



ENSAH



جامعة عبد المالك السعدي  
Université Abdelmalek Essadi

# Les bases du langage Java

## Langage Java

**2<sup>ème</sup> Année Génie Informatique**

**Ecole Nationale des Sciences Appliquées – Al Hoceima**

Prof A. Bahri  
abahri@uae.ac.ma

A.U 2020/2021

# Organisation du cours

## Les chapitres essentiels

- ☐ Introduction à JAVA: compilation/exécution, la syntaxe et les éléments de bases de Java, ...
- ☐ L'orientée objet en JAVA
- ☐ La gestion des exceptions
- ☐ Les flux
- ☐ L'accès aux bases de données
- ☐ Les Threads
- ☐ Les collections
- ☐ Programmation générique
- ☐ ...

# JAVA c'est quoi ?

- ❑ Une technologie développée par SUN Microsystems (acheté par Oracle en 2009) en 1995
  - un langage de programmation
  - une plate-forme, environnement logiciel dans lequel les programmes s'exécutent
  
- ❑ Évolution gérée par JavaSOFT (SunSoft) :
  - 1991-1995 sortie de la première version de **JDK (Java Development Kit)**

# JDK : Java Development Kit

- ❑ Le JDK est l'ensemble des programmes nécessaires pour le développement d'applications Java.
- ❑ Il regroupe ainsi les programmes:
  - `javac.exe`,
  - `java.exe`,
  - `appletviewer` pour exécuter les applets,
  - ainsi que d'autres classes et utilitaires de développements.

# Versions JAVA

❑ Les versions se succédèrent alors :

- 1996 : JDK 1.0
- 1997 : JDK 1.1
- 1998 : JDK 1.2, appelé Java 2
- 2000 : JDK 1.3
- 2002 : JDK 1.4
- 2004 : JDK 1.5, appelé Java 5
- 2007 : JDK 1.6, appelé Java 6.
- 2011 : JDK 1.7, appelé Java 7.
- 2014: JDK 1.8, appelé Java 8.
- ....
- 2018: JDK 1.11, appelé Java 11.
- etc.

# Java Aujourd'hui

➡ 3 environnements d'exécutions différents:

- ❑ Java ME (Micro Edition) pour PDA, téléphone Android
- ❑ Java SE (Standard Edition) pour desktop
- ❑ Java EE (Entreprise Edition) pour serveur
  - Servlet/JSP/JSTL/JSF
  - JTA/JTS, EJB,
  - JMS,
  - JavaMail, etc.

# Papa et Maman de Java

## ❑ SmallTalk :

- Tout est objet (même if)
- **Machine Virtuelle**
- etc.

## ❑ C/C++ :

- Ecriture du code {, /\*, //
- etc.

# La notion de machine virtuelle

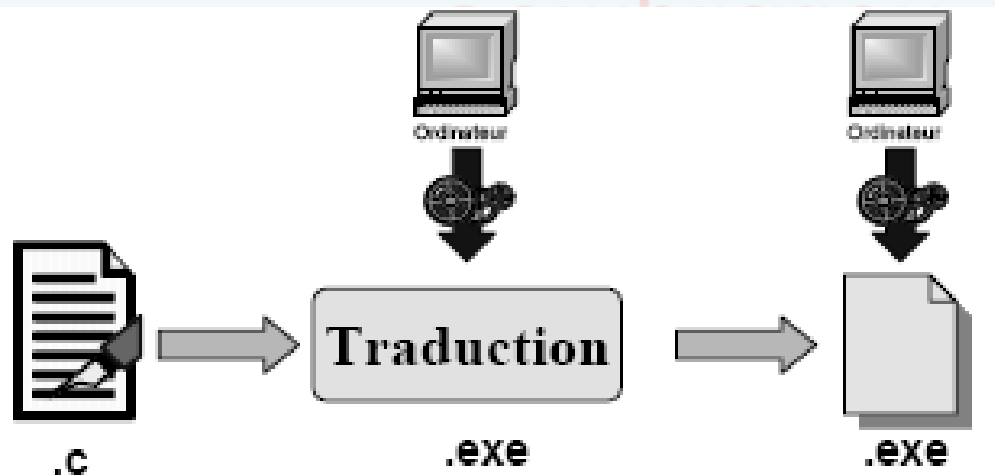
## -Compilateur et interpréteur-

- ❑ Les Compilateurs et les Interpréteurs sont des **traducteurs** de langage de programmation de haut niveau en séries d'instructions-machine directement exécutables par l'ordinateur.
- ❑ **le compilateur** traduit les programmes dans leur ensemble : **tout le programme** doit être fourni en bloc au compilateur pour la traduction.
- ❑ **l'interpréteur** traduit les programmes **instruction par instruction** dans le cadre d'une interaction continue avec l'utilisateur



# La notion de machine virtuelle

## - Compilation -



### ❑ Avantages

- Rapidité

### ❑ Inconvénient

- Manque de portabilité
- Rigidité

# La notion de machine virtuelle

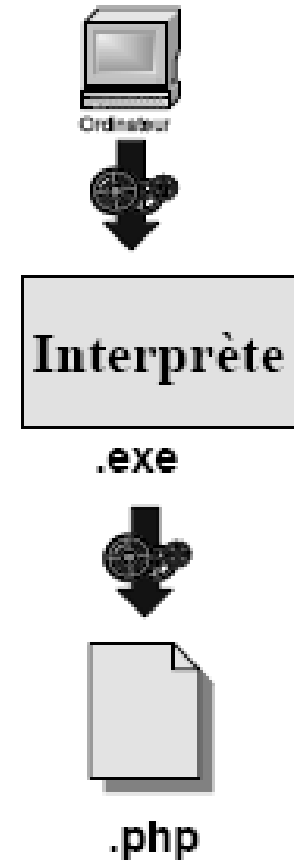
## - Interprétation -

### □ Avantages

- Portabilité
- Souplesse

### □ Inconvénient

- Lenteur



# La notion de machine virtuelle

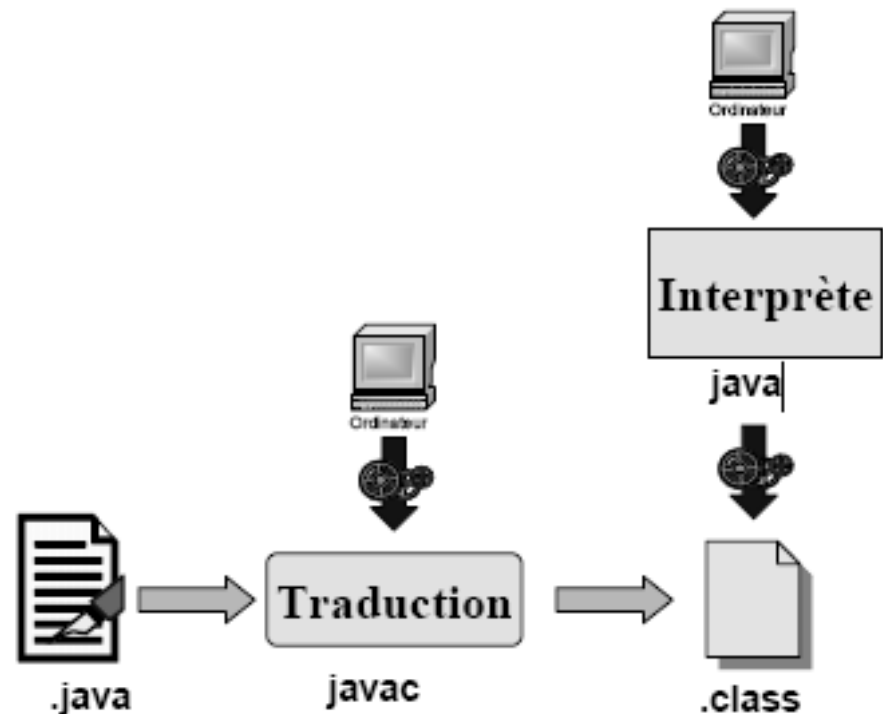
## - Machine virtuelle -

### □ Avantages

- Portabilité
- Souplesse

### □ Inconvénient

- lenteur



□ **Java utilise les deux (traducteur et interpréteur )!**

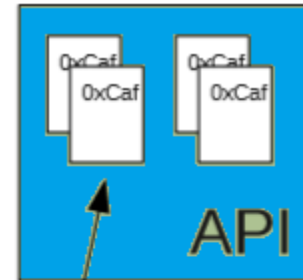
# La notion de machine virtuelle -compilation-

- ❑ Le compilateur convertit le programme source en un programme exécutable sur une « machine virtuelle »

Code en Unicode



Compilateur



- ❑ Le code est compilé dans un code intermédiaire (**bytecode**)

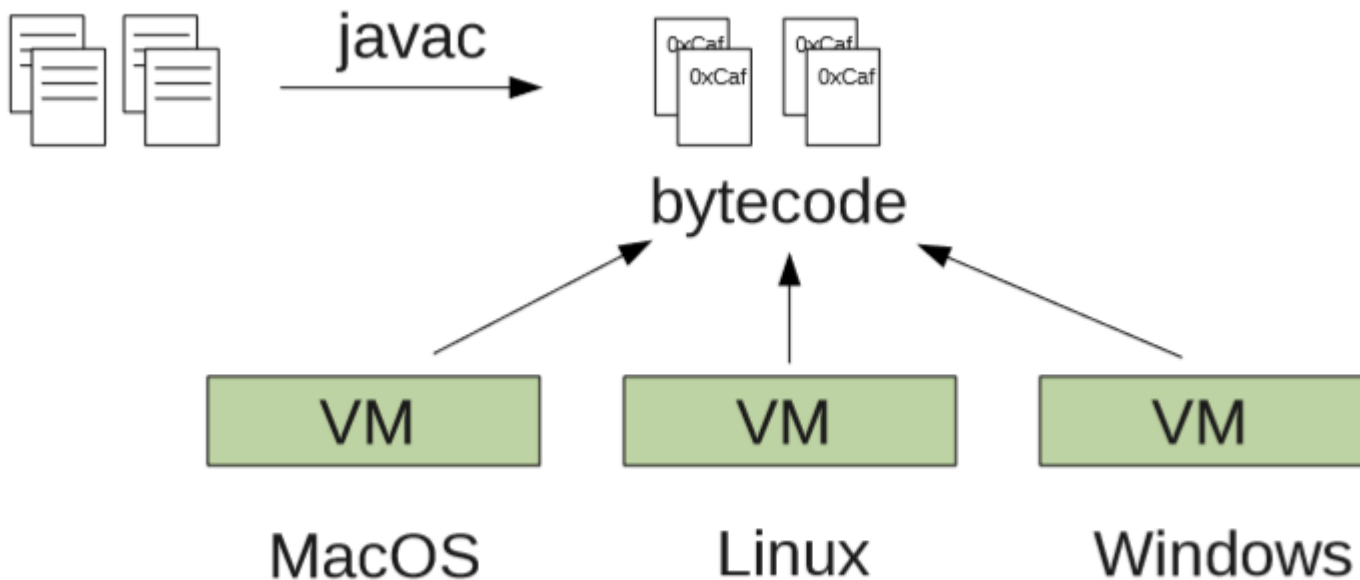
- ❑ **La Machine Virtuelle (VM)**  
charge les classes et interprète le code



↓  
Plateforme + OS

# Le byte-code

- ❑ Il assure la portabilité entre différents environnements (machine/OS)



Et Solaris, HP-UX, BSD etc...

# JAVA : write once, run anywhere

- ❑ La compilation d'un programme ne génère pas d'instructions spécifiques pour la machine sur laquelle vous travaillez.
- ❑ Mais du **bytecode** Java, qui sont des instructions pour la **machine virtuelle Java (JVM)**
- ❑ Ce qui signifie que si vous disposez d'une **JVM** (Windows, Unix, Android, IOS, navigateur internet,...), elle peut exécuter le **bytecode**

# Caractéristiques de Java

- ❑ Langage interprété : source compilé en pseudo-code (byte code), puis exécuté par un interpréteur Java (JVM)
- ❑ Les programmes java sont multi plateforme
- ❑ Langage de programmation orienté objet
- ❑ Langage simple : éléments mal exploités sont abandonnés (pointeurs, héritage multiple, surcharge des opérateurs, ...)
- ❑ Langage fortement typé
- ❑ Langage haute sécurité
- ❑ Java est utilisé pour créer des programmes et des Applets

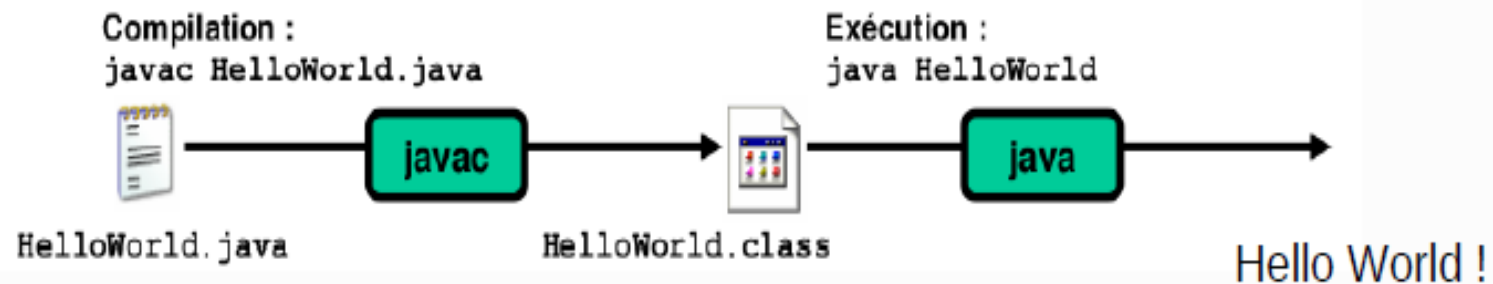
# Outils pour développer

- ❑ Un Editeur de texte ASCII: on peut utiliser un simple éditeur comme **Edit** de Dos ou **Notepad** de windows mais il est préférable d'utiliser un éditeur conçu pour la programmation java exemples: **Ultraedit**, **JCreator** ....
- ❑ Environnement de développement intégré : On peut également utiliser des environnement visuels et graphiques qui simplifient énormément la tâche du programmeur. Exemples : **Visual Café**, **Jbuilder**, **Eclipse**, **NetBeans**, etc...
- ❑ Pour tester les applets java, vous aurez besoin d'un navigateur web ex. **Internet Explorer** ou **Netscape**. Sinon, vous pouvez utiliser **Appletviewer.exe** du JDK.



# Premier programme java

```
public class HelloWorld {  
    public static void main(String [] args){  
        System.out.println("Hello World !");  
    }  
}
```



# Premier programme java

- ❑ Lancer un éditeur de texte ASCII (exemple Bloc note ou Ultraedit )
- ❑ Ecrire le code source de ce programme.
- ❑ Enregistrer ce fichier dans le répertoire c:\program files\java\jdk11\bin sous le nom **HelloWord.java**
- ❑ Compiler ce programme sur ligne de commande Dos :  
c:\program files\java\jdk11\bin >**javac HelloWord.java**
- ❑ Corriger les Erreurs de compilation
- ❑ Exécuter le programme sur ligne de commande  
c:\program files\java\jdk11\bin >java HelloWord

# Premier programme java

Remarques:

- ❑ Tout code Java doit être défini à l'intérieure d'une classe:  
EX:public class HelloWorld
- ❑ Le nom du fichier java doit être le même que celui de la classe qui contient la fonction principale main.
- ❑ Pour compiler le programme source, il faut faire appel au programme javac.exe qui se trouve dans le dossier c:\program files\java\jdk11\bin.
- ❑ Pour rendre accessible ce programme depuis n 'importe quel répertoire, il faut ajouter la commande : path c:\program files\java\jdk11\bin dans le fichier autoexec.bat.
- ❑ Après compilation du programme test.java, il y a génération du fichier test.class qui représente le ByteCode de programme.
- ❑ Pour exécuter ce programme en byte code, il faut faire appel au programme java.exe qui représente l'interpréteur du bytecode.

# Les primitives

Java dispose des primitives (types) suivantes :

<b><i>Primitive</i></b>	<b><i>Taille</i></b>
<b>char</b>	<b>16 bits</b>
<b>byte</b>	<b>8 bits</b>
<b>short</b>	<b>16 bits</b>
<b>int</b>	<b>32 bits</b>
<b>long</b>	<b>64 bits</b>
<b>float</b>	<b>32 bits</b>
<b>double</b>	<b>64 bits</b>
<b>boolean</b>	<b>1 bit</b>
<b>void</b> -	<b>0 bit</b>

# Utilisation des primitives

- ❑ Les primitives sont utilisées de façon très simple. Elles doivent être déclarées, par exemple :
  - `int i;`
  - `char c;`
  - `double v;`
  - `boolean fini;`
  
- ❑ les primitives peuvent être initialisées en même temps que la déclaration.
  - `int i = 12;`
  - `char c = 'a';`
  - `v = 23456.3456;`
  - `boolean fini = true;`

# Utilisation des primitives

**Remarque** (Valeurs par défaut des primitives):

- ❑ Toutes les primitives de type numérique sont initialisées à la valeur 0.
- ❑ Le type `boolean` est initialisé à la valeur `false`.

# Conversion de types

En java, on dispose de trois types de conversions :

## ❑ Conversion implicite

```
int i=999999999;  
float f=i; // conversion implicite
```

## ❑ Conversion explicite (cast)

```
int i=999999999;  
byte b=(byte) i; // conversion explicite  
b=i; // erreur de compilation
```

## ❑ Conversion en utilisant des méthodes

```
String nombre="120";  
int resultat=Integer.parseInt(nombre);
```

# Conversion implicite de types

□ Conversion implicite est aussi appelé conversion élargissante (pas de perte d'information). Elle permet de passer de :

- byte : short, int, long, float, double
- short: int, long, float, double
- char : int, long, float, double
- int : long, float, double
- long : float, double
- float : double



# Conversion explicite de types

□ La conversion explicite est appelée aussi conversion rétrécissante (perte d'informations)

- byte: char
- short: char, byte
- char: short, byte
- int: char, short, byte
- long: int, char, short, byte
- float: long, int, char, short, byte
- double: float, long, int, char, short, byte

# Le type String

- ❑ Le type de données String n'est pas un type élémentaire en Java, c'est une classe.
- ❑ Une chaîne est un objet qui n'est utilisable qu'à travers les méthodes de la classe String.
- ❑ La définition de la classe String se trouve dans package:

***import java.lang.String ;***

# Déclaration d'une chaîne

## ❑ Déclaration :

- `String str;`

## ❑ Initialisation :

- `str = new String("Bonjour");`

## ❑ Déclaration avec initialisation :

- `String str = new String("abcdef");`

# Déclaration d'une chaîne

□ Puisque les chaînes de caractères sont très utilisées, on peut les déclarer comme de simples variables. Par exemple :

- `String x;`
- `x="Bonjour";`
- `String y="Bonjour";`

# Manipulation du type String

- `int length()`: longueur d'une chaîne
- `String concat(String)`: concaténation de deux chaînes
- `char charAt(int)`: caractère situé à une position donnée
- `int indexOf(String)`: 1<sup>er</sup> indice d'une sous chaîne
- `int lastIndexOf(String)`: dernier indice d'une sous chaîne
- `boolean equals(String)`: compare deux chaînes
- `char[] toCharArray()`: convertir une chaîne en tableau
- `String toUpperCase()`: convertir en majuscule
- `String toLowerCase()`: convertir en minuscule
- `String substring(int,int)`: sous chaîne entre deux indices
- `String replace(char,char)`: remplace un caractère par un autre

# Tableaux en Java

- ❑ Regroupent des données de même type
- ❑ Sont gérés par Java comme des objets
- ❑ Peuvent être à une ou plusieurs dimensions
  
- ❑ Deux types
  - Tableaux statiques (Array)
  - Tableaux dynamiques (Vector)

# Déclaration de tableaux statique

## ❑ Déclaration:

- `int [] table1;`
- `char [] table2;`
- `float [] table3;`
- `String [] tableStr;`

## ❑ Déclaration avec définition explicite de taille :

- `int [] table1 = new int [5];`
- `char [] table2 = new char [12];`
- `float [] table3 = new float [8];`
- `String [] tableStr = new String [9];`

# Déclaration de tableaux statique

❑ Déclaration et initialisation d'un tableau avec définition implicite de taille :

- `int [] table1 = {17,-9,4,3,57};`
- `char[] table2 = {'a','j','k','m','z'};`
- `float[] table3 = {15.7f,75,-22.03f,3,57};`
- `String[] Str =  
{"chat","chien","souris","rat","vache"};`



# Manipulation de tableaux statique

- ❑ L'accès aux éléments d'un tableau se fait avec des indices (à partir de 0)
- ❑ `int length`: (propriété) taille d'un tableau
- ❑ `Char[] toCharArray()`: transforme une chaîne en un tableau

# Opérateurs

## ❑ Opérateur d'affectation:

- `x=3;`
- `x=y=z=w+5;`
- pour la comparaison ( `if (x == 5)` )
- D'autres opérateurs combinés : `+=`, `-=`, `*=`, `/=`, `%=`, `^=`, `<<=`, `>>=`

## ❑ Les opérateurs arithmétiques à deux opérandes:

- `+` addition
- `-` soustraction
- `*` multiplication
- `/` division
- `%` modulo (reste de la division)

# Opérateurs

## ❑ Les opérateurs arithmétiques à deux opérandes

• exemples de raccourcis:

```
x = x + 4; ou x+=4;  
z = z * y; ou z*=y;  
v = v % w; ou v%=w;
```

## ❑ Les opérateurs relationnels:

- == égal
- < plus petit que
- > plus grand que
- <= plus petit ou égal
- >= plus grand ou égal
- != différent

# Opérateurs

## ❑ Les opérateurs logiques

- && Et (deux opérandes)
- || Ou (deux opérandes )
- ! Non (un seul opérande)

## ❑ L'opérateur à trois opérandes ?:

- condition ? expression\_si\_vrai : expression\_si\_faux
- exemple : `x = y < 5 ? 4 * y : 2 * y;`

**Equivalent à :**

`if (y < 5) x = 4 * y;`

`else x = 2 * y;`

# Opérateurs priorité

Opérateurs unaires	++exp,--exp,+exp,-exp, !
Multiplicatifs	*, /, %
Additifs	+, -
Relationnels	<, <=, >, >=
Égalité	==, !=
Et logique	&&
Ou logique	
affectation	=, +=, -=, *=, /=, %=

## ❑ Remarques:

- Les opérateurs les plus prioritaires sont évalués en premier
- Deux opérateurs de même priorité sont évalués de gauche à droite, à l'exception des opérateurs d'affectation qui sont évalués de droite à gauche

# Structures de contrôle

## ❑ L'instruction conditionnelle if

```
if (expression) instruction;
```

ou :

```
if (expression) {  
    instruction1;  
    instruction2;  
}
```

## ❑ L'instruction conditionnelle else

```
if (expression) {  
    instructions1;  
} else {  
    instructions2;  
}
```

# Structures de contrôle

## ❑ Les instructions conditionnelles imbriquées

```
if (expression1) {  
    bloc1;  
}  
else if (expression2) {  
    bloc2;  
}  
else if (expression3) {  
    bloc3;  
}  
else {  
    bloc5;  
}
```



# Structures de contrôle

## ❑ L'instruction switch

```
switch( variable) {  
    case valeur1: instructions1;break;  
    case valeur2: instructions2;break;  
    case valeur3: instructions2;break;  
    .....  
_case valeurN: instructionsN;break;  
    default: instructions;break;  
}
```

# Structures de contrôle

## ❑ La boucle for

La boucle **for** est une structure employée pour exécuter un bloc d'instructions un nombre de fois en principe connu à l'avance. Elle utilise la syntaxe suivante :

```
for (initialisation; test; incrémentation) {  
    instructions;  
}
```

Exemple :

```
int i = 0;  
for (i = 2; i < 10; i++) {  
    System.out.println("Vive Java !");  
}
```

- ❑ Boucle **for each** (introduite par le JDK 5.0) .
  - boucle Elle ne peut être utilisée qu'au parcours des éléments d'une collection ou d'un tableau.
  - Cette s'utilise en lecture seule, ainsi, on ne pas l'utiliser pour modifier un tableau ou une collection

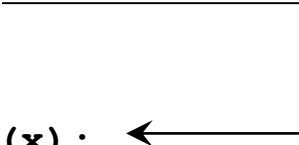
```
for (type element:  
collection) {  
    instructions;  
}
```

# Structures de contrôle

## □ Branchement au moyen des instructions `break` et `continue`


### ■ *break:*

```
int x = 10;  
for (int i = 0; i < 10; i++) {  
    x--;  
    if (x == 5) break;  
}  
System.out.println(x);
```

A diagram consisting of a horizontal line from the end of the 'if' statement to a vertical line, which then has an arrow pointing left to the 'System.out.println(x);' statement, indicating that the loop is exited.

### ■ *continue:*

```
for (int i = 0; i < 10; i++) {  
    if (i == 5) continue;  
    System.out.println(i);  
}
```

A diagram consisting of a horizontal line from the end of the 'if' statement to a vertical line, which then has an arrow pointing left to the 'for' loop header, indicating that the rest of the loop body is skipped and the next iteration begins.

# Structures de contrôle

## □ L'instruction While

```
while (condition){  
    BlocInstructions;  
}
```

Exemple :

```
int s=0,i=0;  
while (i<10){  
    s+=i;  
    i++;  
}
```

## □ L'instruction do .. While

```
do  
    BlocInstructions;  
while (condition)
```

Exemple :

```
int s=0,i=0;  
do{  
    s+=i;  
    i++;  
}  
while (i<10)
```

# Entrées-Sorties

## ❑ Affichage :

- `System.out.print(...)`
- `System.out.println(...)`

## ❑ Lecture :

### ▪ Lecture d'un entier :

```
import java.util.Scanner ;  
Scanner keyb = new Scanner(System.in) ;  
int i = keyb.nextInt() ;
```

### ▪ Lecture d'une chaîne :

```
Scanner sc = new Scanner(System.in) ;  
String str = sc.nextLine();
```

- De façon générale, pour récupérer un type de variable, il suffit d'appeler `next<Type de variable commençant par une majuscule>`
- Exemple: `nextDouble`, `nextByte`, ....
- Sauf le type `char`

# Méthodes statiques

- ❑ Le modificateur **static** permet de définir une donnée ou une méthode commune à toutes les instances de la classe. Ce sont des entités qui existent en l'absence de tout objet de la classe.
- ❑ Avant l'étude du concept d'objets on ne peut utiliser que des variables et méthodes statiques dans une classe.
- ❑ Toute méthode statique d'une classe peuvent être invoquée depuis n'importe quelle autre méthode statique de la classe
  - **l'ordre de déclaration des méthodes n'a pas d'importance**
- ❑ Pour invoquer méthode d'une autre classe il faut la préfixer par **NomClasse**.

```
public class A {  
    static void m1() {  
        ...  
    }  
  
    static void m2() {  
        m1();  
        m3();  
    }  
  
    static void m3() {  
        ...  
    }  
}
```

# Méthode et passage des arguments

- ❑ Quant aux paramètres de types base, le passage est par valeur.
- ❑ Le passage par adresse est pour les types objets de classe.
- ❑ Afin de réaliser un passage par adresse pour les types de base on peut les envelopper dans les objets de classe que l'on crée pour ce fin.

## ■ Exemple:

```
class Entier{  
    int val ;  
    public int getVal(){ return val ;}  
    public void setVal(int v){ val=v ;}  
}
```

# Quelques conventions

- ❑ Les noms de class commencent par une majuscule
  - `Personne`, `Objet`
- ❑ Les mots contenus dans un identificateur commencent par une minuscule:
  - `main`, `uneMethode`, `uneVariable`
- ❑ Les constantes sont en majuscules
  - `UNE_CONSTANTE`



# Liens Importants

## ❑ JDK

<https://www.oracle.com/java/technologies/javase-downloads.html>

## ❑ Eclipse IDE for Java Developers

<https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers>

# Exercices

- ❑ **Exercice1** : Ecrivez un programme qui Affiche les nombres impairs entre 0 et 100 .
- ❑ **Exercice 2** : Ecrivez un programme qui Affiche les 20 premiers termes de la suite :

$$u_0 = 15$$

$$u_{n+1} = u_n - 6$$