

Programme

- Introduction à la programmation informatique et aux différents langages de programmation.
- Présentation des concepts de base de la programmation de haut niveau. Aspect algorithmique. Introduction au langage Java.
- Syntaxe de base du langage Java, la sémantique et le concept de la compilation.
- Les variables, types et opérations mathématiques
- Opérateurs, affectation,
- conditionnelles et boucles (1 et 2)
- Tableaux (accès aux composants, ...).
- **Procédures et Fonctions.**
- Présentations de quelques classes clés.
- Des algorithmes fondamentaux.
- Des exemples applicatifs de l'ensemble des notions de Java présentées.

Aujourd'hui

Procédures et Fonctions

- Introduction
- Créer sa propre méthode
- Quelques méthodes utiles

Introduction

- Il y a certaines choses que nous effectuons souvent dans un programme: recopier sans arrêt les mêmes morceaux de code.

Sinon: écrire une méthode... (« fonction » ou « procédure » dans d'autres langages).

- Il existe énormément de méthodes dans le langage Java, présentes dans des objets;

- On peut les catégoriser en deux « familles » : les natives et les vôtres.

Jusque-là, nous n'avons écrit que des programmes comportant une seule classe, ne disposant elle-même que d'une méthode : la méthode main. Le moment est donc venu d'utiliser des méthodes existantes et de créer vos propres méthodes.

Quelques méthodes utiles (1/12)

Des méthodes concernant les chaînes de caractères

- - - > Écrire un programme qui permet de transformer tout caractère alphabétique en son équivalent minuscule.

- - - > Solution : exploiter la méthode `toLowerCase()`

Quelques méthodes utiles (2/12)

toLowerCase()

Solution en utilisant la méthode toLowerCase() :

- elle permet de transformer tout caractère alphabétique en son équivalent minuscule.
- Elle n'a aucun effet sur les chiffres : ce ne sont pas des caractères alphabétiques. On peut donc l'utiliser sans problème sur une chaîne de caractères comportant des nombres.

Quelques méthodes utiles (3/12)

toLowerCase()

Exemple de solution

```
public static void main(String[] args) {  
    String chaine = new String("BONJOUR TOUT LE MONDE !"),  
    chaine2 = new String();  
    chaine2 = chaine.toLowerCase();  
    System.out.println(chaine2);  
}
```

Quelques méthodes utiles (4/12)

toUpperCase()

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

Écrire un programme qui permet de transformer tout caractère alphabétique en son équivalent majuscule.

Quelques méthodes utiles (5/12)

toUpperCase()

Exemple de solution

```
public static void main(String[] args) {  
    String chaine = new String("Bonjour le monde"), chaine2 =  
new String();  
    chaine2 = chaine.toUpperCase();  
    System.out.println(chaine2);  
}
```


Quelques méthodes utiles (6/12)

charAt ()

- Le résultat de la méthode charAt () sera un caractère : il s'agit d'une méthode d'extraction de caractère. Elle ne peut s'opérer que sur des String
 - elle présente la même particularité que les tableaux, c'est-à-dire que, pour cette méthode, le premier caractère sera le numéro 0.
 - Cette méthode prend un entier comme argument.
- > Écrire un programme qui permet d'extraire le caractère 0.

Quelques méthodes utiles (7/12)

charAt()

Exemple de solution

```
public static void main(String[] args) {  
    String nbre = new String("bonjour");  
    char caract = nbre.charAt(4);  
    System.out.println(caract);  
}
```

Quelques méthodes utiles (8/12)

substring()

- 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.

Là encore, le premier caractère porte le numéro 0.

---> Écrire un programme qui permet d'extraire La chaîne de caractère « il fait beau » de la phrase « aujourd'hui, il fait beau » .

Quelques méthodes utiles (9/12)

substring()

Exemple de solution

```
public static void main(String[] args) {  
    String chaine = new String("Aujourd'hui il  
fait beau"), chaine2 = new String();  
    chaine2 = chaine.substring(12,24);  
    System.out.println(chaine2);  
}
```

Quelques méthodes utiles (10/12)

- Les méthodes que nous allons voir nécessitent la classe `Math`, présente dans `java.lang`: elle fait donc partie des fondements du langage.

Aucun import particulier n'est nécessaire pour utiliser la classe `Math` qui regorge de méthodes utiles :

Quelques méthodes utiles (11/12)

Écrire un programme qui permet de calculer:

- la fonction pour des valeurs aléatoire: 0.11623710711306934
- le sinus de 120 grâce à la fonction sin
- le cosinus de 120 grâce à la fonction cos
- la tangente de 120 grâce à la fonction tan
- la valeur absolue de -120.25 grâce à la fonction abs
- le exposant de 2 grâce à la fonction pow

Quelques méthodes utiles (12/12)

```
public static void main(String[] args) {  
    double X = 0.0;  
    X = Math.random();  
    //Retourne un nombre aléatoire  
    //compris entre 0 et 1, comme 0.0001385746329371058  
  
    double sin = Math.sin(120);    //La fonction sinus  
    double cos = Math.cos(120);    //La fonction cosinus  
    double tan = Math.tan(120);    //La fonction tangente  
    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  
    System.out.println("la fonction pour des valeurs aléatoire: " + X);  
    System.out.println("le sinus de 120 est : " + sin);  
    System.out.println("le cosinus de 120 est: " + cos);  
    System.out.println("la tagente de 120 est : " + tan);  
    System.out.println("la valeur absolue de -120.25 est : " + abs);  
    System.out.println("le exposant de 2 est : " + exp);} 
```

Créer sa propre méthode (1/9)

- Impossible de faire un récapitulatif de toutes les méthodes présentes dans Java, ...
- Toutes ces méthodes sont très utiles.
- Sont aussi utiles, celles que nous écrivons nous-mêmes !

Voici un exemple de méthode:

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


Créer sa propre méthode (2/9)

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

Décortiquons un peu cela :

- Tout d'abord, il y a le mot clé public. C'est ce qui définit la portée de la méthode, (programmerons des objets).
- Ensuite, il y a static (à voir plus loin).
- Juste après, nous voyons double. Il s'agit du type de retour de la méthode. Pour faire simple, ici, notre méthode va renvoyer un double !

Créer sa propre méthode (3/9)

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

- Vient ensuite le nom de la méthode. C'est avec ce nom que nous l'appellerons.
- Puis arrivent les arguments de la méthode. Ce sont en fait les paramètres dont la méthode a besoin pour travailler. Ici, nous demandons d'arrondir le doubleA avec B chiffres derrière la virgule.
- Finalement, vous pouvez voir une instruction return à l'intérieur de la méthode. C'est elle qui effectue le renvoi de la valeur, ici un double.

Créer sa propre méthode (4/9)

Deux choses concernant les méthodes :

1. elles ne sont pas limitées en nombre de paramètres;
2. 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 ;
 - les méthodes qui retournent des types primitifs (double, int etc.). Elles sont de type double, int, char etc. Celles-ci possèdent une instruction return ;
 - les méthodes qui retournent des objets. Par exemple, une méthode qui retourne un objet de type String. Celles-ci aussi comportent une instruction return.

Créer sa propre méthode (5/9)

- Nous avons besoin de déclarer et d'afficher le contenu de 20 tableaux. Comment il faut faire?
- Écrire vingt-deux boucles for !!! ou
- Écrire une méthode. Celle-ci va :
 - prendre un tableau en paramètre ;
 - parcourir le tableau à notre place ;
 - effectuer tous les `System.out.println()` nécessaires ;
 - ne rien renvoyer.

Créer sa propre méthode (6/9)

La méthode sera de type void et qu'elle prendra un tableau en paramètre.

Voici un exemple de code complet :

```
package natives;

public class Classe_principale {

    public static void main(String[] args)
    {
        String[] tab = {"toto", "tata", "titi", "tete"};
        parcourirTableau(tab);
    }
    static void parcourirTableau(String[] tabBis)
    {
        for(String str : tabBis)
            System.out.println(str);
    }
}
```

Créer sa propre méthode (7/9)

- Les méthodes ajoutées dans la classe `main` doivent être déclarées `static`.
- La méthode `parcourirTableau` passé en paramètre.
- Si vous créez plusieurs tableaux et appelez la méthode sur ces derniers, la méthode affiche le contenu de chaque tableau !

Créer sa propre méthode (8/9)

Un autre exemple:

Écrire un programme composé de deux méthodes:

- Une méthode permettant de parcourir un tableau « parcourirTableau » et écrit au fur et à mesure le contenu du tableau dans la console.
- Une deuxième méthode toString() qui retourne une chaîne de caractères, que nous devons afficher à l'aide de l'instruction System.out.println().

Créer sa propre méthode (9/9)

Exemple de solution

```
public class Classe_principale {  
    public static void main(String[] args)  
    {  
        String[] tab = {"toto", "tata", "titi", "tete"};  
        parcourirTableau(tab);  
        System.out.println(toString(tab));  
    }  
    static void parcourirTableau(String[] tab)  
    {  
        for(String str : tab)  
            System.out.println(str);  
    }  
    static String toString(String[] tab)  
    {  
        System.out.println("Méthode toString() !\n-----");  
        String retour = "";  
        for(String str : tab)  
            retour += str + "\n";  
        return retour;  
    }  
}
```