



Ecole Supérieure  
d'Informatique et du Numérique  
COLLEGE OF ENGINEERING & ARCHITECTURE

# Chap2: JAVA

---

Syntaxe de base JAVA



# Plan

- I. Introduction et objectifs
- II. Structures Lexicales & convention de codage
- III. Types de base et la conversion (cast)
- IV. Constantes
- V. Structures de contrôle (conditionnelle et itérative)
- VI. Opérateurs
- VII. Les tableaux

# Introduction & Objectifs

- L'objectif de ce cours est de vous présenter la syntaxe et les mots réservés en JAVA.
- Type et déclaration des attributs:
  - Les types primitifs/les enveloppeurs
  - Les structures conditionnelles
  - Les structures itératives
  - Les tableaux

Pour simplifier l'apprentissage de ces éléments, nous allons utiliser une seule classe principale avec une méthode main

# Structures Lexicales & convention de codage

## ❖ Identificateurs

- commence par une lettre et après ... chiffres, "\_", "\$"
- Note : les lettres peuvent être **de n'importe quelle langue caractère Unicode (universel)**

**Exemple : `int μ = 10;`**

## ❖ **Case sensitive**: JAVA est langage sensible à la casse (MAJ/min)

## ❖ Séparateur ";"

## ❖ Commentaires

- `//` pour une ligne de commentaire
- `/*` commentaires `*/` pour plusieurs lignes

## ❖ Documentation en ligne avec javadoc

- `/**` Documentation de classe et méthodes `*/`

# Structures Lexicales & convention de codage



myGreatVariableName  
Camel Case

## ➤ Nommage des variables et des méthodes

- Si l'identificateur contient un seul mot on commence par une lettre minuscule

**Exemples:** int **a**ge; String **n**om; **a**jouter();

- Si l'identificateur contient plus qu'un mot, le premier sera en minuscule, les suivants commenceront par une majuscule

**Exemples:** String **f**iliere**E**tudiant, **g**et**S**tring(); **a**jouter**U**n**N**ouveau**E**tudiant();

## ➤ Nommage des constantes

- une constante doit être en lettre capitales

**Exemple:** **final** int **MAXSIZE** = 100;

## ➤ Nommage des classes

- Le nom des classes commence toujours par une lettre majuscule

**Exemple:** class **E**tudiant    **Exemple:** class **E**sin**I**ng3

# Types des attributs

- Il existe plusieurs types primitifs en JAVA:

Type	Signification	Taille (Octets)
char	Caractère Unicode	2
byte	Entier très court	1
short	Entier court	2
int	Entier	4
long	Entier long	8
float	Nombre réel simple	4
double	Nombre réel double	8
boolean	Valeur logique (booléen)	1

# Les enveloppeurs (Wrapper)

- Chacun des types primitifs peut être "enveloppé" dans un objet provenant d'une classe prévue à cet effet et appelée *Wrapper* (mot anglais signifiant *enveloppeur*).
- Les enveloppeurs sont donc des classes représentant un type primitif.

Enveloppeur	Type primitif
Character	char
Byte	byte
Short	short
Integer	int

Enveloppeur	Type primitif
Long	long
Float	float
Double	double
Boolean	boolean

# Affichage des messages

- Pour afficher un message sur la console il faut utiliser l'instruction

1. **System.out.println**(" Message à afficher avec nouvelle ligne");
2. **System.out.print**(" Message à afficher sur la même ligne");

```
// message affiché sur une seule ligne
System.out.println("Message afficher dans une nouvelle ligne ");
// message affiché sur la meme ligne
System.out.print("Message afficher dans la meme ligne ");
System.out.println(" ***** Message 2 ");

// integration des variables dans un message
int nbrEtd = 12;
String filiere = "Info 2";
System.out.println("La filiere  " + filiere + " contient " + nbrEtd + " etudiants");
```



# Affichage des messages

- Les constants caractères

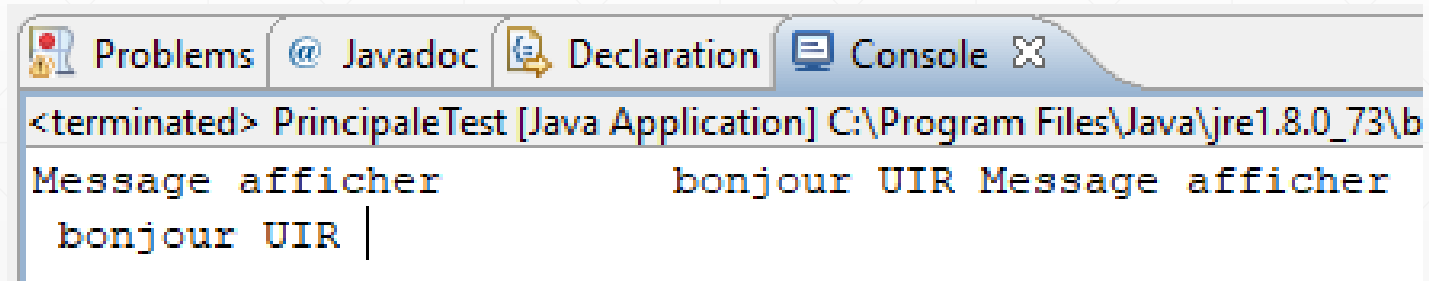
Séquence	Signification
\n	Retour a la ligne
\t	Tabulation
\b	Retour arrière
\r	Retour chariot
\f	Saut de page
\\	Slash inversé
\'	Guillemet simple
\ "	Guillemet double

# Affichage des messages

- Exemples d'application

```
System.out.print("Message afficher \t bonjour UIR ");  
System.out.print("Message afficher \n bonjour UIR ");
```

## Résultat



The screenshot shows the 'Console' tab of a Java IDE. The title bar includes 'Problems', 'Javadoc', 'Declaration', and 'Console'. The console output displays the result of the Java application: the first line shows 'Message afficher' followed by a tab character and 'bonjour UIR', and the second line shows 'Message afficher' followed by a newline character and 'bonjour UIR'.

```
<terminated> PrincipaleTest [Java Application] C:\Program Files\Java\jre1.8.0_73\bin  
Message afficher          bonjour UIR Message afficher  
  bonjour UIR |
```

# Conversion de type (Casting)

- La conversion de type le transtypage
- Il existe deux types de conversion en JAVA:
- **Conversion implicite:** Il s'agit d'une modification du type de données réalisé automatiquement par un compilateur. Exemple de conversion d'un type primitif vers son Wrapper

```
int nombre = 10;  
Integer nombreWrap = nombre;  
System.out.println("nombre = " + nombre + " nombreWrap = " + nombreWrap);
```

- **Conversion explicite:** consiste en une modification de type de donnée forcé. Il faut utiliser l'opérateur de Cast pour spécifier la conversion

```
double nbr1 = 3.14;  
int nbr1Cast = (int)nbr1;
```

```
String nbr2 = "15";  
int resultat;  
resultat = Integer.parseInt(nbr2);
```

# Les constantes

- Les constantes en JAVA se déclare en utilisant le mot final pour que personne ne puisse changée leurs valeurs durant l'exécution du programme
- **Exemple:**

- **final** double PI= 3.14

```
final double PI = 3.14;  
System.out.println("==>> 4 PI = "+ PI);
```

- **Résultat:**

```
4==>> PI = 3.14
```

# Les opérateurs

- Opérateurs arithmétiques

▪ +	Addition	6 + 4
▪ -	Soustraction	9 - 8
▪ *	Multiplication	5 * 5
▪ /	Division	14 / 7
▪ %	Modulo	20 % 5

# Les opérateurs

- **Opérateurs arithmétiques avec affectation**

- **+=**     $x += y$                      $x = x + y$
- **-=**     $x -= y$                      $x = x - y$
- **\*=**     $x *= y$                      $x = x * y$
- **/=**     $x /= y$                      $x = x / y$

- **Opérateurs unaires**

- **"-", "+"**
- $a * -b$     Ok                     $a * (-b)$  plus lisible

# Les opérateurs

- Incrémentation et décrémentation des variables

- **Incrémentation**

- pré incrémentation ++i
- post incrémentation i++

## Décrémentation

- prés décrementation --i
- post décrementation i--

```
int a = 2;
int b = 3;

int c = a++ + b;
System.out.println(" c= " + c + " a " + a + " b " + b);

a= 2; b = 3;
c = ++a + b;
System.out.println(" c= " + c + " a " + a + " b " + b);
```

# Les opérateurs

- **Opérateurs relationnels**

- `>` Supérieur à
- `<` Inférieur à
- `>=` Supérieur ou égale à
- `=<` Inférieur ou égale à
- `!=` différent
- `==` Egal Egal
- `var1String.equals(var2String)` pour la comparaison des chaînes de caractères

## Opérateurs logiques

- `&&` et
- `||` ou
- `!` Non



# Les structures conditionnelles (SI)

- Les conditions en JAVA peuvent être réalisé en utilisant le mot clé **IF**

```
if (Condition){  
    // traitement  
}  
else{  
    // traitement  
}
```

```
if (Condition){  
    // traitement1  
}  
else if (Condition 2){  
    // traitement2  
}  
else{  
    // traitement3  
}
```

# Les structures conditionnelles (Switch case)

- JAVA offre l'utilisation des Switch case pour structurer plusieurs conditions de déclenchement d'un traitement, comme suite :

```
Switch (key){  
    Case value:  
    Break;  
  
    Case value2:  
    Break;  
  
    Default:  
    Break;  
}
```

```
int mois = 2;  
switch (mois) {  
    case 1 :  
        System.out.println("le mois de Janvier");  
        break;  
    case 2:  
        System.out.println("le mois de Fevrier");  
        break;  
    case 3:  
        System.out.println("le mois de Mars");  
        break;  
    default:  
        System.out.println("le mois par défaut");  
        break;  
}
```

# Structure itérative (boucle For)

- L'utilisation des boucles For en JAVA se fait de la manière suivante:

```
For(compteur; condition d'arrêt; incrément){  
    //traitement  
}
```

```
//incrément par 1  
for(int i=0; i < 10; i++){  
    System.out.println("i = " + i);  
}  
  
// incrément par 2  
int k;  
for(k=0; k < 10; k= k+2){  
    System.out.println("k = " + k);  
}
```

# Structure itérative (Boucle while)

- L'utilisation de la boucle while en JAVA se fait de la manière suivante

```
while (condition){  
    //traitement  
}
```

```
boolean condition = true;  
  
while(condition == true){  
    System.out.println("je suis une boucle infinie");  
}
```

# Structure itérative (Do – While )

- L'utilisation de la boucle DO - while en JAVA se fait de la manière suivante

```
do{  
    //traitement  
} while (condition);
```

```
boolean condition2 = true;  
do{  
    System.out.println("boucle do while");  
}while(condition2 == true);
```

# Les tableaux (1 dimension)

Constructeur



- La création des tableaux à une dimension

1. **type** nomTableau[] = **new** **type**[**dimension**];

2. **type** nomTableau[] = {val1, val2, val3, val4};

```
int tblEntier[] = new int[6];
int tblEntier2[] = {2, 4, 6, 8, 10, 12};

// affichage tableau 1
for(int i = 0; i < tblEntier.length ; i++){
    System.out.println(tblEntier[i]);
}

// affichage tableau 2
for(int i = 0; i < tblEntier2.length ; i++){
    System.out.println(tblEntier2[i]);
}
```

# Les tableaux (2 dimensions)

- La déclaration d'un tableau à deux dimensions et son parcours en utilisant la boucle for

1. `type nomTableau[][] = new type[NbrLigne][NbrCol];`
2. `type nomTableau[][] = {{val1, val2, val3}, {val4, val5, val6}};`

```
int tbl2D[][] = new int[3][3];
int tbl2D2[][]= {{2, 4, 6} ,{8, 10, 12} };
// affichage tableau 2
// parcourir les lignes
for(int i = 0; i< tbl2D2.length ; i++){
    // parcourir les colonnes
    for(int j = 0; j < tbl2D2[i].length ; j++){
        System.out.print(tbl2D2[i][j] + " | ");
    }
    // pour avoir une nouvelle ligne
    System.out.println();
}
```

# Interaction avec l'utilisateur (Scanner)

- Lire les informations à partir du clavier en utilisant un Scanner
- Il faut importer la librairie **import java.util.Scanner;**

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

```
int i = sc.nextInt();  
double d = sc.nextDouble();  
long l = sc.nextLong();
```

- Lecture d'une chaîne de caractère

```
System.out.println("veuillez saisir votre nom");  
sc.nextLine(); //attention on utilise nextLine deux fois  
String mot = sc.nextLine();  
System.out.println("votre nom est " + mot);
```



# Interaction avec l'utilisateur (Scanner)

- Pour simplifier l'utilisation des Scanner, il est plus simple d'utiliser deux scanner :
  - 1 pour la lecture des variables int, float double ....
  - 2 pour la lecture des chaînes de caractère

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

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

---

# Interaction avec l'utilisateur

- Lire les informations à partir du clavier en utilisant la classe (**Clavier.java**)
- Il faut ajouter la classe clavier dans le projet
- La classe Clavier.java permet d'utiliser les fonctions suivantes
  - lireString()
  - lireLigne()
  - lireInt()
  - lireDouble()
  - lireFloat()

```
System.out.println("veuillez saisir votre nom");  
s = Clavier.lireString();  
System.out.println("la chaine de caractere " + s);  
  
System.out.println("veuillez saisir votre nom");  
s = Clavier.lireLigne();  
System.out.println("la ligne " + s);  
  
int a = Clavier.lireInt();  
System.out.println("a= " + a);
```