

# Découverte du Java

Michael BOUTBOUL

# Sommaire

<b>Avant Propos.....</b>	<b>3</b>
<b>1) Le langage Java.....</b>	<b>4</b>
<b>2) Les bases de développement.....</b>	<b>5</b>
a) Les types primitifs.....	5
b) Les commentaires.....	6
c) Afficher du texte dans la console.....	7
d) Récupérer la saisie de l'utilisateur.....	8
e) Structures conditionnelles.....	9
f) Boucles.....	11
g) Fonctions.....	12
h) Tableaux.....	13
i) ArrayList.....	15
j) Informations complémentaires.....	16

# Avant Propos

Le cours de Programmation Orientée Objet sera composée en deux parties :

- Initiation au Java (1 jour et demi)
- Concept de la Programmation Orientée Objet (2 jour et demi)

Les explications fournies lors de l'initiation au java part du principe que le cours d'algorithmique a été suivi. Les concepts de structures conditionnelles, boucles et autres outils de base nécessaires à la construction d'un algorithme ne seront expliqués que dans l'optique de comprendre leurs usages dans le cadre du langage Java.

La prise de note étant importante, le cours sera fourni en format modifiable afin de vous permettre de rajouter les éléments de cours que vous jugez opportun.

Conseil important : comme le cours de Programmation Orientée Objet mis en pratique par le langage Java ne dure que 4 jours, l'exhaustivité des concepts liés à l'IDE, au Java ou encore à la Programmation Orientée Objet ne pourra être abordée. Ainsi, la curiosité sera votre plus grand allié !

# 1)Le langage Java

Java est un langage orienté objet. Cela signifie qu'il est particulièrement bien indiqué pour les développements en POO (Programmation Orientée Objet). Toutefois, il est possible dans une certaine mesure, de développer en procédurale (les instructions sont réalisées dans l'ordre où elles se présentent).

Le Java est un langage stable, largement utilisé dans le monde entier. Il fait partie du top 5 des langages les plus utilisés depuis une vingtaine d'années.

Dans ce premier cours, afin d'assurer une prise en main du langage Java réussie, nous utiliserons uniquement les fonctions basiques que Java propose. La Programmation Orientée Objet fera l'objet d'une seconde partie.

Quelques généralités à savoir sur le Java :

- Les variables en Java sont typées ce qui signifie qu'il faut préciser le type des variables à chaque déclaration ;
- Nous n'avons pas besoin de gérer la mémoire en Java, la JVM le fait pour nous grâce à son système de ramasse miettes (autrement appelé **Garbage Collector**) ;
- En java, en dehors des types primitifs, tout est objet !

## 2) Les bases de développement

### a) Les types primitifs

Les types primitifs correspondent aux types de variables “basiques”. Contrairement à tous les autres objets, on les déclare toujours en minuscule. On peut les considérer comme des exceptions : ce ne sont pas des objets contrairement à tous les autres types que l’on peut voir en Java. On retrouve leurs caractéristiques dans le tableau ci-dessous :

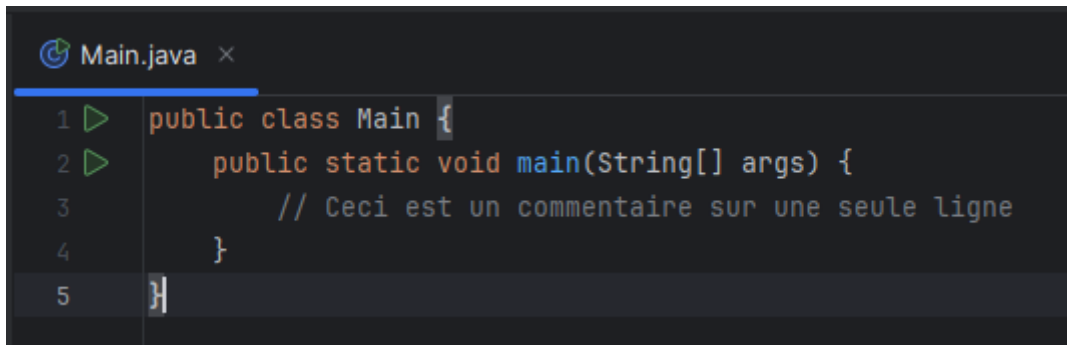
Type	Taille en octets	Valeur minimale	valeur maximale
byte	1	-128	127
short	2	-32 768	32 767
int	4	-2 147 483 648	2 147 483 647
long	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
float	4	-1.40239846E-45	3.40282347E38
double	8	-4.9406564584124654E-324	1.797693134862316E308
char	2		
boolean	true ou false		

Le type **String** (chaînes de caractères) n’est pas un type primitif. Toutefois, il bénéficie d’un traitement à part au niveau de la déclaration propre aux types primitifs. En effet, il est possible d’instancier un String sans utiliser le mot clé **new** (voir cours sur la Programmation Orientée Objet).

## b) Les commentaires

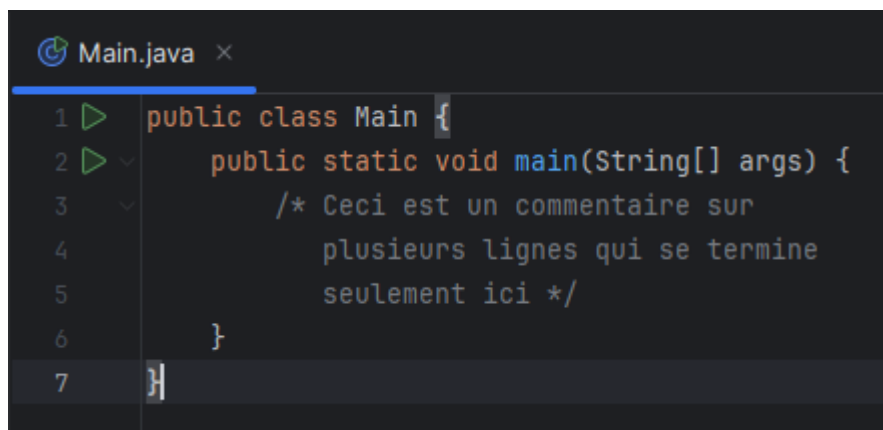
Il existe 2 syntaxes pour déclarer des commentaires :

- `//` : permet de déclarer une ligne de commentaire. La prochaine ligne ne sera pas considérée comme commentaire ;



```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         // Ceci est un commentaire sur une seule ligne
4     }
5 }
```

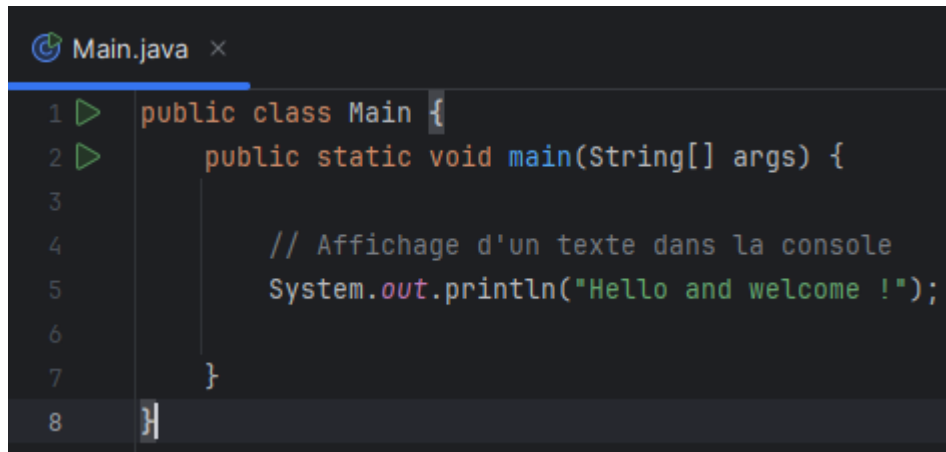
- `/*` : permet de déclarer un commentaire sur plusieurs lignes. La fin du commentaire est signalé par `*/` : tout ce qui se trouve entre `/*` et `*/` sera considéré comme du commentaire.



```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         /* Ceci est un commentaire sur
4            plusieurs lignes qui se termine
5            seulement ici */
6     }
7 }
```

## c) Afficher du texte dans la console

Pour afficher du texte dans la console on utilise le flux de sortie grâce à la commande **System.out.println(*mon texte*);**



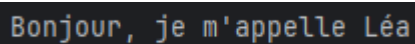
```
1 public class Main {  
2     public static void main(String[] args) {  
3  
4         // Affichage d'un texte dans la console  
5         System.out.println("Hello and welcome !");  
6  
7     }  
8 }
```

Pour afficher du texte dans la console en utilisant des variables, on va tout simplement concaténer les variables avec le texte que l'on veut voir apparaître :



```
1 public class Main {  
2     public static void main(String[] args) {  
3  
4         // Affichage d'un texte dans la console : concaténation avec une variable  
5         String prenom = "Léa";  
6         System.out.println("Bonjour, je m'appelle " + prenom);  
7  
8     }  
9 }
```

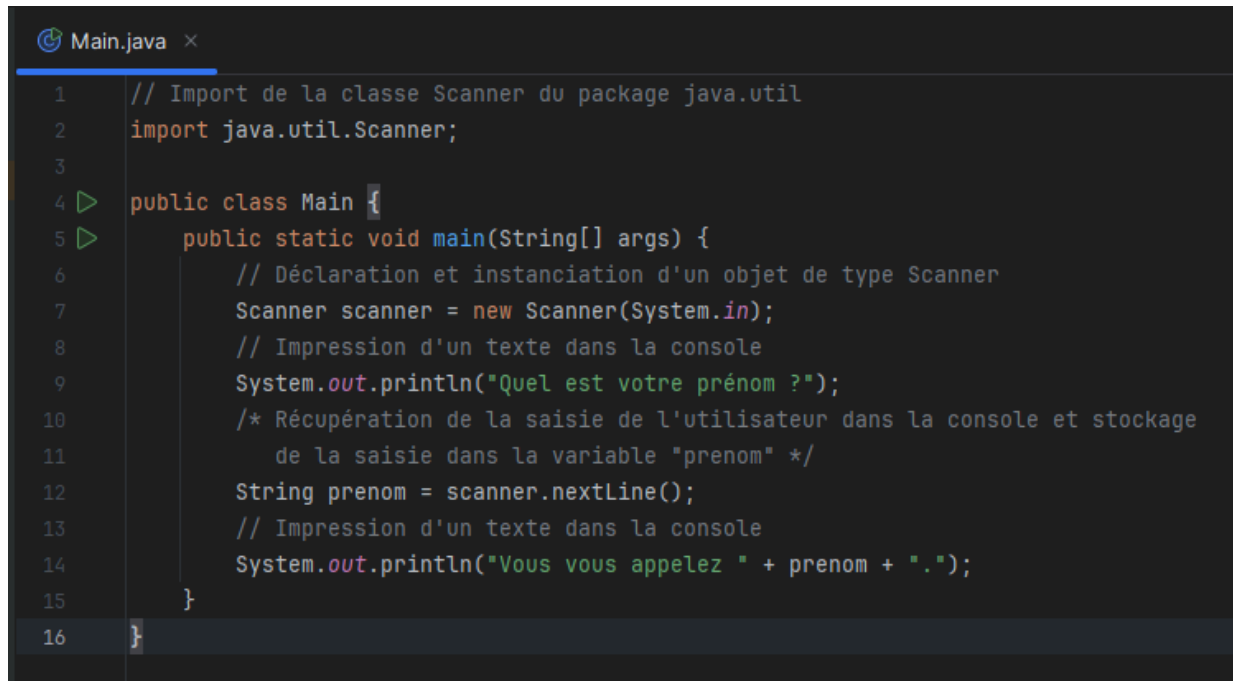
Résultat dans la console :



```
Bonjour, je m'appelle Léa
```

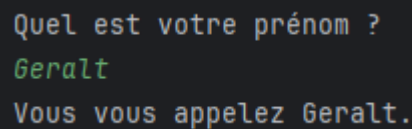
## d) Récupérer la saisie de l'utilisateur

La récupération des données saisies au clavier par l'utilisateur se fait grâce à l'objet **Scanner**. La classe permettant de créer cet objet se trouve dans le package **java.util**, il est nécessaire de l'importer dans le programme pour pouvoir l'utiliser.



```
1 // Import de la classe Scanner du package java.util
2 import java.util.Scanner;
3
4 public class Main {
5     public static void main(String[] args) {
6         // Déclaration et instanciation d'un objet de type Scanner
7         Scanner scanner = new Scanner(System.in);
8         // Impression d'un texte dans la console
9         System.out.println("Quel est votre prénom ?");
10        /* Récupération de la saisie de l'utilisateur dans la console et stockage
11           de la saisie dans la variable "prenom" */
12        String prenom = scanner.nextLine();
13        // Impression d'un texte dans la console
14        System.out.println("Vous vous appelez " + prenom + ".");
15    }
16 }
```

Résultat dans la console :



```
Quel est votre prénom ?
Geralt
Vous vous appelez Geralt.
```

D'autres méthodes de récupération sont disponibles:

<https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html>



## e) Structures conditionnelles

Les structures conditionnelles permettent de tester si une condition est vraie ou non. Cela permet d'exécuter des instructions seulement si certaines conditions sont réunies.

On a d'une part les **if...else if... else** :

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         int age = 12;
4         if(age<18){
5             System.out.println("Vous êtes mineur.");
6         } else {
7             System.out.println("Vous êtes majeur.");
8         }
9     }
10 }
```

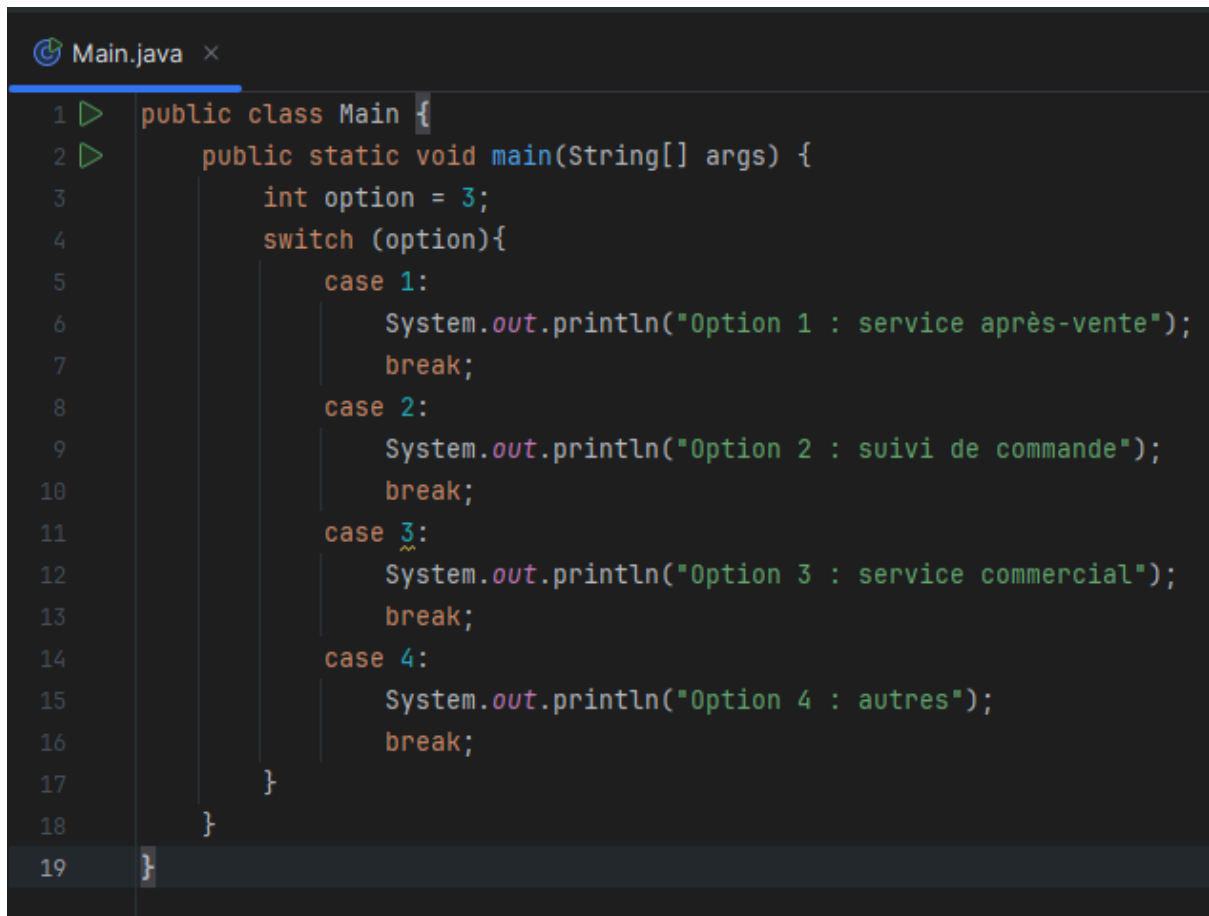
Syntaxe alternative :

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         int age = 12;
4         String etat;
5         etat = (age<18)?"mineur":"majeur";
6         System.out.println("Vous êtes " + etat + ".");
7     }
8 }
```

Pour utiliser plusieurs conditions, on utilise les caractères **&&** ('et' logique) et **||** ('ou' inclusif) :

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         int age1 = 10;
4         int age2 = 15;
5         if(age1 == age2 && age1<18){
6             System.out.println("Vous avez le même age et vous êtes mineurs.");
7         }
8         if(age1<18 || age2<18){
9             System.out.println("Au moins un de vous 2 est mineur.");
10        }
11    }
12 }
```

D'autre part on a les **switch ... case** :



```
1  public class Main {  
2      public static void main(String[] args) {  
3          int option = 3;  
4          switch (option){  
5              case 1:  
6                  System.out.println("Option 1 : service après-vente");  
7                  break;  
8              case 2:  
9                  System.out.println("Option 2 : suivi de commande");  
10                 break;  
11                 case 3:  
12                     System.out.println("Option 3 : service commercial");  
13                     break;  
14                 case 4:  
15                     System.out.println("Option 4 : autres");  
16                     break;  
17             }  
18         }  
19     }
```

## f) Boucles

Les boucles permettent de répéter plusieurs fois certaines lignes d'instructions. Les boucles **for** permettent de répéter les instructions selon un compteur :

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         for (int i = 0; i < 5; i++) {
4             System.out.println("Compteur i de la boucle = " + i + ". Cette instruction a été exécuté " + (i+1) + " fois.");
5         }
6     }
7 }
```

Résultat dans la console :

```
Compteur i de la boucle = 0. Cette instruction a été exécuté 1 fois.
Compteur i de la boucle = 1. Cette instruction a été exécuté 2 fois.
Compteur i de la boucle = 2. Cette instruction a été exécuté 3 fois.
Compteur i de la boucle = 3. Cette instruction a été exécuté 4 fois.
Compteur i de la boucle = 4. Cette instruction a été exécuté 5 fois.
```

Les boucles **while** permettent d'exécuter les instructions jusqu'à ce que la condition donnée soit vérifiée :

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         int compteur = 0;
4         while (compteur < 5) {
5             System.out.println("Compteur de la boucle = " + compteur +
6                 ". Cette instruction a été exécuté " + (compteur+1) + " fois.");
7             compteur++;
8         }
9     }
10 }
```

Résultat dans la console :

```
Compteur de la boucle = 0. Cette instruction a été exécuté 1 fois.
Compteur de la boucle = 1. Cette instruction a été exécuté 2 fois.
Compteur de la boucle = 2. Cette instruction a été exécuté 3 fois.
Compteur de la boucle = 3. Cette instruction a été exécuté 4 fois.
Compteur de la boucle = 4. Cette instruction a été exécuté 5 fois.
```

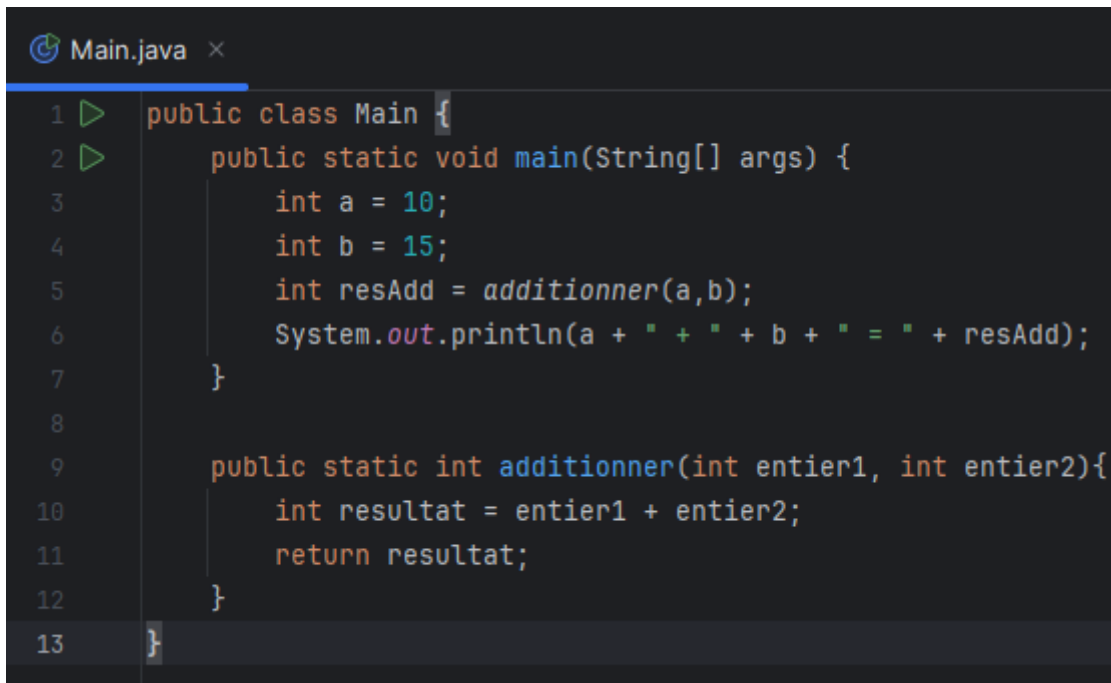
Les boucles **do...while** permettent d'exécuter au moins une fois les instructions avant de tester si la condition de sortie est vérifiée :

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3         int compteur = 0;
4         do {
5             System.out.println("Compteur de la boucle = " + compteur +
6                 ". Cette instruction a été exécuté " + (compteur+1) + " fois.");
7             compteur++;
8         } while (compteur < 5);
9     }
10 }
```

## g) Fonctions

Une fonction permet de créer un bloc d'instruction auquel on attribue un nom. On va pouvoir exécuter ces instructions en appelant la fonction grâce à sa signature composée de :

- **le type de la variable de retour** (dans l'exemple ci-dessous **int**). Si la fonction ne retourne rien, le mot clé **void** remplace le type. Dans notre exemple, la variable **resAdd** à gauche du égal, récupère ce que nous retourne la fonction **additionner**.
- **le nom de la fonction** (dans l'exemple ci-dessous **additionner**). Le nom peut être choisi librement à l'exception des mots-clés référencés en Java.
- **le ou les paramètres** nécessaires au bon fonctionnement de la fonction (dans l'exemple ci-dessous **(int entier1, int entier2)**). En effet, pour avoir le résultat de l'addition de deux nombres, nous avons besoin... de deux nombres ! Les paramètres sont nécessaires car la fonction "additionner" ne peut pas accéder directement aux variables "a" et "b" de la fonction **main**.



```
1 public class Main {
2     public static void main(String[] args) {
3         int a = 10;
4         int b = 15;
5         int resAdd = additionner(a,b);
6         System.out.println(a + " + " + b + " = " + resAdd);
7     }
8
9     public static int additionner(int entier1, int entier2){
10         int resultat = entier1 + entier2;
11         return resultat;
12     }
13 }
```

Résultat dans la console :

```
10 + 15 = 25
```

On constate que la fonction **main** a un type de retour **void** : cela implique que la fonction ne retourne rien.

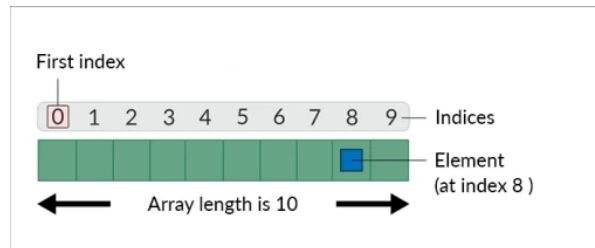
Attention :

- Les variables déclarées dans une fonction ne sont accessibles que dans la fonction elle-même et pas dans les autres fonctions.
- La fonction principale d'un programme s'appelle toujours **main**, c'est cette fonction qui sera automatiquement exécutée lors du lancement du programme. Cela implique que le code principal du programme se trouve dans cette fonction.
- Il est possible de passer des paramètres aux fonctions afin qu'elles aient toutes les informations nécessaires à leur traitement.

## h) Tableaux

Les tableaux permettent de stocker un nombre fini d'éléments (objets ou primitifs). Un tableau ne peut contenir que des éléments d'un même type : si le tableau est déclaré comme étant un tableau d'entier alors il ne pourra contenir que des entiers.

Chaque cellule d'un tableau est désignée par un indice. Les indices de tableaux débutent à 0. Si l'on veut récupérer la valeur stockée dans la première cellule d'un tableau, on ira chercher la valeur à l'indice 0. Les indices d'un tableau comportant 10 cellules seront compris entre 0 et 9 inclus.



Au niveau de la syntaxe :

- Pour déclarer un tableau, il suffit d'utiliser les caractères crochets **[ et ]** après le type du tableau
- Pour créer le tableau on utilise la syntaxe **new type[tailleTableau]**
- Pour accéder à une valeur, on utilise à nouveau les **[ ]** avec l'indice correspondant à la cellule dont on veut récupérer la valeur. Exemple: **notesMaths[i]** (ligne 11), récupère la variable du tableau **notesMaths** à l'indice **i**
- Pour récupérer la taille d'un tableau on utilise **nomTab.length**

```
Main.java x
1 public class Main {
2     public static void main(String[] args) {
3
4         // Déclaration et instanciation du tableau en renseignant sa taille
5         double[] notesMaths = new double[5];
6
7         // Remplissage du tableau indice par indice
8         notesMaths[0] = 14;
9         notesMaths[1] = 8;
10        notesMaths[2] = 18.5;
11        notesMaths[3] = 17.5;
12        notesMaths[4] = 11;
13
14        /* Parcours du tableau pour afficher les notes.
15         notesMaths.length permet de récupérer la taille du tableau : ici 5 */
16        for (int i = 0; i < notesMaths.length; i++) {
17            System.out.println("Note N°" + (i+1) + " : " + notesMaths[i]);
18        }
19    }
20 }
```

Résultat dans la console :

```
Note N°1 : 14.0  
Note N°2 : 8.0  
Note N°3 : 18.5  
Note N°4 : 17.5  
Note N°5 : 11.0
```

## i) ArrayList

### Exercice de lecture de la JAVADOC :

Grâce à la documentation officielle sur l'ArrayList, écrire un programme permettant de créer une liste de marque de voiture.

Lien de la doc : <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>

- Dans un premier temps, déclarer et instancier un objet de type ArrayList et un objet de type Scanner ;
- Créer 3 variables de type String et les remplir en demandant (faire apparaître un texte dans la console) à l'utilisateur de donner un nom de marque de voiture ;
- Ajouter ces 3 variables à l'ArrayList créée plus haut ;
- Supprimer la 2ème marque de voiture de l'ArrayList ;
- Afficher dans la console la marque de voiture présente dans la première cellule de l'ArrayList.

## j) Informations complémentaires

- **Documentation Java**

Java est un langage avec une documentation exemplaire.

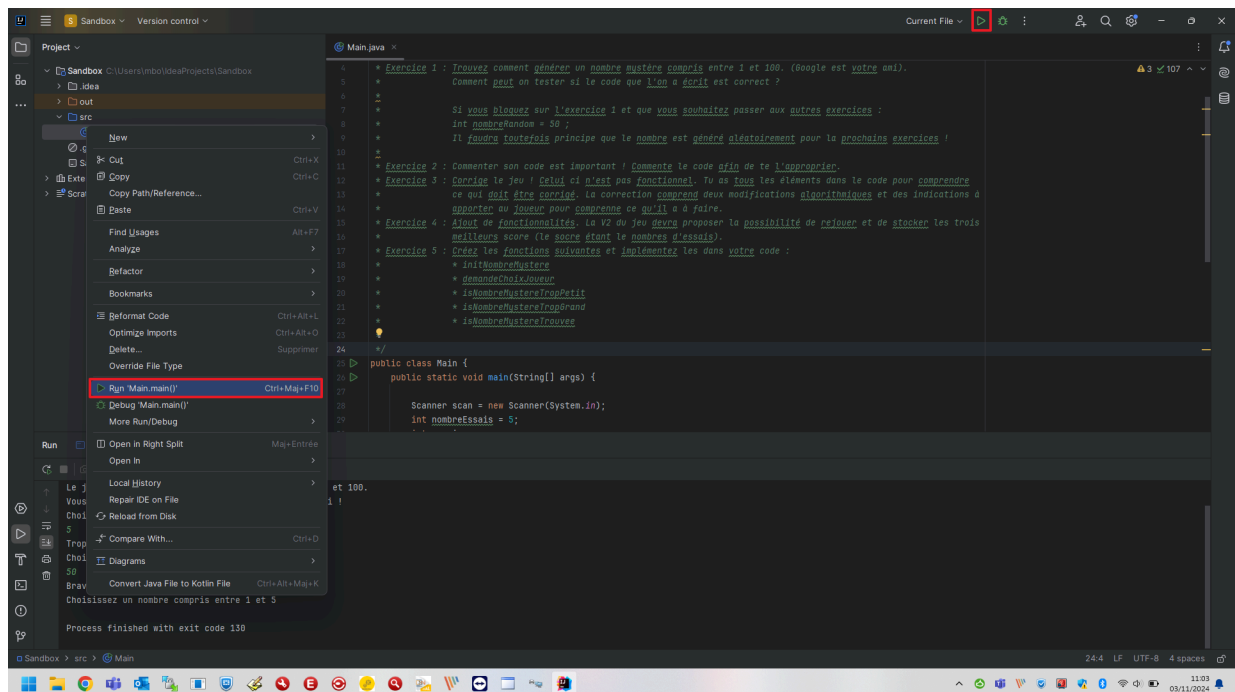
Le site d'oracle reste la meilleure des références en matière de documentation. On y retrouve une description, tous les constructeurs et toutes les méthodes avec des détails sur leur utilité, les paramètres à passer etc...

Lien de la documentation Oracle : <https://docs.oracle.com/javase/8/docs/api/>

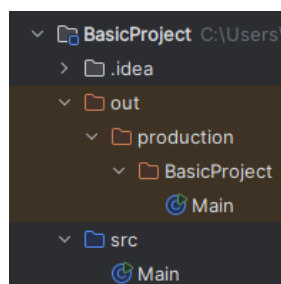
Pour trouver la documentation d'un objet, il suffit de taper dans google "java doc + le nom de la classe que vous souhaitez consulter"

- **Compilation des programmes dans IntelliJ Idea**

Afin de compiler et de lancer un programme dans IntelliJ Idea vous pouvez soit faire un clic droit sur une classe qui contient la fonction main dans l'arborescence de gauche, soit cliquer sur le bouton "play" en haut à droite, soit utiliser le raccourci clavier... entre autres méthodes.



Idea compile les programmes dans un dossier spécifique à la racine du projet. Les fichiers .class se retrouvent dans le dossier `out/production/nomProjet/`.

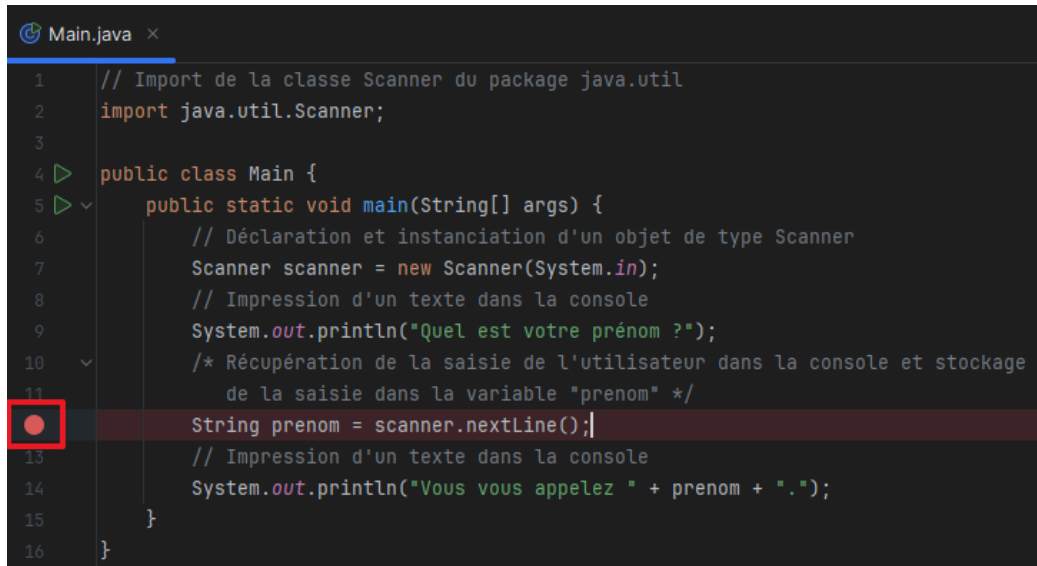




- **Debug**

L'outil de debug est très utilisé lors des développements : il permet d'arrêter l'exécution du programme à une ligne donnée. Cette pause permet de vérifier la valeur des variables à un instant T, de vérifier si le programme passe bien sur certaines lignes de code, de comprendre un comportement involontaire, etc...

Pour mettre un point d'arrêt dans Idea, il faut cliquer dans la marge sur le numéro de la ligne sur laquelle on souhaite s'arrêter :



Il faut ensuite lancer le programme via le bouton de debug :

