

Java : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

`elmouelhi.achref@gmail.com`



Plan

- 1 Introduction
- 2 Avant de commencer
- 3 Un premier projet Java
- 4 Les variables
- 5 Les opérations sur les variables
- 6 La lecture d'une saisie
- 7 Les conditions et les boucles
- 8 Les tableaux
- 9 Les constantes

Java

Java ?

- langage de programmation
 - orienté objet
 - fortement typé
- présenté officiellement en 1995 par **Sun Microsystems** (rachat par **Oracle Corporation** le 20 avril 2009)
- syntaxe très proche du C (procédural) et C++ (procédural, orienté objet)

Java

Java ?

- langage de programmation
 - orienté objet
 - fortement typé
- présenté officiellement en 1995 par **Sun Microsystems** (rachat par **Oracle Corporation** le 20 avril 2009)
- syntaxe très proche du C (procédural) et C++ (procédural, orienté objet)

Attention

Java \neq JavaScript

Java

Java, pourquoi ?

- Langage de haut niveau (pas de gestion de mémoire, pas d'allocation dynamique, pas de pointeur... comme en C et C++)
- Disposant d'une bonne documentation, des supports vidéos, plusieurs exemples sur internet
- Énorme communauté : un des langages les plus utilisés dans le monde
- Permettant de développer des programmes :
 - robustes
 - sécurisés et fiables
 - bien structurés et maintenables
 - portables : Windows, Mac OS, Linux (Write once, run everywhere ou Écrire une fois, exécuter partout)
 - ...

Quel type d'application ?

- applications consoles (JSE)
- applications du bureau (Client lourd avec Swing)
- applications web (JEE)
- applications mobiles
- web services (Jersey...)
- jeux...

Java

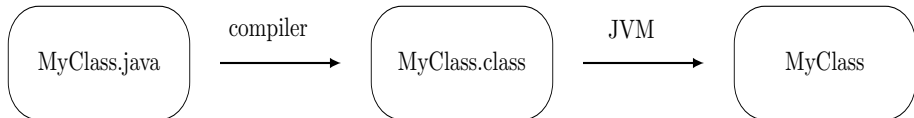
Quelques versions de Java (les plus importantes)

- Java 1 (sortie en 1996)
- Java 2 (sortie en 1998, nommée **Playground**) : Swing + collection
- Java 4 (sortie en 2002, nommée **Merlin**) : expressions régulières + parser XML (JAXP)
- Java 5 (sortie en 2004, nommée **Tiger**) : généricité, annotation + énumérations + plus besoin de convertir les types wrappers en primitifs (et inversement)
- Java 6 (sortie en 2006, nommée **Mustang**) : JAX-WS (Web services REST)
- Java 7 (sortie en 2011, nommée **Dolphin**) : `String` dans `switch`
- Java 8 (sortie en Mars 2014, nommée **Spider**) : interface fonctionnelle, méthode par défaut, expression Lambda, MapReduce pour les collections
- Java 10 (sortie en Mars 2018) : mot-clé `var`
- Java 11 (sortie en Septembre 2018) : simplifier l'exécution d'un programme en ligne de commande
- Java 12 (sortie en Mars 2019) : simplification de `switch` et `String` multi-lignes

Java

Comment ça fonctionne ?

- On écrit un programme dans un fichier `.java`
- Ensuite, le compilateur génère un fichier `.class` du même nom (contenant du bytecode)
- Puis, la machine virtuelle exécute le bytecode en le traduisant en langage natif (langage de bas niveau, ce qui assure la portabilité d'un programme java)



Java

De quoi on a besoin (le minimum) ?

- Un éditeur de texte (Bloc-notes, Notepad++, Sublime Text...)
- Un kit de développement (JDK : Java Development Kit) contenant :
 - Java Runtime Environment (JRE, incluant la machine virtuelle de Java (JVM))
 - Des bibliothèques (JSE : Java Standard Edition, JEE : Java Enterprise Edition, JME Java Micro Edition, Swing, JDBC...)
 - Des commandes permettant la création, la compilation et l'exécution d'un programme Java
 - `javac` : pour compiler
 - `java` : pour exécuter
 - `javadoc` : pour générer une documentation
 - `jar` : pour archiver

Introduction

JDK : téléchargement

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Remarque

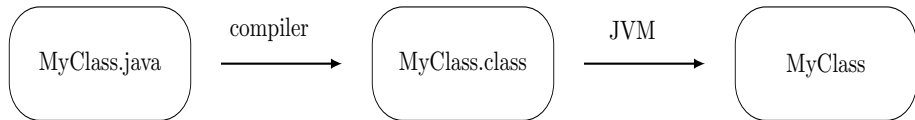
Pour lancer un programme en ligne de commande, il faut :

- aller dans Panneau de configuration, chercher Système et cliquer sur Paramètres systèmes avancés
- choisir Variables d'environnement puis dans la zone Variables utilisateur sélectionner Path et cliquer sur Modifier
- cliquer sur Nouveau puis saisir le chemin vers la **JDK** dans la zone de saisie qui a apparue
- valider

Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

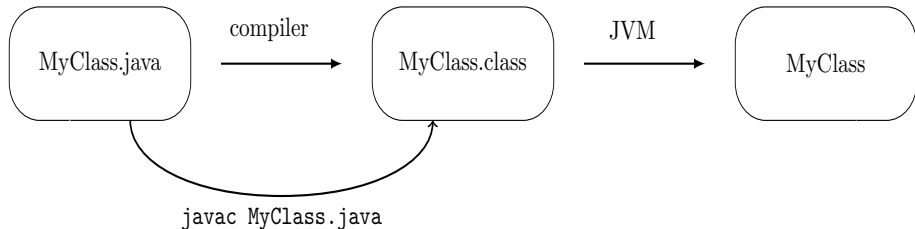
```
public class MyClass {  
  
    public static void main(String[] args) {  
        System.out.print("Hello world from console");  
    }  
}
```



Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

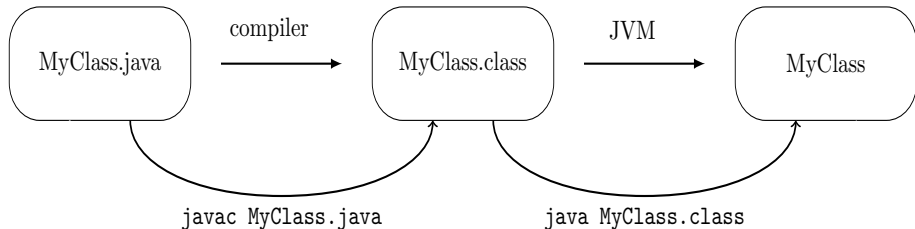
```
public class MyClass {  
  
    public static void main(String[] args) {  
        System.out.print("Hello world from console");  
    }  
}
```



Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

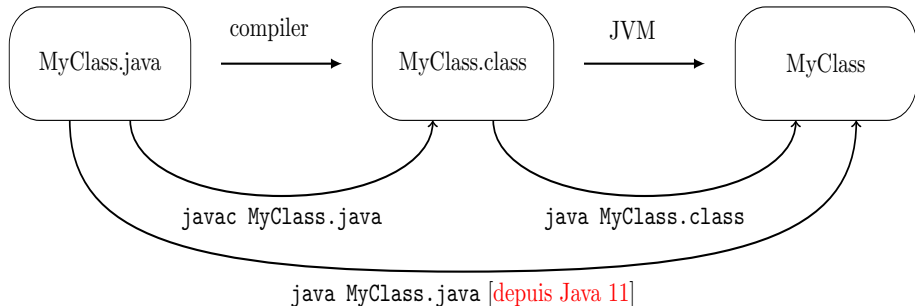
```
public class MyClass {  
  
    public static void main(String[] args) {  
        System.out.print("Hello world from console");  
    }  
}
```



Java

Créons une classe `MyClass` dans un fichier `MyClass.java`

```
public class MyClass {  
  
    public static void main(String[] args) {  
        System.out.print("Hello world from console");  
    }  
}
```



Java

Pour compiler

```
javac MyClass.java
```


Java

Pour compiler

```
javac MyClass.java
```

S'il existe plusieurs versions de JDK sur la machine

```
javac -target 8 -version 8 MyClass.java
```

Java

Pour compiler

```
javac MyClass.java
```

S'il existe plusieurs versions de JDK sur la machine

```
javac -target 8 -version 8 MyClass.java
```

Pour exécuter (ça affiche Hello world from console)

```
java MyClass
```

On peut aussi utiliser un IDE (Environnement de développement intégré)

- pour éviter d'utiliser la console et les commandes
- car un IDE intègre un compilateur lancé même pendant l'écriture du code
- pour profiter de la coloration syntaxique, l'auto-complétion, l'indentation automatique...
- pour avoir une bonne structuration du projet

Exemple d'IDE pour Java

- **Eclipse**
- Netbeans
- JDeveloper
- IntelliJ IDEA
- JBuilder
- JCreator...
- ...


Eclipse, pourquoi ?


- open-source
- écrit en Java
- multi-langage : Java, C++, PHP, Cobol, C#, JavaScript...


Introduction

Eclipse : téléchargement


```
https://www.eclipse.org/downloads/download.php?file=/oomph/epp/2019-06/R/eclipse-inst-win64.exe
```

 by Oomph


type filter text 




Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration




Eclipse IDE for Enterprise Java Developers
Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and...



Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn Integration.



Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.





Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.



eclipseinstaller by Oomph



type filter text 



Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration




Eclipse IDE for Enterprise Java Developers
Tools for Java developers creating Enterprise Java and Web applications, including a Java IDE, tools for Enterprise Java, JPA, JSF, Mylyn, Maven, Git and...



Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.



Eclipse IDE for JavaScript and Web Developers
The essential tools for any JavaScript developer, including JavaScript, HTML, CSS, XML languages support, Git client, and Mylyn.



Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, HTML, CSS and XML.

Les règles de nommage en Java

- Pour les classes et les fichiers : **Le Pascal case**
- Pour les variables, les objets et les méthodes : **Le Camel case**

Java

Les règles de nommage en Java

- Pour les classes et les fichiers : **Le Pascal case**
- Pour les variables, les objets et les méthodes : **Le Camel case**

Pour plus de détails

<https://wprock.fr/blog/conventions-nommage-programmation/>

Les instructions

- Chaque instruction se termine par ;
- Il est possible d'écrire plusieurs instructions sur une même ligne (**mais** ce n'est pas une bonne pratique)
- **Eclipse** nous facilite le formatage et l'indentation du code avec le raccourci `CTRL + Shift + F`

Comment organiser un projet Java ?

- Une classe par fichier
- Organiser les classes par package selon la sémantique
- Une classe ne peut être définie dans plusieurs fichiers (pas de classe partielle en Java)
- Il est possible de créer deux classes avec le même nom dans deux packages différents

Comment créer un projet sous **Eclipse** ?

- Aller dans `File > New > Java Project`
- Remplir le champ `Project name :` avec `FirstJavaProject` puis cliquer sur `Next`
- Valider en cliquant sur `Finish`

Java

Comment créer un projet sous **Eclipse** ?

- Aller dans `File > New > Java Project`
- Remplir le champ `Project name` : avec `FirstJavaProject` puis cliquer sur `Next`
- Valider en cliquant sur `Finish`

Que contient ce projet ?

- `JRE System Library` : l'ensemble de `.jar` indispensable pour le lancement du projet
- `src` : le répertoire qui contiendra les fichiers sources (les classes)

Comment créer une classe ?

- Aller dans `File > New > Class`
- Dans `Package`, saisir `org.eclipse.classes`
- Dans `Class`, saisir `FirstClass`
- Cocher la case `public static void main (String[] args)`
- Cliquer sur `Finish`

Java

Comment créer une classe ?

- Aller dans `File > New > Class`
- Dans `Package`, saisir `org.eclipse.classes`
- Dans `Class`, saisir `FirstClass`
- Cocher la case `public static void main (String[] args)`
- Cliquer sur `Finish`

Remarque

Si on a un package, on peut le sélectionner au moment de la création de la classe

Comment créer un package ?

- Aller dans `File > New > Package`
- Saisir le nom du package et valider

Java

Contenu de la classe `FirstClass`

```
package org.eclipse.classes;

public class FirstClass {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
    }
}
```

Java

Explication

- En java, et contrairement à plusieurs langages OO comme C++, C#, PHP..., on ne peut écrire une instruction en dehors d'une (méthode de) classe.
- En java, un fichier contient une seule classe et une classe ne peut être déclarée dans plusieurs fichiers (contrairement à C#)
- La première ligne `package org.eclipse.classes` nous informe que la classe actuelle se situe dans `src/org/eclipse/classes` dans un répertoire `FirstJavaProject` situé dans le `work-space` d'Eclipse
- Dans un projet Java, il faut qu'au moins une classe contienne la méthode `public static void main(String[] args) :` point d'entrée du projet

Java

Les commentaires

Trois types de commentaires

Java

Les commentaires

Trois types de commentaires

Commentaire mono-ligne

```
// ceci est un commentaire sur une seule ligne
```

Java

Les commentaires

Trois types de commentaires

Commentaire mono-ligne

```
// ceci est un commentaire sur une seule ligne
```

Commentaire sur plusieurs lignes

```
/* ceci est un  
commentaire  
sur trois lignes */
```

Java

Les commentaires

Trois types de commentaires

Commentaire mono-ligne

```
// ceci est un commentaire sur une seule ligne
```

Commentaire sur plusieurs lignes

```
/* ceci est un  
commentaire  
sur trois lignes */
```

Commentaire pour documentation

```
/**  
@author Achref El Mouelhi  
*/
```

Java

Pour afficher `Hello World`, on modifie la classe `FirstClass`

```
package org.eclipse.classes;

public class FirstClass {

    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

Java est un langage 100% (ou presque) orienté objet

- Pour afficher un message, il faut utiliser la classe `System`
- La classe `System` a deux objets pour l'entrée/sortie (`in/out`)
- L'objet `out` a plusieurs méthodes d'affichage comme `print()` et `println()`

Comment exécuter le programme ? (voir le résultat)

- Soit en faisant clic droit sur `FirstJavaProject` dans Package Explorer et aller dans Run As > Java Application
- Soit en faisant clic droit sur la classe contenant `public static void main()` (ici `FirstClass`) dans le panneau central et aller dans Run As > Java Application
- Soit en cliquant sur le triangle vert dans la liste de raccourci

Java

Où voir le résultat ?

- Dans la console d'Eclipse
- Si la console n'est pas visible, aller dans `Window > Show View > Other...`, saisir `console` et la sélectionner puis valider.

Java

Où voir le résultat ?

- Dans la console d'Eclipse
- Si la console n'est pas visible, aller dans `Window > Show View > Other...`, saisir `console` et la sélectionner puis valider.

Où sont les `.class` générés ?

- Dans le `work-space`, aller voir dans le répertoire portant le nom du projet (ici `FirstJavaProject`)
- Dans `org/eclipse/classes`, un fichier `FirstClass.class` a été généré.

Une variable ?

- Un pointeur vers une zone mémoire
- Permettant de stocker une ou plusieurs données
- Pouvant avoir plusieurs valeurs différentes dans un programme

Java

Une variable ?

- Un pointeur vers une zone mémoire
- Permettant de stocker une ou plusieurs données
- Pouvant avoir plusieurs valeurs différentes dans un programme

Java est un langage de programmation fortement typé

- Il faut préciser le type de chaque variable
- Une variable peut avoir de valeurs différentes mais ne peut changer de type.

Java

Déclarer une variable

```
type nomVariable;
```

Déclarer une variable

```
type nomVariable;
```

Deux choix possibles pour typer les variables

- Types simples (primitifs) : commencent par une lettre en minuscule
- Types objets : commencent par une lettre en majuscule

Principaux types primitifs en Java

- `byte` : entier codé sur 1 octet (entre -128 et 127)
- `short` : entier codé sur 2 octets (entre -32 768 et 32 767)
- `int` : entier codé sur 4 octets (entre -2 147 483 648 et 2 147 483 647)
- `long` : entier codé sur 8 octets (entre -9 223 372 036 854 775 808 et +9 223 372 036 854 775 807)
- `float` : nombre à virgule codé sur 4 octets
- `double` : nombre à virgule codé sur 8 octets
- `boolean` : soit `true` soit `false` (1 octet)
- `char` : caractère codé sur 2 octet situé entre deux ' '

Java

Principaux types primitifs en Java

- `byte` : entier codé sur 1 octet (entre -128 et 127)
- `short` : entier codé sur 2 octets (entre -32 768 et 32 767)
- `int` : entier codé sur 4 octets (entre -2 147 483 648 et 2 147 483 647)
- `long` : entier codé sur 8 octets (entre -9 223 372 036 854 775 808 et +9 223 372 036 854 775 807)
- `float` : nombre à virgule codé sur 4 octets
- `double` : nombre à virgule codé sur 8 octets
- `boolean` : soit `true` soit `false` (1 octet)
- `char` : caractère codé sur 2 octet situé entre deux ' '

Pas de type primitif pour les chaînes de caractère.

Java

Exemple

```
int x;
```

Java

Exemple

```
int x;
```

Déclarer et initialiser une variable

```
int x = 5;
```

Java

Exemple

```
int x;
```

Déclarer et initialiser une variable

```
int x = 5;
```

Ceci est une erreur

```
byte x = 130;
```

Java

Exemple

```
int x;
```

Déclarer et initialiser une variable

```
int x = 5;
```

Ceci est une erreur

```
byte x = 130;
```

Pour convertir le contenu d'une variable (le `cast` pour les types compatibles)

```
int x = 100;  
byte z = (byte) x;  
System.out.println(z); // affiche 100
```

Attention aux valeurs qui dépassent l'intervalle

```
int x = 200;  
byte z = (byte) x;  
System.out.println(z); // affiche -56
```

Les emballages (wrapper) de types primitifs (types objets) en Java

- Byte **pour** byte
- Short **pour** short
- Long **pour** long
- Integer **pour** int
- Float **pour** float
- Double **pour** double
- Boolean **pour** boolean
- Character **pour** char

Java

Les emballages (wrapper) de types primitifs (types objets) en Java

- Byte **pour** byte
- Short **pour** short
- Long **pour** long
- Integer **pour** int
- Float **pour** float
- Double **pour** double
- Boolean **pour** boolean
- Character **pour** char

Il existe aussi plusieurs autres types objets : String, Date...

Java

Exemple d'utilisation d'une chaîne de caractère

```
String string = new String();  
string = "bonjour";  
System.out.println(string); // affiche bonjour
```

Java

Exemple d'utilisation d'une chaîne de caractère

```
String string = new String();  
string = "bonjour";  
System.out.println(string); // affiche bonjour
```

On peu aussi faire

```
String string = new String("bonjour");  
System.out.println(string); // affiche bonjour
```

Java

Exemple d'utilisation d'une chaîne de caractère

```
String string = new String();  
string = "bonjour";  
System.out.println(string); // affiche bonjour
```

On peu aussi faire

```
String string = new String("bonjour");  
System.out.println(string); // affiche bonjour
```

Ou encore

```
String string = "bonjour";  
System.out.println(string); // affiche bonjour
```

Java

Exemple d'utilisation d'une chaîne de caractère

```
String string = new String();  
string = "bonjour";  
System.out.println(string); // affiche bonjour
```

On peu aussi faire

```
String string = new String("bonjour");  
System.out.println(string); // affiche bonjour
```

Ou encore

```
String string = "bonjour";  
System.out.println(string); // affiche bonjour
```

Une chaîne de caractère doit être située entre deux "contenu"

Pourquoi les types primitifs et les types objets ?

- Les types primitifs sont moins coûteux en mémoire
- Les types objets offrent plusieurs méthodes à appliquer sur les valeurs : conversion, nombre de caractère...
- Toutes les classes wrappers contiennent une méthode (`TypeObjet.parseTypeSimple(string)`) qui permet de convertir une chaine de caractère en type primitif de cette classe

Java

Quelques méthodes de la classe `String`

- `length()` : retourne le nombre de caractère de la chaîne.
- `indexOf(x)` : retourne l'indice de la première occurrence de la valeur de `x` dans la chaîne, -1 sinon.
- `contains()` : retourne `true` si la chaîne contient `x`, `false` sinon.
- `charAt(i)` : retourne le caractère d'indice `i` dans la chaîne.
- `substring(i, j)` : permet d'extraire une sous-chaîne de la chaîne à partir de l'indice `i` jusqu'au l'indice `j - 1`
- `equals(str)` : permet de comparer la chaîne à `str` et retourne `true` en cas d'égalité, `false` sinon.
- `replace(old, new)` : permet de remplacer toute occurrence de la chaîne `old` dans la chaîne courante par `new` et retourne la nouvelle chaîne
- ...

Java

Exemple : `replace(old,new)`

```
String string = "bonjour les bons jours";  
String string2 = string.replace("jour", "soir");  
System.out.println(string2);  
// bonsoir les bons soirs
```

Java

Exemple : `replace(old,new)`

```
String string = "bonjour les bons jours";  
String string2 = string.replace("jour", "soir");  
System.out.println(string2);  
// bonsoir les bons soirs
```

Exemple : `indexOf(str,fromIndex)`

```
String string = "bonjour les bons jours";  
int pos = string.indexOf("bon", 5);  
System.out.println(pos);  
// affiche 12
```


Java

Exemple de conversion d'une chaîne de caractère

```
String string ="2";  
byte z = Byte.parseByte(string);  
System.out.println(z); // affiche 2
```

Java

Exemple de conversion d'une chaîne de caractère

```
String string = "2";  
byte z = Byte.parseByte(string);  
System.out.println(z); // affiche 2
```

Ceci déclenche une exception (arrêt d'exécution)

```
String string = "a";  
byte z = Byte.parseByte(string);  
System.out.println(z);
```

Le résultat

```
Exception in thread "main" java.lang.  
    NumberFormatException: For input string: "a"
```

Exemple de conversion de Integer en String

```
Integer x = 2;  
String string = x.toString();  
System.out.println(string); // affiche 2
```

Java

Quatre méthodes pour convertir `int` en `String`

```
int y = 3;  
String chaine = ((Integer) y).toString();  
System.out.println(chaine); // affiche 3
```

```
int z = 4;  
String str = Integer.toString(z);  
System.out.println(str); // affiche 4
```

```
int t = 5;  
String s = String.valueOf(t);  
System.out.println(s); // affiche 5
```

```
int u = 6;  
String s = "" + u;  
System.out.println(s); // affiche 6
```

Attention, le `cast` entre deux types incompatibles n'est pas autorisé

```
int z = 4;  
String str =(String) z;  
System.out.println(str);
```

Pas besoin de convertir d'un type simple vers son wrapper (ou inversement)

```
Integer j = 2;  
int k = j;  
System.out.println(k); // affiche 2  
  
int l = 3;  
Integer m = l;  
System.out.println(m); // affiche 3
```

Pour les variables numériques (`int`, `float`...)

- `=` : affectation
- `+` : addition
- `-` : soustraction
- `*` : multiplication
- `/` : division
- `%` : reste de la division

Quelques raccourcis

- $i = i + 1 \Rightarrow i++;$
- $i = i - 1 \Rightarrow i--;$
- $i = i + 2 \Rightarrow i+=2;$
- $i = i - 2 \Rightarrow i-=2;$

Exemple de post-incrémentation

```
int i = 2;  
int j = i++;  
System.out.println(i); // affiche 3  
System.out.println(j); // affiche 2
```

Java

Exemple de post-incrémentation

```
int i = 2;  
int j = i++;  
System.out.println(i); // affiche 3  
System.out.println(j); // affiche 2
```

Exemple de pre-incrémentation

```
int i = 2;  
int j = ++i;  
System.out.println(i); // affiche 3  
System.out.println(j); // affiche 3
```

L'opérateur + pour concaténer deux chaînes de caractère

```
String string = "bon";  
String string2 = "jour";  
System.out.println(string + string2);  
// affiche bonjour
```

Pour lire une valeur saisie par l'utilisateur dans la console

- il faut utiliser la classe `Scanner`
- il faut préciser le type de la valeur à récupérer

Java

Instancier la classe `Scanner`

```
Scanner scanner = new Scanner(System.in);
```

Java

Instancier la classe `Scanner`

```
Scanner scanner = new Scanner(System.in);
```

La classe `Scanner` **est signalée en rouge, il faut l'importer**

```
import java.util.Scanner;
```

Java

Instancier la classe `Scanner`

```
Scanner scanner = new Scanner(System.in);
```

La classe `Scanner` est signalée en rouge, il faut l'importer

```
import java.util.Scanner;
```

Exemple de lecture d'un entier

```
int i = scanner.nextInt();
```

Java

Instancier la classe `Scanner`

```
Scanner scanner = new Scanner(System.in);
```

La classe `Scanner` est signalée en rouge, il faut l'importer

```
import java.util.Scanner;
```

Exemple de lecture d'un entier

```
int i = scanner.nextInt();
```

Fermer le `scanner`

```
scanner.close();
```


Remarques

- Pour lire une chaîne de caractère, il faut utiliser la méthode `next()` ou `nextLine()`
- Pour lire un nombre réel, il faut utiliser la méthode `nextFloat()`
- ...

Java

Exemple avec `next()`

```
Scanner scanner = new Scanner(System.in);  
// si on saisit "bonjour tout le monde"  
String string = scanner.next();  
System.out.println(string);  
// affiche bonjour
```

Java

Exemple avec `next()`

```
Scanner scanner = new Scanner(System.in);  
// si on saisit "bonjour tout le monde"  
String string = scanner.next();  
System.out.println(string);  
// affiche bonjour
```

Exemple avec `nextLine()`

```
Scanner scanner = new Scanner(System.in);  
// si on saisit "bonjour tout le monde"  
String string = scanner.nextLine();  
System.out.println(string);  
// affiche bonjour tout le monde
```

Java

À chaque appel à `next()`, on récupère le token suivant

```
Scanner scanner = new Scanner(System.in);  
// si on saisit "bonjour tout le monde"  
String string = scanner.next();  
String s = scanner.next();  
System.out.println(s);  
// affiche tout
```

Java

À chaque appel à `next()`, on récupère le token suivant

```
Scanner scanner = new Scanner(System.in);  
// si on saisit "bonjour tout le monde"  
String string = scanner.next();  
String s = scanner.next();  
System.out.println(s);  
// affiche tout
```

Le code suivant déclenche une exception car on ne peut affecter la chaîne "tout" à un entier

```
Scanner scanner = new Scanner(System.in);  
// si on saisit "bonjour tout le monde"  
String string = scanner.next();  
int v = scanner.nextInt();  
System.out.println(v);
```

Il n'existe pas une méthode spécifiques aux caractères, mais on peut faire

```
Scanner scanner = new Scanner(System.in);  
String string = scanner.next();  
char c = scanner.next().charAt(0);  
System.out.println(c);
```

Java

On peut aussi utiliser les méthodes statiques de la classe `Math`

- `Math.abs(x)` : retourne la valeur absolue de `x`
- `Math.pow(x, y)` : retourne la `x` puissance `y`
- `Math.max(x, y)` : retourne le max de `x` et `y`
- `Math.min(x, y)` : retourne le min de `x` et `y`
- `Math.sqrt(x)` : retourne la racine carré de `x`
- `Math.floor(x)`, `Math.ceil(x)` et `Math.round(x)` :
retournent l'arrondi de `x`
- `Math.random()` : retourne une valeur aléatoire entre 0 et 1
- ...

Java

Exemple avec `Math.floor(x)`, `Math.ceil(x)` **et**
`Math.round(x)`

```
System.out.println(Math.round(1.9)); // affiche 2
System.out.println(Math.round(1.5)); // affiche 2
System.out.println(Math.round(1.4)); // affiche 1
System.out.println(Math.ceil(1.9)); // affiche 2
System.out.println(Math.ceil(1.5)); // affiche 2
System.out.println(Math.ceil(1.4)); // affiche 2
System.out.println(Math.floor(1.9)); // affiche 1
System.out.println(Math.floor(1.5)); // affiche 1
System.out.println(Math.floor(1.4)); // affiche 1
```


Java

Tester une condition

```
if (condition1) {  
    ...  
}  
[  
else if (condition2) {  
    ...  
}  
...  
else {  
    ...  
}  
]
```

Opérateurs logiques

- `&&` : **et**
- `||` : **ou**
- `!` : **non**

Java

Opérateurs logiques

- `&&` : et
- `||` : ou
- `!` : non

Tester plusieurs conditions (en utilisant des opérateurs logiques)

```
if (condition1 && !condition2 || condition3) {  
    ...  
}  
[else ...]
```

Java

Opérateurs logiques

- `&&` : et
- `||` : ou
- `!` : non

Tester plusieurs conditions (en utilisant des opérateurs logiques)

```
if (condition1 && !condition2 || condition3){  
    ...  
}  
[else ...]
```

Pour les conditions, on utilise des opérateurs de comparaisons

Opérateurs de comparaison

- `==` : pour tester l'égalité
- `!=` : pour tester l'inégalité
- `>` : supérieur à
- `<` : inférieur à
- `>=` : supérieur ou égal à
- `<=` : inférieur ou égal à

Opérateurs de comparaison

- == : pour tester l'égalité
- != : pour tester l'inégalité
- > : supérieur à
- < : inférieur à
- >= : supérieur ou égal à
- <= : inférieur ou égal à

En java, on ne peut comparer deux valeurs de type incompatible

Java

Structure conditionnelle avec `switch`

```
int x = 5;
switch (x) {
    case 1:
        System.out.println("un");
        break;
    case 2:
        System.out.println("deux");
        break;
    case 3:
        System.out.println("trois");
        break;
    default:
        System.out.println("autre");
}
```

La variable dans `switch` peut être

- de types simples : `int`, `short`, `byte` et `char`
- de type wrappers correspondants : `Integer`, `Short`, `Byte` et `Character`
- de type énumération depuis Java 6
- de type chaîne de caractère depuis Java 7

Java

Considérons l'exemple suivant

```
String x = "2";  
switch (x) {  
    case "1":  
        System.out.println("un");  
        break;  
    case "2":  
        System.out.println("deux");  
        break;  
    case "3":  
        System.out.println("trois");  
        break;  
    default:  
        System.out.println("autre");  
}
```

Simplifions l'écriture avec l'expression ternaire

```
int x = 2;  
String type = (x%2==0) ? "pair" : "impair";  
System.out.println(type); // affiche pair
```

Java

Boucle `while`

```
while (condition[s]) {  
    ...  
}
```

Java

Boucle `while`

```
while (condition[s]) {  
    ...  
}
```

Boucle `do ... while`

```
do {  
    ...  
}  
while (condition[s]);
```

Java

Boucle `while`

```
while (condition[s]) {  
    ...  
}
```

Boucle `do ... while`

```
do {  
    ...  
}  
while (condition[s]);
```

Attention aux boucles infinies, vérifier que la condition d'arrêt sera bien atteinte après un certain nombre d'itérations.

Boucle `for`

```
for (initialisation; condition[s]; incrementation) {  
    ...  
}
```

Java

Boucle `for`

```
for (initialisation; condition[s]; incrementation) {  
    ...  
}
```

Attention aux boucles infinies si vous modifiez la valeur du compteur à l'intérieur de la boucle.

Java

En java, les tableaux sont statiques

- Tous les éléments doivent avoir le même type
- Il faut préciser une taille qu'on ne peut dépasser

Java

En java, les tableaux sont statiques

- Tous les éléments doivent avoir le même type
- Il faut préciser une taille qu'on ne peut dépasser

Déclaration d'un tableau : syntaxe

```
type [] nomTableau = new type[taille];
```

Java

En java, les tableaux sont statiques

- Tous les éléments doivent avoir le même type
- Il faut préciser une taille qu'on ne peut dépasser

Déclaration d'un tableau : syntaxe

```
type [] nomTableau = new type[taille];
```

Déclaration d'un tableau d'entier de taille 2 : exemple

```
int [] tab = new int[2];
```

Java

Affectation de valeurs aux cases de tableau

```
tab[0] = 5;
```

```
tab[1] = 3;
```

Java

Affectation de valeurs aux cases de tableau

```
tab[0] = 5;  
tab[1] = 3;
```

Ceci déclenche une exception

```
tab[2] = 10;
```

L'exception

```
Exception in thread "main" java.lang.  
    ArrayIndexOutOfBoundsException: 2  
        at org.eclipse.classes.FirstClass.main(  
            FirstClass.java:11)
```

Java

On peut faire une déclaration + une initialisation

```
int [] tab = {5,3};
```

Comme si la taille du tableau a été fixée à deux

Java

On peut faire une déclaration + une initialisation

```
int [] tab = {5,3};
```

Comme si la taille du tableau a été fixée à deux

Donc, ceci déclenche aussi une exception

```
tab[2] = 10;
```

L'exception

```
Exception in thread "main" java.lang.  
    ArrayIndexOutOfBoundsException: 2  
        at org.eclipse.classes.FirstClass.main(  
            FirstClass.java:11)
```

Pour parcourir le tableau

```
for (int i=0; i<tab.length; i++) {  
    System.out.println(tab[i]);  
}
```

Pour parcourir le tableau

```
for (int i=0; i<tab.length; i++) {  
    System.out.println(tab[i]);  
}
```

Où encore la version simplifiée (foreach)

```
for (int i : tab) {  
    System.out.println(i);  
}
```


Java

Une constante ?

- c'est un élément qui ne peut changer de valeur

Java

Une constante ?

- c'est un élément qui ne peut changer de valeur

Pour déclarer une constante

- il faut utiliser le mot-clé `final`

Java

Une constante ?

- c'est un élément qui ne peut changer de valeur

Pour déclarer une constante

- il faut utiliser le mot-clé `final`

Déclaration d'une constante

```
final double PI = 3.1415;
```

Java

Une constante ?

- c'est un élément qui ne peut changer de valeur

Pour déclarer une constante

- il faut utiliser le mot-clé `final`

Déclaration d'une constante

```
final double PI = 3.1415;
```

L'instruction suivante ne peut être acceptée

```
PI = 5;
```