Les Fonctions & Procédures
(Les méthodes)

Définition

1. Une méthode est un bloc de code qui ne s'exécute que lorsqu'elle est appelée. Les méthodes sont utilisées pour effectuer certaines actions, et elles sont également appelées des fonctions.
2. Les méthodes nous permettent de définissez le code une seule fois et utilisez-le plusieurs fois.
3. Les méthodes ne sont pas limitées en nombre de paramètres ;
4. Il en existe trois grands types :
    - Les méthodes qui ne renvoient rien. Les méthodes de ce type n'ont pas d'instruction return, et elles sont de type void (procédure) ;
    - Les méthodes qui retournent des types primitifs (double, int etc.). Elles sont de type double, int, char etc. Celles-ci possèdent une instruction return (fonction) ;
    - Les méthodes qui retournent des objets. Par exemple, une méthode qui retourne un objet de type String. Celles-ci aussi comportent une instruction return.

Emplacement des méthodes

*Sdz1.java

```
import java.util.Scanner;


public class Sdz1 {

    /**
     * @param args
     */
    public static void main(String[] args) {
        /**
         * Méthode main
         */
    }

    public static double arrondi(double A, int B) {
        return (double) ( (int) (A * Math.pow(10, B) + .5)) / Math.pow(10, B);
    }
}
```

Si vous placez une de vos méthodes à l'intérieur de la méthode main ou à l'extérieur de votre classe, le programme ne compilera pas.

Comment appeler une méthode ?

Pour appeler une méthode en Java, il suffit d'écrire le nom de la méthode suivi de parenthèses () et d'un point-virgule ;

```
public class MaClasse {
      static void message() {
           System.out.println("Bonjour à Tous!");
            }
      public static void main(String[] args) {
           message();
      }
}
```

- **public** : modificateur d'accès, c'est à dire que la méthode est accessible en d'hors de la classe.
- **static** : signifie que la méthode appartient à la classe MaClasse et non à un objet de la classe MaClasse.
- **void :** signifie que cette méthode ne renvoie rien.
- **message** : le nom de la méthode.

Surcharge des méthodes

La surcharge permet à différentes méthodes d'avoir le même nom, mais des signatures différentes où la signature peut différer en fonction du nombre de paramètres d'entrée, du type de paramètres d'entrée ou des deux.

```
public class Test {


    // Cette méthode prend deux paramètres int
    public int somme(int x, int y) {
        return (x + y);
    }
    // Cette méthode prend trois paramètres int
    public int somme(int x, int y, int z) {
        return (x + y + z);
    }
    // Cette méthode prend deux paramètres double
    public double somme(double x, double y) {
        return (x + y);
    }
    public static void main(String args[]) {
        Test t = new Test();
```

```
        System.out.println(t.somme(10, 20));
        System.out.println(t.somme(10, 20, 30));
        System.out.println(t.somme(10.5, 20.5));
    }
}
```

Des méthodes concernant les chaînes de caractères

La méthode toLowerCase() permet de transformer tout caractère alphabétique en son équivalent minuscule. Elle n'a aucun effet sur les chiffres.

```
String chaine = new String("COUCOU TOUT LE MONDE !"), chaine2 = new
String();
chaine2 = chaine.toLowerCase();   //Donne "coucou tout le monde !"
```

La méthode toUpperCase() est simple, puisqu'il s'agit de l'opposé de la précédente. Elle transforme donc une chaîne de caractères en capitales.

```
String chaine = new String("coucou coucou"), chaine2 = new String();
chaine2 = chaine.toUpperCase();   //Donne "COUCOU COUCOU"
```

La méthode length() renvoie la longueur d'une chaîne de caractères (en comptant les espaces).

```
String chaine = new String("coucou ! ");
int longueur = 0;
longueur = chaine.length();   //Renvoie 9
```

La méthode equals() permet de vérifier (donc de tester) si deux chaînes de caractères sont identiques.

```
String str1 = new String("coucou"), str2 = new String("toutou");

if (str1.equals(str2))
 System.out.println("Les deux chaînes sont identiques !");

else
 System.out.println("Les deux chaînes sont différentes !");
```

Pour la vérification de l'inégalité on utilise l'opérateur de négation. Il s'agit de « ! ».

```
String str1 = new String("coucou"), str2 = new String("toutou");

if (!str1.equals(str2))
 System.out.println("Les deux chaînes sont différentes !");

else
 System.out.println("Les deux chaînes sont identiques !");
```

La méthode charAt() renvoie un caractère : il s'agit d'une méthode d'extraction de caractère. Elle ne peut s'opérer que sur des String. Cette méthode prend un entier comme argument.

```
String nbre = new String("1234567");
char carac = nbre.charAt(4);   //Renverra ici le caractère 5
```

La méthode substring() extrait une partie d'une chaîne de caractères. Elle prend deux entiers en arguments : le premier définit le premier caractère (inclus) de la sous-chaîne à extraire, le second correspond au dernier caractère (exclu) à extraire. Le premier caractère porte le numéro 0.

```
String chaine = new String("la paix niche"), chaine2 = new String();
chaine2 = chaine.substring(3,13);   //Permet d'extraire "paix niche"
```

La méthode indexOf() explore une chaîne de caractères à la recherche d'une suite donnée de caractères, et renvoie la position (ou l'index) de la sous-chaîne passée en argument. La méthode lastIndexOf() explore en partant de la fin, mais renvoie l'index à partir du début de la chaîne. Ces deux méthodes prennent un caractère ou une chaîne de caractères comme argument, et renvoient un int.

```
String mot = new String("anticonstitutionnellement");
int n = 0;
n = mot.indexOf('t');         //n vaut 2
n = mot.lastIndexOf('t');     //n vaut 24
n = mot.indexOf("ti");        //n vaut 2
n = mot.lastIndexOf("ti");    //n vaut 12
n = mot.indexOf('x');         //n vaut -1
```

Les méthodes de la classe Math, présentent dans java.lang. Elle fait donc partie des fondements du langage. Par conséquent, aucun import particulier n'est nécessaire pour utiliser la classe Math qui regorge de méthodes utiles :

```
double X = 0.0;
X = Math.random();
//Retourne un nombre aléatoire
//compris entre 0 et 1, comme 0.0001385746329371058
```

```
double abs = Math.abs(-120.25); //La fonction valeur absolue (retourne le nombre sans le signe)
double d = 2;
double exp = Math.pow(d, 2);    //La fonction exposant
//Ici, on initialise la variable exp avec la valeur de d élevée au carré
//La méthode pow() prend donc une valeur en premier paramètre, et un exposant en second
```

Ces méthodes retournent toutes un nombre de type double.

En résumé

- Une méthode est un morceau de code réutilisable qui effectue une action bien définie.
- Les méthodes se définissent dans une classe.
- Les méthodes ne peuvent pas être imbriquées. Elles sont déclarées les unes après les autres.