**Introduction au java**

**\*Java** est un langage de programmation populaire, créé en 1995. Il appartient à Oracle.

\***utilisation de java**:

* Applications mobiles (en particulier les applications Android)
* Applications de bureau
* des applications Web
* Serveurs Web et serveurs d'applications
* Jeux
* Connexion à la base de données
* …

\***importance de Java**?

* Java fonctionne sur différentes plates-formes (Windows, Mac, Linux, Raspberry Pi, etc.)
* C'est l'un des langages de programmation les plus populaires au monde
* Il est facile à apprendre et simple à utiliser
* Il est open-source et gratuit
* Il est sécurisé, rapide et puissant
* Il a un énorme soutien de la communauté (des dizaines de millions de développeurs)
* Java est un langage orienté objet qui donne une structure claire aux programmes et permet de réutiliser le code, réduisant ainsi les coûts de développement

\* JRE = Java Runtime Environment - ce dont vous avez besoin pour exécuter des programmes / logiciels nécessitant Java ou utiliser des bibliothèques écrites en Java. Par exemple, OpenOffice nécessite l'environnement d'exécution Java

\* JDK / Java SDK = Kit de développement Java / Kit de développement logiciel Java - ce dont vous avez besoin pour écrire des programmes nécessitant Java ou utiliser des bibliothèques écrites en Java. Par exemple, si vous deviez écrire votre propre outil de traitement de texte en Java.

\* java est livré avec le JRE car il lance la VM (machine virtuelle). Il peut prendre des fichiers de classe qui sont des fichiers qui ont été compilés à l'aide du JDK.

\* Le JDK est livré avec javac car c'est ce dont vous avez besoin pour compiler vos fichiers .java dans des fichiers .class qui peuvent ensuite s'exécuter sur le JRE.

\* Chaque ligne de code exécutée en Java doit se trouver dans un fichier **class**.

\* Une classe doit toujours commencer par une première lettre majuscule parce que java est « case sensitive ».

# \***main() méthode**, Tout code à l'intérieur de la main()méthode sera exécuté.

# \* **System.out.println ()** : méthode pour imprimer une ligne de texte à l'écran

# \* **variables** sont des élément utiliser pour stocker les donnes et pour créer une variable, vous devez spécifier le type et lui attribuer une valeur : type variable = value;

\* **nextLine()** method of **[java.util.Scanner](https://www.geeksforgeeks.org/scanner-class-in-java/)** class advances this scanner past the current line and returns the input that was skipped

\***Fonction** or bien Une **méthode :** est un bloc de code qui ne s'exécute que lorsqu'elle est appelée.

\* **collection** : est un terme général qui fait référence à plusieurs objets maintenus ensemble en mémoire de manière structurée

**\*Différence entre set, list, map**



|  |  |  |  |
| --- | --- | --- | --- |
|  | set | List | map |
| Duplicates | doesn’t allow duplicates. | List allows duplicate elements. | stored the elements as key & value pair. Map doesn’t allow duplicate keys while it allows duplicate values |
| Ordering | Unordered | Maintains order | Unordered |
| Index based | No | yes | No |

**\*When to use List, Set and Map in Java?**

# 1) Si vous ne souhaitez pas avoir de valeurs en double dans la base de données, Set devrait être votre premier choix car toutes ses classes n'autorisent pas les doublons.

# 2) S'il y a un besoin d'opérations de recherche fréquentes basées sur les valeurs d'index, alors List (ArrayList) est un meilleur choix.

# 3) S'il est nécessaire de maintenir l'ordre d'insertion, la liste est également une interface de collecte préférée.

4) Si l'exigence est d'avoir les mappages clé et valeur dans la base de données, alors Map est votre meilleur pari.