paramètres effectifs 10 et 6 et affiche la valeur retournée par la méthode max2 : 10.

Dans l'entête de la méthode `max3` on constate que cette méthode

- ▷ s'appelle `max3` ;
- ▷ reçoit *en paramètres* trois entiers, ces entiers seront manipulés grâce à leur nom `a`, `b` et `c` ;
- ▷ *retourne* un entier.

Cette méthode fait appel par deux fois à la méthode `max2`.

La méthode principale fait un appel à la méthode `max3` avec les paramètres effectifs 10, 6 et 19 et affiche la valeur retournée : 19.

Exercice 3 Minimum

Dans la classe `MathUtil` écrivez les 2 méthodes suivantes :

- ▷ `double min2(double x, double y)` qui reçoit deux doubles en paramètres et retourne le minimum.
- ▷ `double min3(double x, double y, double z)` qui reçoit trois doubles en paramètres et retourne le minimum.

Dans la méthode principale testez ces 2 méthodes comme cela est fait dans l'exemple ci-dessus avec les fonctions `max2` et `max3`.

Exercice 4 Moyenne

Dans la classe `MathUtil` ajoutez la méthode `double moyenne(double x, double y)` qui reçoit deux doubles en paramètres et retourne leur moyenne.

Par exemple si la méthode reçoit 10.5 et 15.5 elle retourne 13.

Testez-la dans la méthode principale.

4 Méthodes et chaînes de caractères

Les méthodes peuvent traiter n'importe quel type de données. Les méthodes de cette section traitent des chaînes de caractères.

```
1  package esi.dev1.td6;
2
3  public class Mot {
4
5     static char premièreLettre(String mot) {
6        return mot.charAt(0);
7     }
8
9     public static void main(String[] args) {
10       System.out.println(premièreLettre("Java"));
11       System.out.println(premièreLettre("Programmation"));
12    }
13 }
```

Mot.java

La méthode `premièreLettre` reçoit en paramètre une chaine de caractères et retourne la première lettre de ce mot.

Exercice 5 Première et dernière lettre

Créez une classe `ChaineUtil` et ajoutez-y les méthodes suivantes :

- `char premièreLettre(String mot)` qui reçoit une chaîne de caractères en paramètre et retourne sa première lettre.

  Par exemple si la méthode reçoit "Java" elle retournera 'J'.

- `char dernièreLettre(String mot)` qui reçoit une chaîne de caractères en paramètre et retourne sa dernière lettre.

  Par exemple si la méthode reçoit "Java" elle retournera 'a'.

Testez-les dans la méthode principale.

5 Méthodes et booléens

Nous avons vu que les *booléens* sont les deux valeurs `true` (vrai) et `false` (faux). Le type associé aux booléens est `boolean`.

Une méthode retournant vrai ou faux sera de type `boolean` comme illustré dans l'exemple suivant.

```java
package esi.dev1.td6;

public class Pair {

   static boolean estPair(int nb) {
      return (nb%2) == 0;
   }

   public static void main(String[] args) {
      if(estPair(10)) {
         System.out.println("10 est pair");
      } else {
         System.out.println("10 est impair");
      }
   }
}
```

Pair.java

La méthode `estPair` retourne la valeur de l'expression `(nb%2)==0`, cette expression est vraie si `nb` est pair et faux sinon.

La condition de l'instruction `if`, `estPair(10)`, sera vraie si l'appel à la méthode retourne vrai (et donc si `10` est pair) et faux sinon.

Exercice 6 Divisible

Dans la classe `MathUtil` ajoutez la méthode `boolean estDivisible(int a, int b)` qui reçoit deux entiers en paramètres et retourne vrai si le premier est divisible par le second.

Par exemple si la méthode reçoit 10 et 5 elle retourne `true`, car 10 est divisible par 5.

Rappel : a est divisible par b si l'expression `a%b==0` est vraie.

Testez-la dans la méthode principale.

6 Méthodes sans retour

Certaines méthodes ne retournent rien, par exemple lorsque la méthode affiche quelque chose sur la sortie standard.

Dans ce cas l'entête de la méthode le signale avec le type de retour `void` mot anglais qui se traduit par " vide " .

```java
package esi.dev1.td6;

public class Ligne {

   static void afficherLigne(int longueur) {
      for(int i=0; i<longueur; i++) {
         System.out.print('-');
      }
      System.out.println(); //on passe à la ligne.

   }

   public static void main(String[] args) {
      afficherLigne(10);
      afficherLigne(20);
   }
}
```

Ligne.java

7 Exercices récapitulatifs

Exercice 7 Valeur absolue

Dans la classe `MathUtil` ajoutez la méthode `abs` qui reçoit un entier en paramètre et retourne sa valeur absolue.

Par exemple si la méthode reçoit -4 elle retourne 4, et si elle reçoit 10 elle retourne 10.

Testez-la dans la méthode principale avec une valeur négative et ensuite une valeur positive.

Exercice 8 Décomposition, recomposition

Dans la classe `MathUtil` ajoutez les méthodes suivantes :

- ▷ `int unité(int nb)` qui reçoit un entier en paramètre et retourne la valeur des unités de cet entier. Par exemple si la méthode reçoit 123 elle retournera 3.
- ▷ `int dizaine(int nb)` qui reçoit un entier en paramètre et retourne la valeur des dizaines de cet entier. Par exemple si la méthode reçoit 123 elle retournera 2.
- ▷ `int centaine(int nb)` qui reçoit un entier en paramètre et retourne la valeur des centaines de cet entier. Par exemple si la méthode reçoit 123 elle retournera 1.
- ▷ `int miroir(int nb)` qui reçoit un nombre compris entre 100 et 999 et retourne son miroir. Par exemple si la méthode reçoit 123 elle retournera 321.

  Astuce : utilisez judicieusement les méthodes `unité`, `dizaine` et `centaine`.

Testez-les dans la méthode principale.

Exercice 9 Voyelles et consonnes

Dans la classe `ChaineUtil` ajoutez les méthodes suivantes :

- ▷ `int nombreVoyelles(String mot)` qui reçoit une chaîne de caractères en paramètre et retourne le nombre de voyelles de ce mot.

  Par exemple si la méthode reçoit "Programmation" elle retournera 5.
- ▷ `int nombreConsonnes(String mot)` qui reçoit une chaîne de caractères en paramètre et retourne le nombre de consonnes de ce mot.

  Par exemple si la méthode reçoit "Programmation" elle retournera 8.

Testez-les dans la méthode principale.

Exercice 10 Palindrome

Dans la classe `ChaineUtil` ajoutez la méthode `boolean estPalindrome(String mot)` qui reçoit une chaîne de caractères en paramètre et retourne vrai si cette chaîne est un palindrome.

Par exemple si la méthode reçoit "été" elle retourne `true`.

Testez-la dans la méthode principale avec un palindrome et ensuite avec un mot qui n'en est pas un.