# Compte-Rendu de la séance n°1

| | |
|---|---|
| **Nom de la formatrice.** | MARZAK Bouchra |
| **Nom de l'apprenant.** | JAMAL EDDINE ELIDRISSI Yassir. |
| **Date.** | 17/11/2021 |

La première journée à Youcode en tant JAVA apprenant peut-être scinder en deux parties :

D'une part, la matinée, dans laquelle j'ai pu avoir une idée globale sur le programme des sept mois à venir, mais aussi de connaitre mes collègues de classe grâce à l'activité brise-glace, en l'occurrence la présentation croisée.

D'une autre part, l'après-midi, pendant laquelle j'ai pu explorer quelques articles, dont je peux en résumer ce qui suit :

- Après un benchmark des trois IDE, à savoir Eclipse, NetBeans et IntelliJ j'ai conclus qu'Eclipse est le plus optimale et ce car son évaluation par la communauté des développeurs est supérieure (4.8/5) quant à la performance, de même Eclipse est doté d'une bonne réputation (face à une performance moyenne pour les deux autres IDE).
- La version Java 11 malgré le fait qu'elle soit LTS (Long-term support) elle n'a pas apporté des mises à jour majeurs, cela n'empêche d'en citer quelque unes à savoir :
  A) Des améliorations pour les développeurs (Sealed classes, en général les classes et interfaces scellées limitent les autres classes ou interfaces qui peuvent les étendre ou les implémenter.).
  B) Des améliorations concernant Apple (nouvelle manière « RENDERING PIPELINE » qui a remplacé l'Apple OpenGL API).
  C) Des améliorations en ce qui concerne le nettoyage (ce sont des mesures de sécurité qui sont obsolète (Applet, RMI …) remplacer par des nouvelles plus robustes).
- D'abord qu'est ce que le langage JAVA ? C'est un langage de programmation de haut niveau, robuste et orienté objet, mais aussi sécurisé. Développé en 1995 par Sun Microsystems une filiale d'Oracle.
- JAVA est une plateforme ? Oui. Étant donné que Java possède un environnement d'exécution (JRE) et une API, on l'appelle donc une plateforme (Une plateforme c'est l'environnement dans lequel un programme peut être exécuter).
- Comment écrire un Hello world en JAVA ?
  A) On crée une classe.

B) On déclare notre méthode main qu'on la trouve dans les langages orienté objet, cette dernière se situe dans la classe principale dans la racine, sachant que dans JAVA tout les programmes doivent contenir la méthode main. Cette méthode doit contenir ce qui suit :
    - L'Access modifier (public, private …)
    - Type de retour (Return type) VOID signifie qu'on n'a pas de type de retour.
    - Le mot clé STATIC, si on souhaite invoquer la méthode sans instancier l'objet de la classe.
    - Le nom de la méthode (main).
    - Les arguments, la méthode main accepte un seul argument sous forme d'un tableau de chaine de caractères String (String[] args) dans lequel le système passe les informations au programme, sachant que chaque string dans l'array est une ligne de commande ou un argument à passer.
    - Nb : Les mots clés public et static peuvent être écrit dans l'ordre qu'on souhaite + on peut donner n'importe quel nom à l'argument sauf que la plupart utilisent args.
C) On fait une assignation de la chaine de caractère (HELLO WORLD) à une instance au non-primitif data-type String* nommé bonjour (je vais en détailler plus tard la différence entre les types de données etc) String bonjour = ''HELLO WORLD''.
D) Enfin pour imprimer l'argument qu'on lui a passé, on se réfère à la méthode println(bonjour) qui permet d'afficher les résultats à l'écran, mais avant ça il faut passer par la classe System.out :
    - System est le nom de la classe qui réside dans java.lang package tandis que out est une instance de java.io.PrintStream donc println est une méthode de java.io.PrintStream.
    - Nb : La méthode println() est similaire à la méthode print(), sauf qu'elle déplace le curseur à la ligne suivante après avoir imprimé le résultat.
    - Au final on peut écrire notre code :
    - A : class Helloworld
    - B: public static void main(String args[])
    - C: String bonjour = ''HELLO WORLD''
    - D: System.out.print(bonjour);

Le langage JAVA est interprété mais aussi compilé, comment le processus est assuré ? En effet, JAVA possède le principe de JDK en l'occurrence Java Développement Kit ce dernier englobe d'une part les outils pour pouvoir développer les programmes Java et d'une autre l'environnement d'exécution de Java dit JRE c'est globalement ce qui nous permet d'écrire compiler et exécuter un programme java. Cependant, un JRE est capable de faire tout ça car il contient des éléments primordiaux à savoir les fichiers de bibliothèque et les classes du paquet java (Math …). Mais aussi, il contient un élément indispensable, en l'occurrence le JVM (Java Virtual Machine) les codes d'octets (Byte codes) sont le langage machine de la JVM ce qui lui permet d'être chargée sur diverses plates-formes, c'est donc la JVM qui exécute le Java Byte Code qui s'est convertis d'un fichier Java à une classe fichier (Byte Code) à l'aide du JRE. Pour résumer :

Le JDK contient le JRE qui englobe le JVM + d'autres ressources (un interpréteur {java}, un compilateur {javac}, un archiveur {jar}, un générateur de documentation {Javadoc}.

Types de données primitives et non-primitives :

- On a 8 types de données primitives en Java : byte, short, int, long, float, double, boolean, char. Un type de donnée primitif spécifie la taille (en bytes) et le type des valeurs de la variable (précédemment cités), et il n'a pas de méthodes supplémentaires (contrairement aux non-primitives).
- Nb : le type de donnée String est appelé "le neuvième type spécial". Cependant, en java le type de donnée String est en fait non primitif et ce principalement car il fait référence à un objet, c'est pour ça qu'on les appels les types de référence.
- Pour résumer, les principales différences entre les types de données primitifs et non primitifs sont les suivantes :
    - A) Les types primitifs sont prédéfinis par contre les types non primitifs ne sont pas définis par Java (Sauf pour le string)
    - B) Les types non primitifs peuvent être utilisés pour appeler des méthodes.
    - C) Un type primitif a toujours une valeur, tandis que les types non primitifs peuvent être nuls.
    - D) Un type primitif commence par une lettre minuscule, tandis que les types non primitifs commencent par une lettre majuscule.
    - NB : Les Strings, Arrays, Classes, Interface... sont des exemples de types non primitifs.