

Programmation Orientée Objet Java

M.WANE

Ingénieur Informaticien

Analyste, Concepteur

Développeur d'Applications

Programme

Cours 1 : Les Fondamentaux de Java

Cours 2 : Programmation orientée Objet Java

- ❖ Objet et Classe
- ❖ Héritage et accessibilité
- ❖ Polymorphisme
- ❖ Collections

Cours 3 : Java Fonctionnel

- ❖ Expressions Lambda et Java Fonctionnel
- ❖ Stream
- ❖ Notions de Design Patterns

Cours 4 : Java bases de données

- ❖ JDBC
- ❖ JPA

Cours 5 : Interface Graphique

- ❖ SWING
- ❖ JAVA FX

Cours 1 : Les Fondamentaux de Java

I) Qu'est-ce que java ?

- ❖ Langage de **programmation orienté objet** (Classe, Objet, Héritage, Encapsulation et Polymorphisme).
- ❖ **Open source** : On peut accéder le code source de java. Ce qui permet aux développeurs, en cas de besoin, de développer ou modifier des fonctionnalités de java.
- ❖ Avec java on peut créer des applications **multiplateformes**, c'est-à-dire le langage Java est utilisé pour créer :
 - ✚ **Des applications Desktop JSE** (Java Standard Edition). On parle de **client lourd**.
 - ✚ **Des applications Web JEE** (Java Entreprise Edition) pour développer les applications web qui vont s'exécuter dans un serveur d'application JEE (Tom Cat, JBoss). On parle de **clients légers**.
 - ✚ **Des applications mobiles ou JME** (Java Micro Edition), pour développer les applications pour les téléphones portables.

✚ Des applications embarquées dans des cartes à puces ou JCA

(Java Card Edition), pour développer les applications qui vont s'exécuter dans des cartes à puces.

- ❖ Les applications java sont **portables**, c'est-à-dire, on peut créer une application java dans une plateforme ([Windows](#) ou [Linux](#) ou [Mac OS](#)) donnée et on peut l'exécuter sur n'importe quelle autre plateforme.

Le principe de java est : **Write Once Run Every Where** .

Cette portabilité du Langage est du fait que java est un Langage **Compilé** et **interprété**.

❖ Exécution d'un Programme en Java

- ✚ Le **JDK** (Java Développement Kit) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en **bytecode** destiné à la **machine virtuelle Java**(JVM).

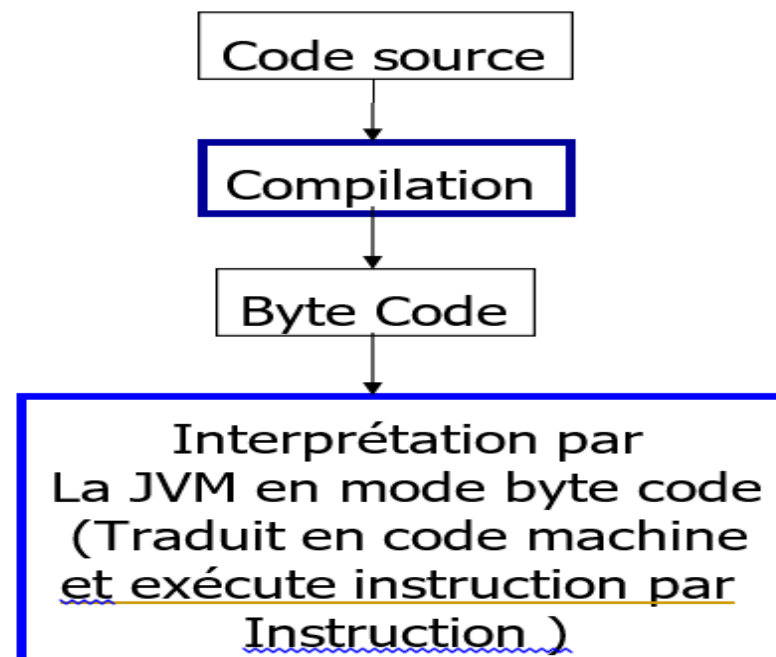
Le JDK contient **JRE** (Java Runtime Environnement).

- ✚ Le JRE permet l'exécution des programmes écrits en langage Java sur de nombreux types d'appareils (ordinateurs personnels, téléphones mobiles, etc.) en faisant abstraction des caractéristiques techniques de la plateforme informatique sous-jacente.

- Le JRE se compose d'une **machine virtuelle** (JVM), de **bibliothèques logicielles** utilisées par les programmes Java et d'un **plugin** pour permettre l'exécution de ces programmes depuis les navigateurs web.

Exemple d'Exécution d'un Programme en JAVA

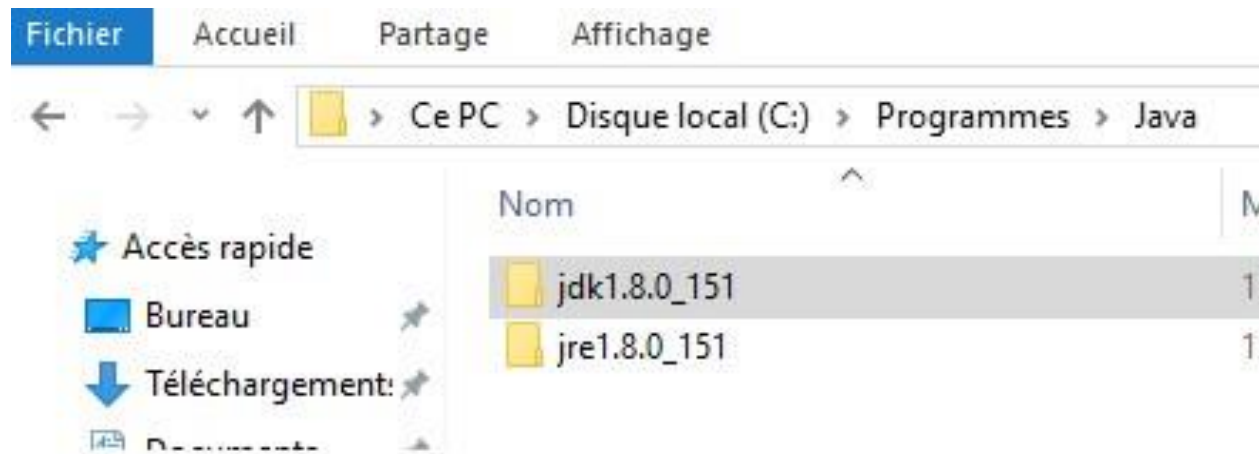
Byte Code



II) Installation de java

Pour créer une application java, il faut installer **Java Développement Kit**

- ☐ Le Kit de développement java JDK peut être téléchargé gratuitement à partir du site de Sun Microsystem son éditeur principal (www.oracle.com).
- ☐ Exécuter [jdk-8u151-nb-8 2-windows-x64.exe](#).
- ☐ Le **JDK** sera installé dans le répertoire **c:\programme\java**, le **JRE** sera aussi installé

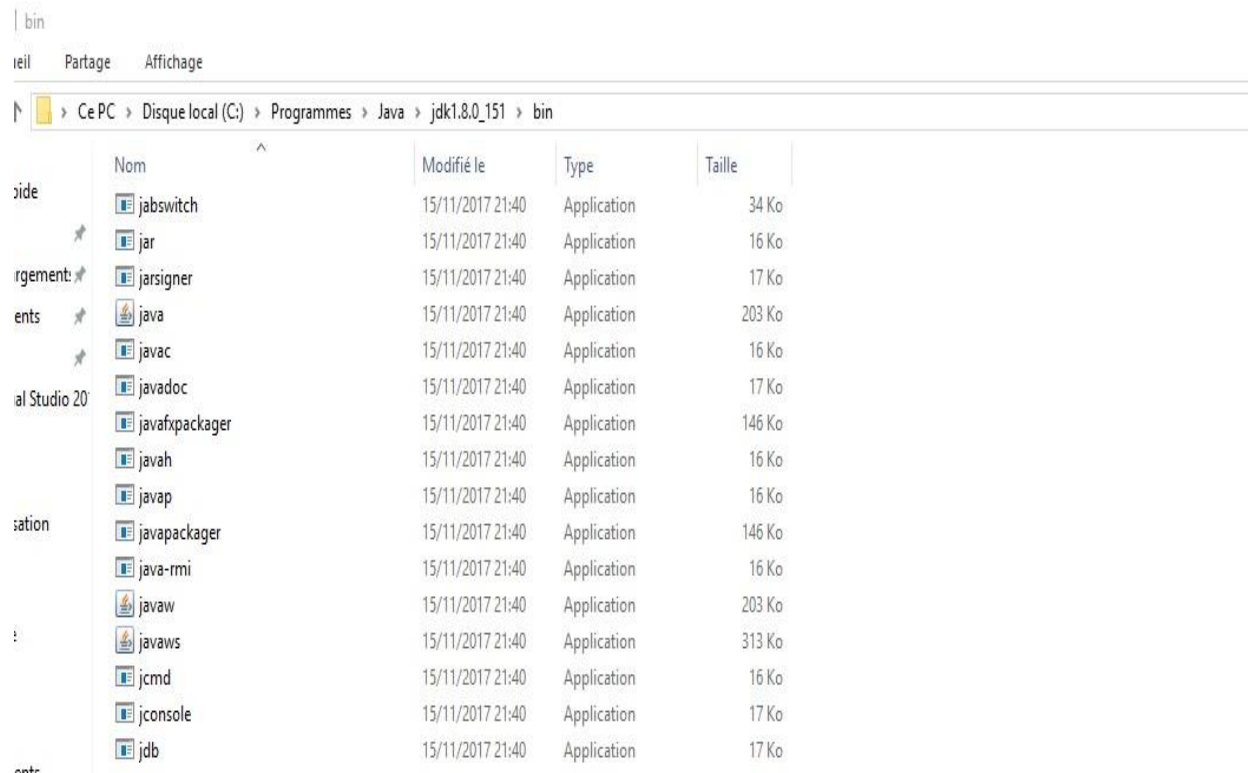


Dossier d'installation du JDK

☐ Ce que contient le JDK

+ Dossier bin

Les programmes nécessaires au développement java sont placés dans le répertoire
C:\Program Files\Java\jdk1.8.0_151\bin



The screenshot shows a Windows Explorer window with the address bar displaying the path: C:\Program Files\Java\jdk1.8.0_151\bin. The left sidebar shows the navigation pane with 'Ce PC' selected. The main pane displays a list of files and folders. The table below represents the data shown in the main pane.

	Nom	Modifié le	Type	Taille
side	jabswitch	15/11/2017 21:40	Application	34 Ko
	jar	15/11/2017 21:40	Application	16 Ko
rgement:	jarsigner	15/11/2017 21:40	Application	17 Ko
ents	java	15/11/2017 21:40	Application	203 Ko
	javac	15/11/2017 21:40	Application	16 Ko
al Studio 20	javadoc	15/11/2017 21:40	Application	17 Ko
	javafxpackager	15/11/2017 21:40	Application	146 Ko
	javah	15/11/2017 21:40	Application	16 Ko
	javap	15/11/2017 21:40	Application	16 Ko
sation	javapackager	15/11/2017 21:40	Application	146 Ko
	java-rmi	15/11/2017 21:40	Application	16 Ko
	javaw	15/11/2017 21:40	Application	203 Ko
	javaws	15/11/2017 21:40	Application	313 Ko
	jcmd	15/11/2017 21:40	Application	16 Ko
	jconsole	15/11/2017 21:40	Application	17 Ko
	jdb	15/11/2017 21:40	Application	17 Ko

- **javac.exe** : Compilateur java.
- **java.exe** : Interpréteur du bytecode java.
- **appletviewer.exe** : Pour tester les applets java.
- **Jdb.exe** : Débogueur java.
- **Javap.exe** : désassembleur du bytecode.
- **Javadoc.exe** : Générer la documentation de vos programmes java.
- **Javah.exe** : Permet de lier des programmes Java avec des méthodes natives, écrites dans un autre langage et dépendant du système.
- **jar.exe** : Permet de compresser les classes Java ainsi que tous les fichiers nécessaires à l'exécution d'un programme (graphiques, sons, etc.). Il permet en particulier d'optimiser le chargement des applets sur Internet.
- **jarsigner.exe** : Un utilitaire permettant de signer les fichiers archives produits par jar.exe.

III) Configuration de l'environnement

❑ **La configuration de l'environnement comporte deux aspects :**

- Définir la variable d'environnement **path** qui indique le chemin d'accès aux programmes exécutables :

- Cette variable **path** devrait contenir le chemin du JDK utilisé :

path= C:\Program Files\Java\jdk1.8.0_151\bin ;

- Quand elle exécute une application java, la **JVM** consulte la variable d'environnement **classpath** qui contient le chemin d'accès aux classes java utilisées par cette application.

classpath= .; c:\monProjet\lib;

c:\programmation

❑ Outils de développement java

- + **Un Editeur de texte ASCII** : on peut utiliser un simple éditeur comme Notepad de Windows mais il est préférable d'utiliser un éditeur conçu pour la programmation java exemples : **Ultra édit, Creator,**
- + **IDE** : est l'environnement de développement intégré en java, on existe de nombreux IDE du Langage Java :
 - **Eclipse** : est l'IDE le plus préféré pour les développeurs java. Il est gratuit et c'est un environnement ouvert.
 - **JDeveloper d'Oracle.**
 - **NetBeans IDE**
 - **etc**

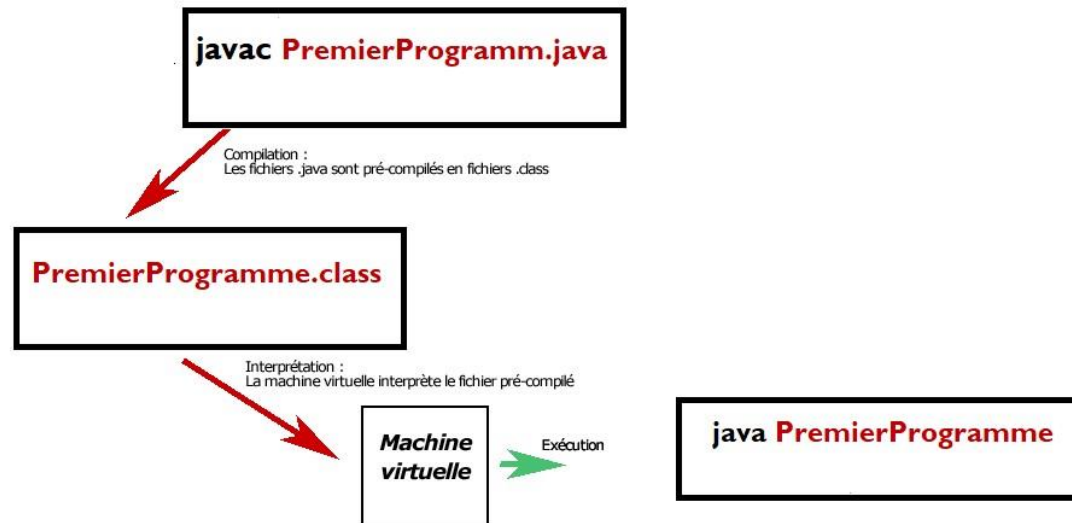
☐ Premier programme java

- ☐ 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\jdk1.8.0_151\bin .

- ☐ Pour rendre accessible ce programme depuis n'importe quel répertoire, il faut ajouter la commande : **path C:\Program Files\Java\jdk1.8.0_151\bin** dans le fichier **autoexec.bat**.
- ☐ Après compilation du programme **PremierProgramme.java**, il y a génération du fichier **PremierProgramme.class** qui représente le **ByteCode** de ce programme.
- ☐ Pour exécuter ce programme en byte code, il faut faire appel au programme **java.exe** qui représente l'interpréteur du bytecode

Principe d'exécution d'un code JAVA



Exécution du Premier programme java

```
public class PremierProgramme
{
    public static void main(String[] args)
    {
        System.out.println("First Test");
    }
}
```

PremierProgramme.java

```
: \Mes Donnees\JAVA\COURS\L2\Exo\2017-2018>javac PremierProgramme.java
: \Mes Donnees\JAVA\COURS\L2\Exo\2017-2018>java PremierProgramme
First Test
: \Mes Donnees\JAVA\COURS\L2\Exo\2017-2018>
```

Compilation et Execution

IV) Structures fondamentales du langage java

Au niveau syntaxe, Java est un langage de programmation qui ressemble beaucoup au langage c++.

Toutefois quelques simplifications ont été apportées à java pour des raisons de sécurité et d'optimisation.

Dans cette partie nous ferons une présentation succincte des types primitifs, les enveloppeurs, déclarations des variables, le casting des primitives, les opérateurs arithmétiques et logiques, les structures de contrôle (if, switch, for et while), les fonctions et les Tableaux.

A) Les types primitifs

Java dispose des primitives suivantes :

<u>Primitive</u>	<u>Étendue</u>	<u>Taille</u>
char	0 à 65 535	16 bits
byte	-128 à +127	8 bits
short	-32 768 à +32 767	16 bits
int	-2 147 483 648 à + 2 147 483 647	32 bits
long		64 bits
float	de $\pm 1.4\text{E-}45$ à $\pm 3.40282347\text{E}38$	32 bits
double		64 bits
boolean	true ou false	1 bit
void	-	0 bit

A.1) Utilisation des primitives

Les primitives sont utilisées de façon très simple. Elles doivent être déclarées, tout comme les handles d'objets, avec une syntaxe similaire:

a) Déclaration d'une Variable : Type nomVar ;

int i; char c;

boolean fini;

Les primitives peuvent être initialisées en même temps que la déclaration.

int i = 12;

char c ='a';

boolean fini = true;

b) Déclaration d'une

Constante :

final type

NOMCONSTANTE=Valeur ;

A.2) Typologie en Java

- ✚ Pour respecter la typologie de java, les noms des variables commencent toujours par un caractère en minuscule et pour indiquer un séparateur de mots, on utilise les majuscules.

Exemples :

int nbPersonnes;

String nomPersonne

- ✚ Les méthodes reprennent la règle du Camel Case c'est-à-dire la première lettre est en majuscule et chaque nouveau mot commence par une lettre majuscule.

Exemples :

void maPremiereFonction() ;

String nextLine() ;

- ✚ Les noms des constantes sont majuscules.

Exemples : PI,

TAILLE

- ✚ Les noms des classes commencent toujours par un caractère en minuscule et chaque nouveau mot commence par une lettre majuscule.

Exemples :

String

Scanner

MaPremiereClasse

A.3) Valeurs par défaut des primitives

Toutes les primitives de type numérique utilisées comme membres d'un objet sont initialisées à la valeur **0**.

Le type booléen est initialisé à la valeur **false**.

.

B) Les enveloppeurs (wearpers)

Les primitives sont enveloppées dans des objets appelés enveloppeurs (Wearpers).

Les enveloppeurs sont des classes

Classe

Character

Byte

Short

Integer

Long

Float

Primitive

char

byte

short

int

long

float

Double
Boolean
Void
BigInteger -
BigDecimal
-

double
boolean
-
-

B.1) Utilisation des primitives et enveloppeurs

Exemple :

double v1=5.5; // v1 est une primitive

Double v2=**new** Double(5.6); // v2 est un objet

long a=5; // a est une primitive Long b=**new** Long(5); // b est un objet

Long c= 5L; // c est un objet

System.**out**.println ("a="+a);

System.**out**.println("b="+b.longValue());

System.**out**.println("c="+c.byteValue());

System.**out**.println("V1="+v1);

System.**out**.println("V2="+v2.intValue()); Résultat:

a=5 b=5
c=5
V1=5.5
V2=5

C) Opérateur

C.1) Opérateur d'affectation :

- **x=3;** // x reçoit 3
- **x=y=z=w+5;** // z reçoit w+5, y reçoit z et x reçoit y

Les opérateurs arithmétiques à deux opérandes:

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

C.2) Les opérateurs arithmétiques à deux opérandes

(Les raccourcis)

$x = x + 4$; ou $x+=4$; $z = z * y$; ou $Z*=y$; $v = vss \% w$; ou $v\%=w$;

c.3) Les opérateurs relationnels :

- $==$: équivalent
- $<$: plus petit que
- $>$: plus grand que
- $<=$: plus petit ou égal
- $>=$: plus grand ou égal
- $!=$: non équivalent

c.4) Les opérateurs d'incrémentations et de décrémentation :

$++$: Pour incrémenter ($i++$ ou $++i$)

$--$: Pour décrémentation ($i--$ ou $--i$)

c.5) Les opérateurs logiques

&& Et (deux opérandes)

|| Ou (deux opérandes)

! Non (un seul opérande)

c.6) 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

$x = 2 * y;$

Structures de contrôle

L'instruction conditionnelle if

La syntaxe de l'instruction **if** peut être décrite de la façon suivante:

```
if (expression) instruction;  
ou :  
if (expression) {  
    instruction1; instruction2;  
}
```

L'instruction conditionnelle else

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

Structures de contrôle

Les instructions conditionnelles imbriquées

Java permet d'écrire ce type de structure sous la forme :

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

L'instruction switch

Syntaxe :

```
switch ( variable) { case  
valeur1: instr1; break;  
case valeur2: instr2; break;  
case valeurN: instrN;break;  
default: instr;break;  
}
```

Exemple:

```
import java.util.Scanner;
public class Test {
    public static void main(String[] args) {
        System.out.print("Donner un nombre:");
        Scanner clavier=new Scanner(System.in);
        int nb=clavier.nextInt();
        switch(nb) {
            case 1 : System.out.println("Lundi");break;
            case 2 : System.out.println("Mardi");break;
            case 3: System.out.println("Mercredi");break;
            default: System.out.println("Autrement");
            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 :

```
for (int i = 2; i < 10;i++) {  
    System.out.println("I="+i);  
}
```

```
}
```

```
}
```

L 'instruction While

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

Exemple :

```
int s=0;int i=0;  
while (i<10){  
    s+=i; i++;  
}  
System.out.println("Somme="+s);
```

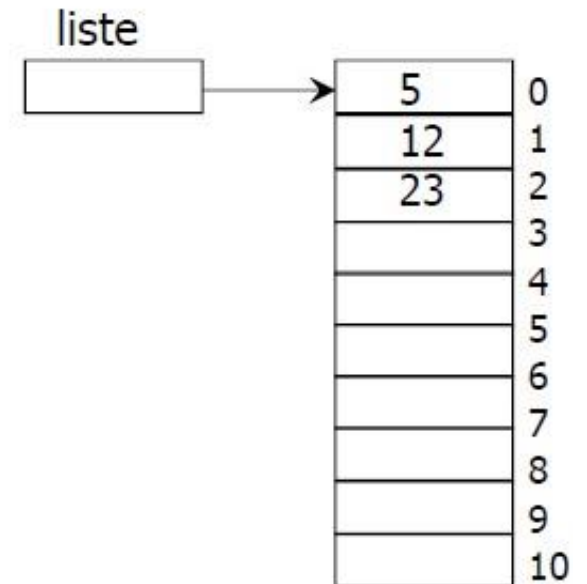
L 'instruction do .. while

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

Exemple :

```
int s=0;int i=0;  
do{ s+=i; i++;  
}while (i<10);  
System.out.println("Somme="+s);
```

Tableaux de primitives



- Tableaux de primitives:
- Déclaration :
 - Exemple : Tableau de nombres entiers
 - `int[] liste;`
 - liste est un handle destiné à pointer vers un tableau d'entier
- Création du tableau
 - `liste = new int[11];`
- Manipulation des éléments du tableau:
 - `liste[0]=5; liste[1]=12; liste[3]=23;`
 - `for(int i=0;i<liste.length;i++){`
 - `System.out.println(liste[i]);`
 - `}`