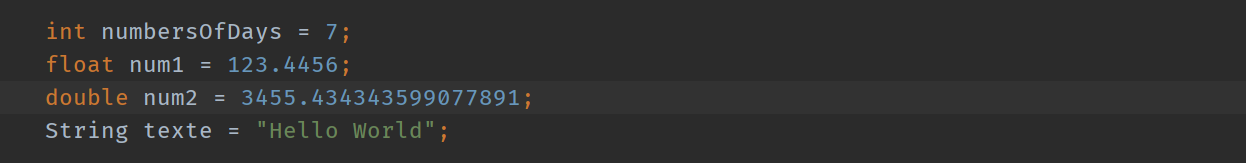
**Découverte de language Java**

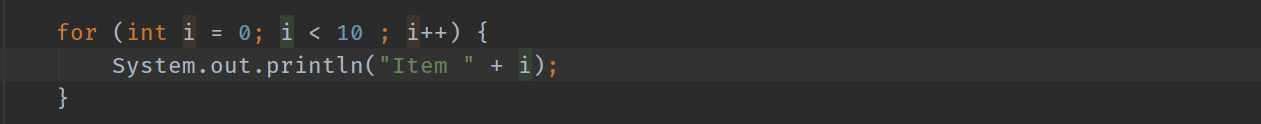
1. **Les variables :**

* Une variable peut être vue comme une boîte qui contient une valeur. Cette boîte est stockée sur une étagère dans un entrepôt gigantesque. L'emplacement de chaque boîte de cet entrepôt est soigneusement enregistré, tout comme votre ordinateur enregistre l'emplacement de votre variable en mémoire.
* Pour utiliser une variable dans le code, vous devez la créer, ou, en langage de développeur, la **déclarer**. Il existe plusieurs **types** de variables en Java, qui seront utilisées en fonction du type de valeur qu'elles contiennent :
  + **String** = texte
  + **int** = nombre entier
  + **double** = nombre en virgule flottante plus précis
  + **float** = nombre en virgule flottante moins précis
  + …

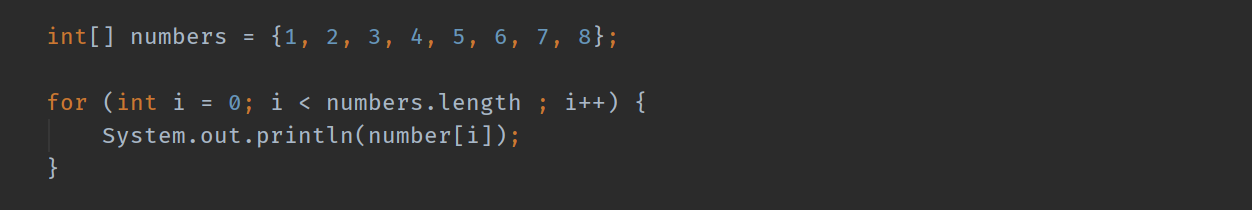


1. **Les boucles :**

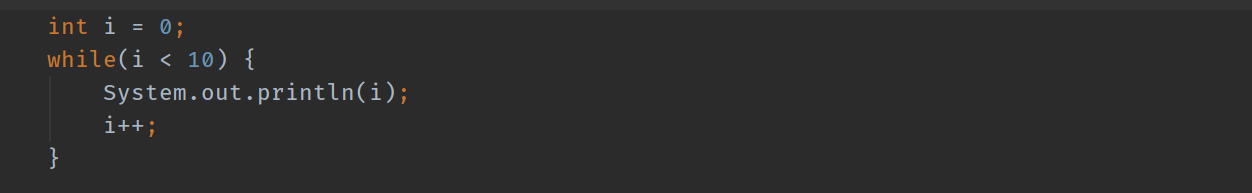
* une **boucle** est une technique qui permet de répéter une ou plusieurs instructions sans avoir à retaper le même ensemble d'instructions plusieurs fois.
* Les boucles énumérées sont des boucles qui sont utilisées si vous savez à l'avance combien de fois vous voulez faire une boucle. En Java, cela s'appelle des*boucles for*.



* Avec le **"for" amélioré**, il vous suffit de définir une variable du type du **tableau** ou de la **collection** que vous souhaitez mettre en boucle. Cette variable se verra attribuer la valeur de chaque élément du tableau ou de la collection, jusqu'à ce que vous ayez atteint la fin.

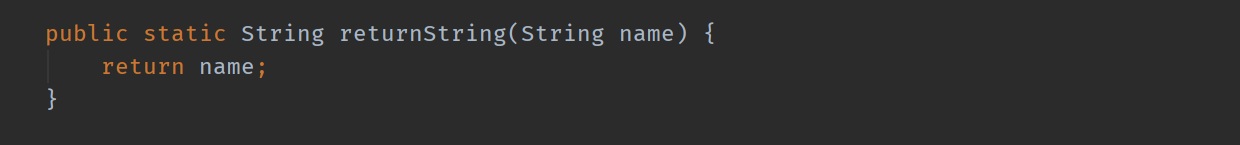


* les boucles conditionnelles exécutent un ensemble d'instructions jusqu'à ce qu'une condition définie soit remplie. En Java, il existe 2 type : *while* et *do..while*.

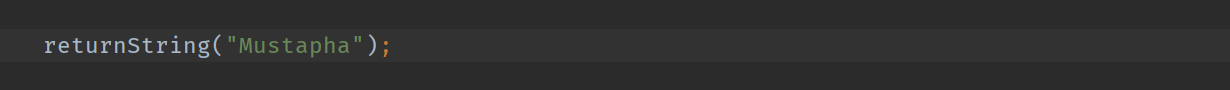


1. **Les fonctions :**

* Une fonction un sous-programme qui permet d'effectuer un ensemble d'instructions par simple appel de la fonction dans le corps du programme principal. Les fonctions permettent d'exécuter dans plusieurs parties du programme une série d'instructions, cela permet une simplicité du code et donc une taille de programme minimale.
* Avant d'être utilisée, une fonction doit être définie car pour l'appeler dans une classe il faut que le compilateur la connaisse, c'est-à-dire qu'il connaisse son nom, ses arguments et les instructions qu'elle contient.

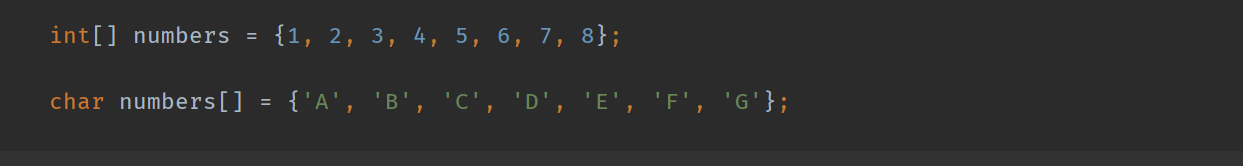


* Pour exécuter une méthode, il suffit de faire appel à le nom de la méthode, suivie de ses arguments entre parenthèse :



1. **Les collections :**
   1. **Les tableaux :**

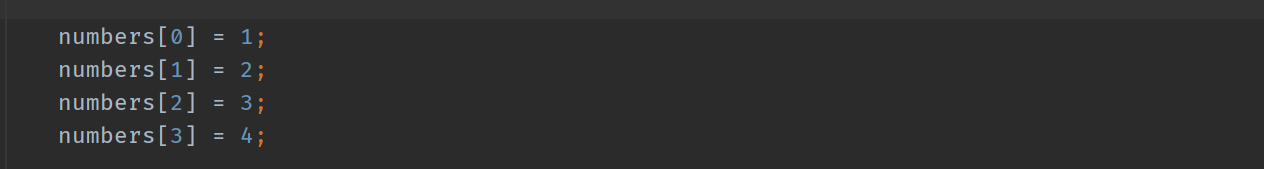
* Un tableau est une structure de données contenant un groupe d'éléments tous du même type. Le type des éléments peut être un type primitif ou une classe. Lors de la définition d’un tableau les [] spécifiant qu’il s’agit d’un tableau peuvent être placés avant ou après le nom du tableau.
* Un tableau peut être initialisé :



* Pour allouer l’espace nécessaire au tableau il faut utiliser new :



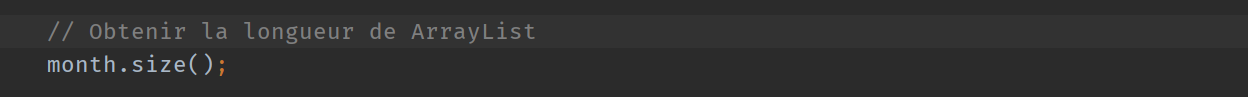
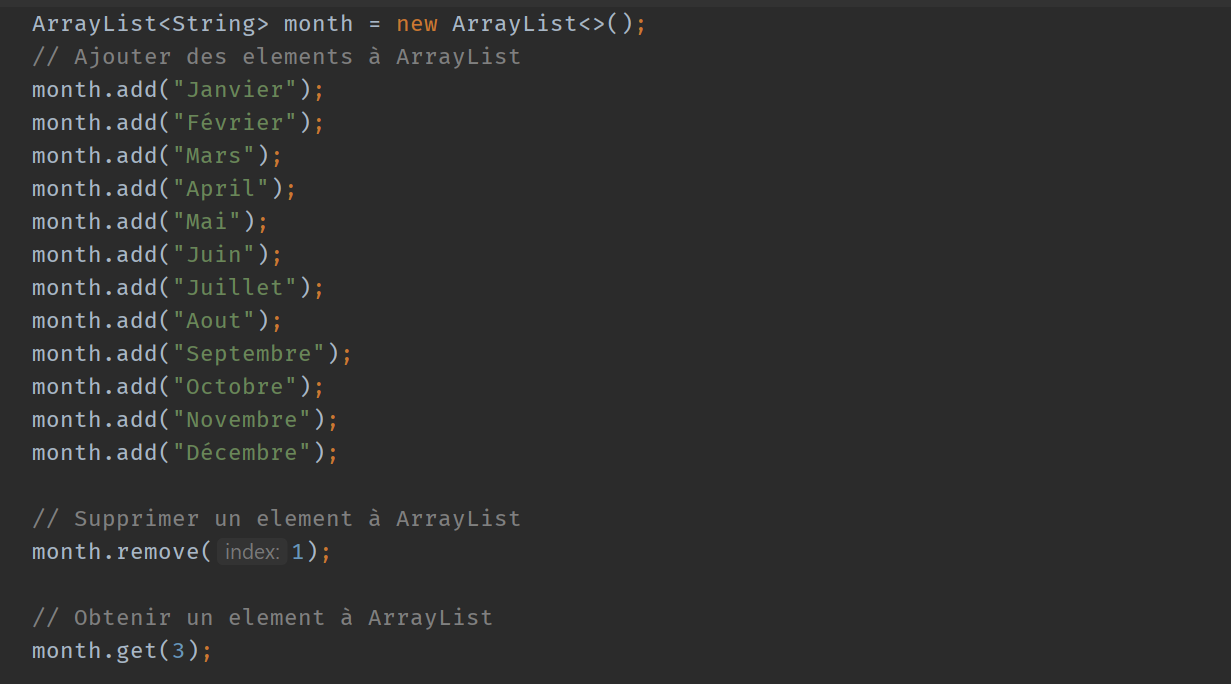
* Les éléments d’un tableau sont indicés à partir de 0. Chaque élément peut être accédé individuellement en donnant le nom du tableau suivi de l’indice entre [] :



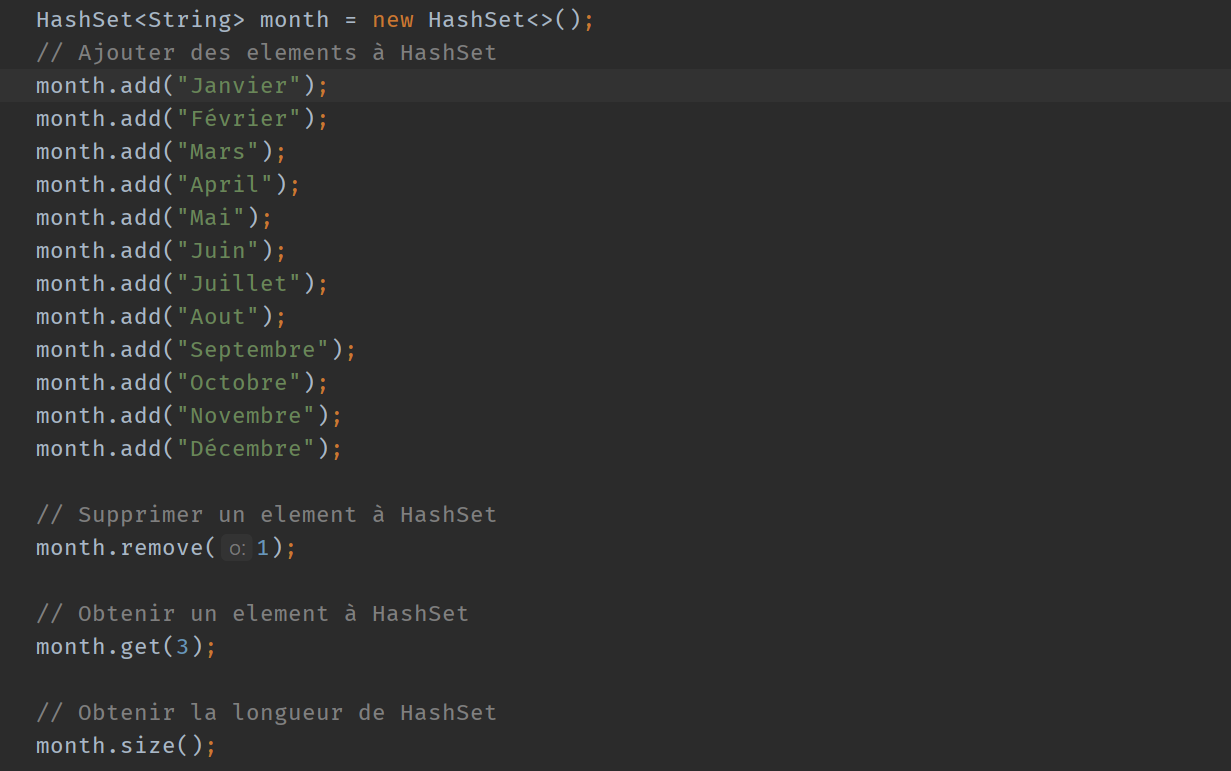
* Un tableau possède un attribut **length** qui permet de connaître le nombre d’éléments d’un tableau :



* 1. **ArrayList :**
* ArrayList est une classe qui peut être utilisée pour créer des conteneurs qui stockent des listes d'objets et aussi est redimensionnable dynamiquement, ce qui signifie que sa taille peut changer pendant l'exécution du programme.



* 1. **HashMap :**
* La classe HashSet présente plusieurs caractéristiques :
* elle ne propose aucune garantie sur l'ordre de parcours lors de l'itération sur les éléments qu'elle contient.
* elle ne permet pas d'ajouter des doublons mais elle permet l'ajout d'un élément **null**.



* 1. **HashMap :**
* La classe HashMap présente plusieurs caractéristiques :
  + Elle permet l'utilisation de la valeur null comme clé et comme valeur.
  + Elle n'est pas thread-safe.
  + Elle ne garantit aucun ordre lors du parcours des éléments de la collection.

