2 Introduction à Java

Définition

La technologie **Java** définit à la fois un langage de programmation **orienté objet** et une plateforme informatique. Créée par l'entreprise Sun Microsystems (Sun) en **1995**, et reprise depuis par la société Oracle en **2009.**

Java s'appuie sur la syntaxe du langage **C** et du **C++**

Syntaxe de base de Java

```
1  package javaapp;
2  import java.util.Scanner;
3  //JavaApp.java: Somme de deux nombres entiers
4  public class JavaApp {
5  public static void main(String[] args) {
6      // TODO code application logic here
7      System.out.println("Premier programme Java !");
8      Scanner in = new Scanner(System.in);
9      int a,b,som;
10     System.out.print("a = ");
11     a = in.nextInt();
12     System.out.print("b = ");
13     b = in.nextInt();
14     som=a+b;
15     System.out.println("a+b = "+som);
16     }
17 }
```

Lecture/écriture

```
1  package javaapp;
2  import java.util.Scanner;
3  //JavaApp.java: Somme de deux nombres entiers
4  public class JavaApp {
5  public static void main(String[] args) {
6      // TODO code application logic here
7      System.out.println("Premier programme Java !");
8      Scanner in = new Scanner(System.in);
9      int a,b,som;
10     System.out.print("a = ");
11     a = in.nextInt();
12     System.out.print("b = ");
13     b = in.nextInt();
14     som=a+b;
15     System.out.println("a+b = "+som);
16     }
17 }
```

Lecture/écriture

```
1  Scanner input = new Scanner(System.in);//import java.util.Scanner
   ;
2   int n=input.nextInt();
3   double d=input.nextDouble();
4   float f=input.nextFloat();
5   String s=input.nextLine();
6   char c=input.next().charAt(0);
7   //and so on ...
```

Types de données primitifs (de base)

Type	Description	Valeurs
byte	un entier signé sur 8 bits	de –128 à +127
short	un entier signé sur 16 bits	de –32 768 à +32 767
int	un entier signé sur 32 bits	de –2147483648 à 2147483647
long	un entier signé sur 64 bits	de l'ordre de (±) 9 milliards de milliards
float	un réel sur 32 bits	de l'ordre de 1.4E-45 à 3.4E38
double	un réel sur 64 bits	de l'ordre de 4.9E-324 à 1.79E308
boolean	vrai ou faux	true ou false
char	un caractère Unicode entier positif sur 16 bits	entre 0 et 65535

Types de données primitifs (de base)

Type	Description	Valeurs
byte	un entier signé sur 8 bits	de −128 à +127

R pour chaque type de données primitifs, des constantes indiquant les maximums et les minimums sont définies, par exemple : `Integer.MIN_VALUE, Integer.MAX_VALUE, Byte.MIN_VALUE, Byte.MAX_VALUE, Short.MIN_VALUE, //etc.`

R En Java, une variable déclarée et non initialisée ne peut pas être utilisée et une erreur de compilation va survenir.

char	un caractère Unicode entier positif sur 16 bits	entre 0 et 65535

Opérateur arithmétiques et logiques

Opérateur	Signification	Exemple
+	addition	n=a+b ;
-	soustraction	n=a-b ;
*	multiplication	n=a*b ;
/	division	n=a/b
%	modulo	n=a%b ;
++	incrémente la variable	i++ ;
- -	décrémente la variable	i- - ;
+=	-	n+=b ; $\Rightarrow n = n + b;$
-=	-	n-=b ; $\Rightarrow n = n - b;$
=	affectation	$a = b;$

Opérateurs arithmétiques et logiques

Opérateur	Signification	Exemple
>	supèrieur	$a > b$;
<	soustraction	$a < b$
>=	supèrieur ou égale	$a >= b$
< +	inférieur ou égale	$<= b$
==	égale	$a == b$
! =	différent à	$a! = b$
&&	et logique	$a\&\&b$
||	ou logique	a || b
!	négation	$!trouve$

L'opérateur de cast (conversion de types)

```
1 //converting a smaller type to a larger type size
2 byte -> short -> char -> int -> long -> float -> double
3 //example:
```

```
4 int n = 1, p = 5;
5 float x = 1.5f;
6 float b= n * x + p; // Automatic casting: float to double
7 long a=n;           // Automatic casting: int to long
8 double d =n+x;      // Automatic casting: int to double
9 char c='a';
10 int e=c;           // Automatic casting: char to int
```

L'opérateur de cast (conversion de types)

```
//converting a smaller type to a larger type size
byte -> short -> char -> int -> long -> float -> double
//example:
```

```
// converting a larger type to a smaller size type
double -> float -> long -> int -> char -> short -> byte

//Example:
double d = 9.78d;
int myInt = (int) d; // Manual casting: double to int
float f=3;
f=(float)(d+3);       // Manual casting: double to float
n= (int) f ;          // Manual casting: float to int
char c='a';
c=(char) e;           // Manual casting: int to char
```

Structures de contrôles

Les instructions de contrôle du déroulement lors de l'exécution des instructions ont la même syntaxe que pour le langage C.

Structures de contrôles

```
1  //if without else
2    if(test) //test is a logic expression
3    {  block_of_instructions }
4
5  //if with one instruction
6    if(test) one_intruction;
7
8  //if with else
9    if(test)
10   {  block_of_instructions }
11   else
12   {  block_of_instructions }
13
14 //if with else if
15   if(test)
16   {  block_of_instructions
17   }
18   else if
19   {  block_of_instructions }
```

Structures de contrôles

```
1  //for with one instruction
2    for(int i=0;i<n;i++) one_intruction;
3
4  //for with several instructions
5    for(int i=0;i<n;i++)
6    {
7             block_of_instructions ;
8    }
9
10 //while with one instruction
11   while(condition) one_intruction;
12
13 //while with one instruction
14   while(condition)
15   {
16            block_of_instructions ;
17   }
18
19 //do ... while
20   do
21   {
22            block_of_instructions ;
23   } while(condition) ;
```

Tableaux

Plusieurs éléments de même type peuvent être regroupés en tableau. On peut accéder à chaque élément à l'aide d'un d'indice précisant le rang de l'élément dans le tableau. Le premier élément porte l'indice 0. Ils sont initialisés par défaut à 0 pour les nombres et à false pour les tableaux de booléens. En Java, on peut déclarer un tableau de différentes manières.

Tableaux

```
1  int tab1[] = {1, 2, 3};// identique au langage C
2
3  int[] tab2 = {1, 2, 3};
4
5  int[] tab3 ;
6  tab3=new int[6];/tab3 fait reference a un tableau de 6 entiers
7
8  int []t1,t2;//t1 et t2 sont des references a des tableaux d'entiers
9
10 int t[10] ; // erreur: on ne peut pas indiquer de dimension ici
```

Tableaux

```
1 // ces trois déclarations sont équivalentes
2 int t [] [] ;
3 int [] t [] ;
4 int [] [] t ;
5
6 //un tableau de deux lignes et un nombre varie des colonnes
7 int t [] [] = {new int [3], new int [2]};
```

Tableaux

R Les éléments d'un tableau peuvent être de type quelconque, et pas seulement d'un type primitif. Il est possible en Java de déclarer des tableaux des objets, par exemple :

```
1 Person [] t ;
2 t = new Person[3] ;//Person est une classe  définit par le
      programmeur
3
4 String[] prenoms = new String [2]; //tableau de 2 String
5 prenoms[0]="Michel";//prenoms[0] référence la chaine "
     Michel"
6 prenoms[1]="Josette";//prenoms[1] référence la chaine "
     Josette"
```

La même chose pour les tableaux à deux dimensions, on peut déclarer des tableaux d'objets à deux dimensions.

Tableaux

Pour itérer ou parcourir un tableau, on utilise une boucle ***for*** (ou ***while***). Aussi, il est nécessaire de connaitre la taille d'un tableau. Pour cela, en java, on utilise l'instruction ***tableau.length***.

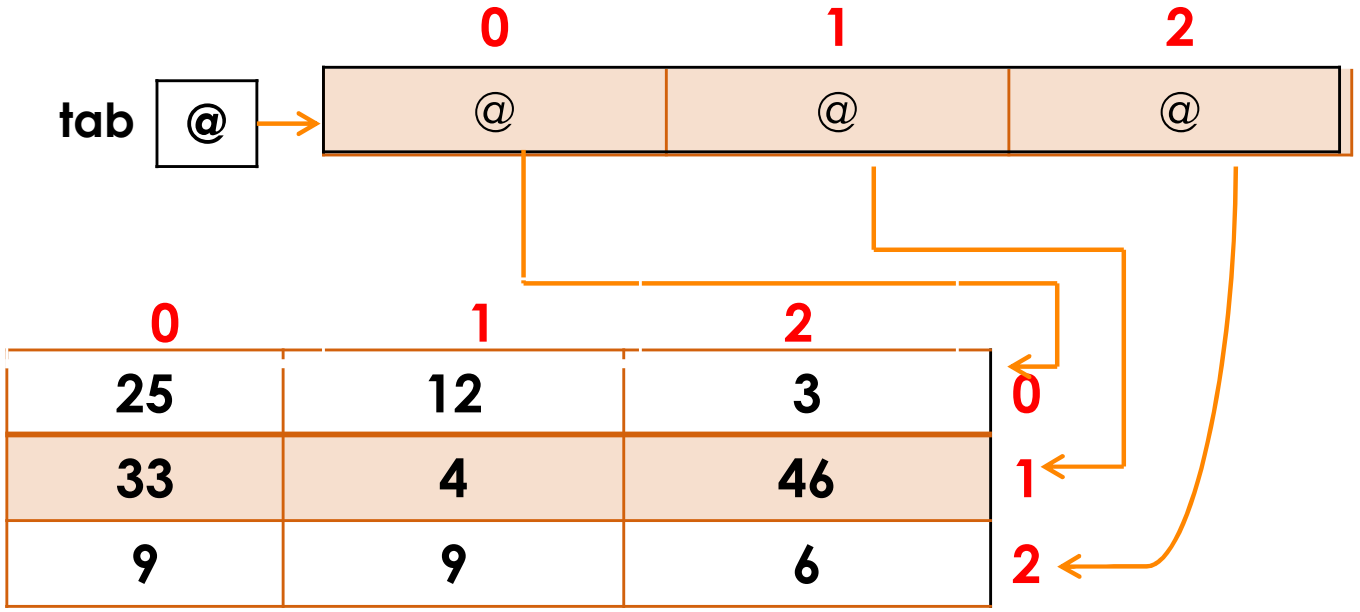
Tableaux

Pour itérer ou parcourir un tableau, on utilise une boucle ***for*** (ou ***while***). Aussi, il est nécessaire de connaitre la taille d'un tableau. Pour cela, en java, on utilise l'instruction ***tableau.length***.

```
1  int t[] = new int[5] ;//tableau de 1 dimension
2  int somt=0, somtt;
3  for (int i =0; i<t.length; i++) somt+=t[i] ;
4  System.out.println("La somme des élément de t="+som);
5
6  int tt[] = new int[5][7] ;//tableau de 2 dimension
7  for (int i =0 ; i<tt.length ; i++)
8  {
9    for (int j=0; j<tt[i].length; j++) somtt+=t[i][j] ;
10  }
11 System.out.println("La somme des élément de tt="+somtt);
```

Tableaux

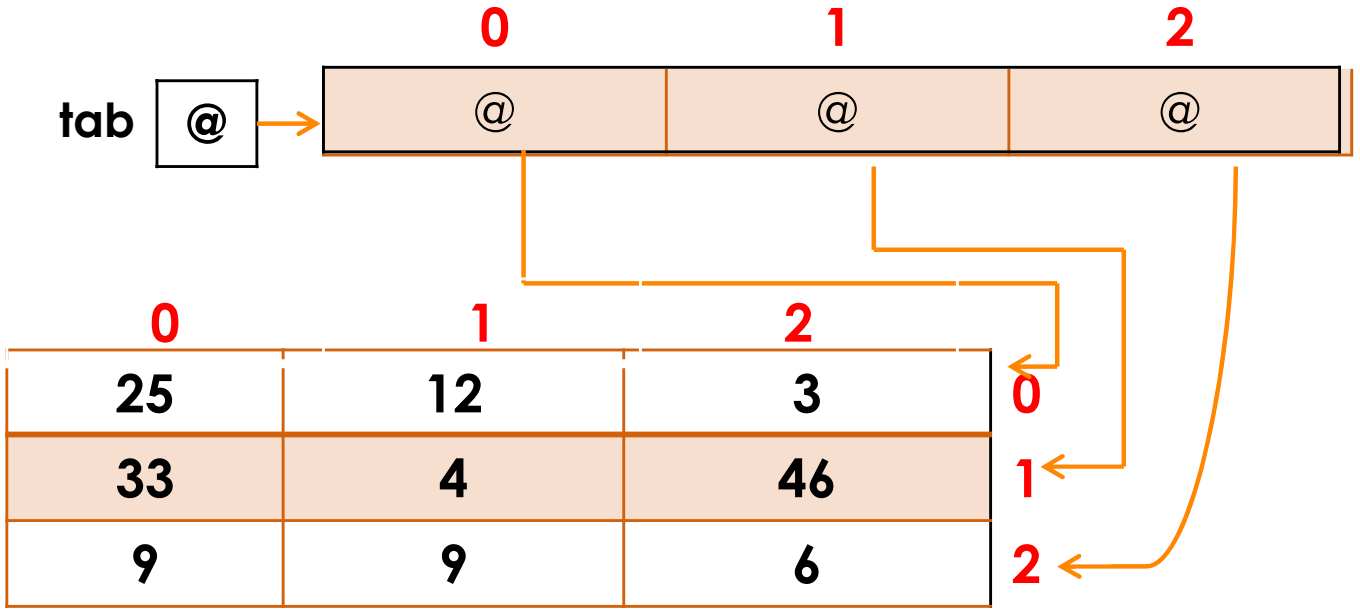
```
1  int [][]tab=new  int [3][3];
2  // i=0
3  tab[0][0]=25;tab[0][1]=12;tab[0][2]=3;
4  // i=1
5  tab[1][0]=33;tab[1][1]=4;tab[2][2]=46;
6  // i=2
7  tab[2][0]=9;tab[2][2]=9;tab[2][2]=6;
8
9  //une autre maniere d'initialisation du tab
10 int [][]tab={{25,12,3},{33,4,46},{9,9,6}};
```



Tableaux

```java
int [][]tab=new int [3][3];
// i=0
tab[0][0]=25;tab[0][1]=12;tab[0][2]=3;
// i=1
tab[1][0]=33;tab[1][1]=4;tab[2][2]=46;
// i=2
tab[2][0]=9;tab[2][2]=9;tab[2][2]=6;

//une autre maniere d'initialisation du tab
int [][]tab={{25,12,3},{33,4,46},{9,9,6}};
```



Chaines de caractères

pour créer et manipuler des chaînes de caractères, l'environnement JAVA définit trois classes: **String**, **StringBuilder** et **StringBuffer**.

Pour initialiser une variable de type **String**, on utilise les **guillemets** ""

par exemple : ***String m = "Bonjour"; Ou:***

String m=new String("Bonjour");

Comme les tableaux, une variable de type **String** contient une **référence** vers une chaine de caractères. Donc, la sémantique des opérateurs **d'affectation** et **d'égalité** est la même que les tableaux.

Chaines de caractères

Méthode	Résultat renvoyé
length	Nombre de caractères contenus dans la chaîne courante
charAt	Le caractère dont la position dans la chaîne courante est passée en paramètre
compareTo	Compare la chaîne courante avec une chaîne passée en paramètre. renvoie zéro si les chaînes sont identiques, une valeur entière négative si l'objet courant est inférieur au paramètre, et une valeur entière positive si l'objet courant lui est supérieur
endsWith	Teste si la chaîne courante se termine par le suffixe passé en paramètre. Renvoie *true* ou *false*
equals	Compare cette chaîne à l'objet spécifié. Le résultat est vrai si et seulement si l'argument n'est pas *nul* et est un objet String qui représente la même séquence de caractères que cet objet
equalsIgnoreCase	Compare cette chaîne à une autre chaîne, en ignorant la casse (majuscule et minuscule). Renvoie *true* ou *false*
contains	Renvoie *true* si et seulement si cette chaîne contient une sous chaîne passée en paramètre
indexOf	renvoie la première position du caractère passé en paramètre dans la chaîne courante
isEmpty	Renvoie *true* si, et seulement si, *String.length* vaut 0
toLowerCase	Convertit tous les caractères de cette chaîne en minuscule
toUpperCase	Convertit tous les caractères de cette chaîne en majuscule

Chaines de caractères

```
1  String c1 = new String ( "bonjour " ) ; // ou String c1="bonjour "
2  String c2 ;
3  c2 = c1 + "á tous " ;
4  String c3 ="BonJour á Tous ";
5  System.out.println( c2 ) ;// bonjour á tous
6  System.out.println(c3) ;// BonJour á Tous
7  System.out.println( c2.length());
8  System.out.println ( c2.charAt( 3 ));
9  System.out.println ( c2.indexOf('o'));
10 System.out.println( c1.contains("á tous "));
11 System.out.println( c3.equalsIgnoreCase(c2));
```

R Les chaînes de caractères d'un objet de type String ne peuvent pas être modifiés. Une même chaîne de caractères de la classe String peut être référencée plusieurs fois puisqu'on est sûr qu'elle ne peut pas être modifiée. Par contre, la classe *StringBuffer* permet la création et la modification d'une chaîne de caractères.

Tableaux dynamiques

Les **tableaux dynamiques** (**liste**) ont la particularité de pouvoir changer de taille pendant l'exécution du programme. En Java, on les définit à l'aide de type ***ArrayList*** de la bibliothèque:

java.util.ArrayList.

Pour ce faire, en écrit :

```
ArrayList <Type > myList ;
myList = new <Type > ArrayList ();
```

Le type des éléments d'un ***tableau dynamique*** ne peut être un type simple (***int*** ou ***double*** par exemple).

Tableaux dynamiques

Il doit être l'un des types **évolués** correspondant aux types de données simples. Par exemple : **Integer** pour **int**, **Float** pour **float**, **Double** pour **double**, etc.

Comme il peut être une classe spécifique (**Person** par exemple).

On peut aussi déclarer un tableau dynamique comme suit :

```
1 ArrayList  myList;
2 myList = new  ArrayList();
```

Dans cette déclaration, il n'y a **aucune restriction** sur le **type** d'objets qu'on peut ajouter au tableau

Tableaux dynamiques

- ☆ *tableau.add*(*valeur*) ajoute *valeur* à la fin de tableau
- ☆ *tableau.size*() : renvoie la taille du tableau (de type int) ;
- ☆ *tableau.set*(*i*, *valeur*) : affecte valeur à la case i du tableau
- ☆ *tableau.get*(*i*) : renvoie l'élément à l'indice *i* dans le tableau (*i* doit être un entier compris entre 0 et *tableau.size*()-1)
- ☆ *tableau.isEmpty*() : détermine si le tableau est vide ou non
- ☆ *tableau.clear*() : supprime tous les éléments du tableau
- ☆ *tableau.remove*(*i*) : supprime l'élément à l'indice i du tableau

```
1 ArrayList <String> sample = new ArrayList<String>();
2 sample.add("One Java");
3 sample.add("Two Java");
4 sample.add("Three Java");
5 System.out.println(sample.size());
6 System.out.println(sample.get(2));
7 System.out.println(sample.get(3));
```

```
1 //soit les tableaux suivants, tab1, tab2, tab3 et m
2 //Rechercher et afficher la valeur max dans les trois premiers
      tableaux
3
4           int tab1[]={2,5,16,9,3};
5           int tab2[]={1,5,6,9,13};
6           int tab3[]={12,5,6,9,3};
7           int m[][]={{12,5,6,9,3}, {1,5,6,9,13}, {2,5,16,9,3}};
8 //
```