TUTORIAL JAVA

Yassine moumen

Ce langage fait partie des nombreux outils indispensables au développement informatique. Dans le web,
beaucoup d'évolutions reposent que l'utilisation de cette technologie.

EN INFORMATIQUE, QU'EST-CE QUE JAVA ?

La technologie Java définit à la fois un langage de programmation orienté objet et une plateforme informatique.
Créée par l'entreprise Sun Microsystems (souvent juste appelée "Sun") en 1995, et reprise depuis par la société Oracle en 2009, la technologie Java est indissociable du domaine de l'informatique et du Web. On la retrouve
donc sur les ordinateurs, mais aussi sur les téléphones mobiles, les consoles de jeux, etc. L'avènement du smartphone et la puissance croissante des ordinateurs, ont entraîné un regain d'intérêt pour ce langage de programmation.

QUELLE EST LA DERNIÈRE VERSION DE JAVA ?

Depuis son lancement en 1995, Java a connu de multiples versions et améliorations. Ces dernières années, les
versions Java 7 et Java 8, toutes deux existant en version Java 64 bits et 32 bits, font figure d'incontournables.
Java 9 est actuellement en préparation.

INSTALLATION DU JDK

Voilà nous avons un peut appris sur le monde de la programmation. Il est temps d'installer nos outils. Alors le
premier outil que nous allons installer est le JDK qui veut dire Java Développer Kit qui nous permettra de
travailler avec java car le JDK nous permettra de compiler ou lancer nos lignes de code.
Pour l'installer nous allons devoir aller sur le site oracle et sélectionner JDK et prendre la dernière version.
https://www.oracle.com/index.html

INSTALLATION DU JRE

L'environnement d'exécution Java (abr. JRE pour Java Runtime Environment), parfois nommé simplement « Java
», est une famille de logiciels qui permet l'exécution des programmes écrits en langage de programmation Java, sur différentes plateformes informatiques

1: Les Variable

Les variables sont des conteneurs pour stocker les valeurs de données. En Java, il existe différents types de variables, par exemple:

- ✓ `Chaîne` - stocke du texte, tel que "Bonjour". Les valeurs de chaîne sont entourées de guillemets doubles
- ✓ `int` - stocke les entiers (nombres entiers), sans décimales, comme 123 ou -123
- ✓ `float` - stocke les nombres à virgule flottante, avec des décimales, telles que 19,99 ou -19,99
- ✓ `char` - stocke des caractères uniques, tels que «a» ou «B». Les valeurs Char sont entourées de guillemets simples
- ✓ `booléen` - stocke les valeurs avec deux états: vrai ou faux

✓ Déclaration (Création) de Variables

✓ Pour créer une variable, vous devez spécifier le type et lui attribuer une valeur:

✓ Syntaxe

```
type variable = value;
```

✓ Une démonstration de la façon de déclarer des variables d'autres types:

- **Example**

```
int myNum = 5;
float myFloatNum = 5.99f;
char myLetter = 'D';
boolean myBool = true;
String myText = "Hello";
```

2 : BOUCLES

- ✓ Les boucles peuvent exécuter un bloc de code tant qu'une condition spécifiée est atteinte.
- ✓ Les boucles sont pratiques car elles permettent de gagner du temps, de réduire les erreurs et de rendre le code plus lisible.

✓ WHILE

- ✓ La `while` boucle parcourt un bloc de code tant qu'une condition spécifiée est `true`.

```
✓ while (condition) {
✓   // code block to be executed
✓ }
```

✓ LA BOUCLE DO / WHILE

La `do/while` boucle est une variante de la `while` boucle. Cette boucle exécutera le bloc de code une fois, avant de vérifier si la condition est vraie, puis elle répétera la boucle tant que la condition est vraie.

```
do {
  // code block to be executed
}
while (condition);
```

✓ POUR LA BOUCLE

Lorsque vous savez exactement combien de fois vous souhaitez parcourir un bloc de code, utilisez la `for`boucle au lieu d'une `while`boucle:

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

3 : JAVA METHODS

Une méthode est un bloc de code qui ne s'exécute que lorsqu'elle est appelée.

Vous pouvez transmettre des données, appelées paramètres, à une méthode.

Les méthodes sont utilisées pour effectuer certaines actions et sont également appelées fonctions.

Pourquoi utiliser des méthodes? Pour réutiliser le code: définissez le code une fois et utilisez-le plusieurs fois.

✓ CRÉER UNE MÉTHODE

Une méthode doit être déclarée dans une classe. Il est défini avec le nom de la méthode, suivi de parenthèses (). Java fournit des méthodes prédéfinies, telles que System.out.println (), mais vous pouvez également créer vos propres méthodes pour effectuer certaines actions:

- **Example :**

Create a method inside Main:

```
public class Main {

  static void myMethod() {

    // code to be executed

  }

}
```

EXEMPLE EXPLIQUÉ :

- ✓ `myMethod ()` est le nom de la méthode
- ✓ `static` signifie que la méthode appartient à la classe Main et non à un objet de la classe Main. Vous en apprendrez plus sur les objets et comment accéder aux méthodes via des objets plus loin dans ce didacticiel.
- ✓ `void` signifie que cette méthode n'a pas de valeur de retour. Vous en apprendrez plus sur les valeurs de retour plus loin dans ce chapitre

APPELER UNE MÉTHODE :

Pour appeler une méthode en Java, écrivez le nom de la méthode suivi de deux parenthèses **()** et d'un point-virgule;

Dans l'exemple suivant, `myMethod ()` est utilisé pour imprimer un texte (l'action), lorsqu'il est appelé:

- **Example :**

Inside `main`, call the `myMethod()` method:

```
public class Main {
  static void myMethod() {
    System.out.println("I just got executed!");
  }
  public static void main(String[] args) {
    myMethod();
  }
}
// Outputs "I just got executed!"
```

✓ JAVA METHOD PARAMETERS:

Paramètres et arguments

Les informations peuvent être transmises aux méthodes en tant que paramètre. Les paramètres agissent comme des variables à l'intérieur de la méthode.

Les paramètres sont spécifiés après le nom de la méthode, entre parenthèses. Vous pouvez ajouter autant de paramètres que vous le souhaitez, séparez-les simplement par une virgule.

L'exemple suivant a une méthode qui prend une chaîne appelée fname comme paramètre. Lorsque la méthode est appelée, nous transmettons un prénom, qui est utilisé à l'intérieur de la méthode pour afficher le nom complet:

- **Example**

```
public class Main {
  static void myMethod(String fname) {
```

```
    System.out.println(fname + " Refsnes");
  }
  public static void main(String[] args) {
    myMethod("Liam");
    myMethod("Jenny");
    myMethod("Anja");
  }
}
// Liam Refsnes
// Jenny Refsnes
// Anja Refsnes
```

4: ARRAY LIST JAVA

La `ArrayList` classe est un tableau redimensionnable , qui peut être trouvé dans le `java.util` package.

La différence entre un tableau intégré et un tableau `ArrayList` en Java, est que la taille d'un tableau ne peut pas être modifiée (si vous souhaitez ajouter ou supprimer des éléments dans / d'un tableau, vous devez en créer un nouveau). Alors que des éléments peuvent être ajoutés et supprimés d'un `ArrayList` fichier quand vous le souhaitez. La syntaxe est également légèrement différente:

```
import java.util.ArrayList; // import the ArrayList class


ArrayList<String> rooms = new ArrayList<String>(); // Create an
ArrayList object
```

✓ AJOUTER DES ARTICLES

La `ArrayList`classe a de nombreuses méthodes utiles. Par exemple, pour ajouter des éléments au `ArrayList`, utilisez la `add()`

✓ ACCÉDER À UN ÉLÉMENT

Pour accéder à un élément dans `ArrayList`, utilisez la `get()`

✓ CHANGER UN ÉLÉMENT

Pour modifier un élément, utilisez la `set()`

✓ SUPPRIMER UN ÉLÉMENT

Pour supprimer un élément, utilisez la `remove()`

Pour supprimer tous les éléments de `ArrayList`, utilisez la `clear()`

5:JAVA HASHMAP

- ✓ Dans ce ArrayList chapitre, vous avez appris que les tableaux stockent les éléments en tant que collection ordonnée et que vous devez y accéder avec un numéro d'index (Int type). A HashMap Toutefois, les éléments de magasin dans « **clé** / **valeur** paires », et vous pouvez y accéder par un index d'un autre type (par exemple String).

- ✓ Un objet est utilisé comme clé (index) vers un autre objet (valeur). Il peut stocker différents types: Tringlés et Integer valeurs, ou le même type, comme: string clés et String valeurs:

```
✓ import java.util.HashMap; // import the HashMap class
✓
✓ HashMap<String, String> objet = new HashMap<String, String>();
```

✓ AJOUTER DES ARTICLES

La `HashMap` classe a de nombreuses méthodes utiles. Par exemple, pour y ajouter des éléments, utilisez la `put()`

✓ ACCÉDER À UN ÉLÉMENT

Pour accéder à une valeur dans `HashMap`, utilisez la `get()`

✓ SUPPRIMER UN ÉLÉMENT

Pour supprimer un élément, utilisez la `remove()`

Pour supprimer tous les éléments, utilisez la `clear()`

✓ TAILLE DE HASHMAP

Pour savoir combien il y a d'éléments, utilisez la `size`

6:JAVA HASHSET

Un HashSet est une collection d'éléments où chaque élément est unique et se trouve dans le `java.util` .

```
import java.util.HashSet; // Import the HashSet class


HashSet<String> objet = new HashSet<String>();
```

✓ AJOUTER DES ARTICLES

La HashSet classe a de nombreuses méthodes utiles. Par exemple, pour y ajouter des éléments, utilisez la add()

✓ VÉRIFIER SI UN ÉLÉMENT EXISTE

Pour vérifier si un élément existe dans un HashSet, utilisez la contains()

✓ SUPPRIMER UN ÉLÉMENT

Pour supprimer un élément, utilisez la remove()

Pour supprimer tous les éléments, utilisez la clear()

✓ TAILLE DU HASHSET

Pour savoir combien il y a d'éléments, utilisez la size()