Chapitre 02: Bases du langage Java

Plan

- Types de données
- Les variables
- Les opérateurs
- Casting implicite et explicite
- Structures conditionnelles
- Structures répétitives
- Lecture en java
- affichage en java
- Convention d’utilisation
- Les commentaires

Types de données

- En Java, nous avons 2 types de variables :

1. variables de type simple ou « primitif »
2. variables de type complexe ou des « objets »

- ✓ Les types référencés réprésentés par les classes et les tableaux

Types primitifs (1/2)

- Il existe 08 types primitifs
  - 4 type numérique entier : byte , short, int, long
  - 2 type numérique réel: double et float
  - 1 type caractère : char
  - 1 type booléen : false et true
- Chaque type est défini par une taille
- Une valeur minimale et une valeur maximale

Types primitifs(2/2)

Primitive	Signification	Taille (en octets)	Description
byte	entier très court	1	Stoke les nombres entiers de: -128 à 127
short	entier court	2	Stoke les nombres entiers de: -32768 à 32767
int	entier	4	Stoke les nombres entiers de: -2 147 483 648 à 2 147 483 647
long	entier long	8	Stoke les nombres entiers de: 9223372036854775808 à 9223372036854775807
float	flottant (réel)	4	-1.4*10-45 à 3.4*1038
double	flottant (double)	8	4.9*10-324 à 1.7*10308
boolean	booléen	1	stoke les valeurs vrai ou faux
char	caractère	2	stoke un seul caractère

Types référencés

☐Les types objets qui sont manipulés par leur référence:

- Exemple :
  - Object, String,int [ ]
  - Une chaîne est peut être considérée comme une variable du langage Java
  - Déclarée à l'aide de l'instruction String.
- Exemple :
  - String Chaine = "bonjour" ;

Les variables (1/2)

☐ La déclaration d'une variable

Doit contenir :

✓ Un nom

✓Le type de données qu'elle peut contenir.

Permet de :

Réserver la mémoire pour en stocker la valeur.

- Exemple :

- int i;
- double x;
- char c;

Les variables (2/2)

☐ Le nom peut être initialisé au même temps que la déclaration (type var= valeur).

- Exemple :

- int i = 5;
- double x= 1000.00
- long a=400;

☐ Par convention les noms de variables commencent par une minuscule

Variables numériques (1/2)

- Six types primitifs: byte, short, int, long, float, double
- (variables entières seulement)
- Nombres à virgule flottante : float,double

- Un nombre entier est un nombre sans virgule qui peut être exprimé dans avec une base hexadécimale ou octale

Variables numériques (2/2)

- Exemple :
- int v = 79; (int v = 0x4f; int v =0117)
- Un nombre réel est un nombre flottant peut être exprimé par un exposant ou l'un des suffixes f, F, d, D.
- Il est possible de préciser des nombres qui n'ont pas le partie entière ou décimale.
- Exemple :
- float pi = 3.141f;
- double v = 3d
- float f = +.1f , d = 1e10f;

Les caractères

☐Le type primitif char représente les caractères.

- Le codage retenu est l'unicode (16 bits)
- Exemples:
- char c= 'c' // code ASCII 99 (en décimal)
- c = '\u00e7'; // code unicode de ç
- c = '\t'; // la tabulation
- c = '\n'; // le saut de ligne
- c = '\\'; // le slash (\)
- c = '\''; //le quote (')

Les booléens

- Le type primitif boolean représente les booléens. Le codage retenu est l'unicode (16 bits)

- Exemples :
  - boolean fini= true;
  - boolean infini= false;

Les constantes

☐On déclare les constantes par le mot clé final

- Le nom de la constante doit être en majuscule
- Si le nom est composé de plusieurs mots, on utilise «_» pour les séparer

- Exemple :
- final int TAILLE ;
- final int MAX_VALUE;
- On ne peut plus changer la valeur d'une constante une fois qu'elle a été initialisée

Opérateurs arithmétiques

Arithmetic Operators

Operator	Meaning	Example	Result
+	Addition	10 + 2	12
-	Subtraction	10 – 2	8
*	Multiplication	10 * 2	20
/	Division	10 / 2	5
%	Modulus (remainder)	10 % 2	0
++	Increment	a++ (consider a = 10)	11
--	Decrement	a-- (consider a = 10)	9
+=	Addition Assignment	a += 10 (consider a = 10)	20
-=	Subtraction assignment	a -= 10 (consider a = 10)	0
*=	Multiplication assignment	a *= 10 (consider a = 10)	100
/=	Division assignment	a /= 10 (consider a = 10)	1
%=	Modulus assignment	a %= 10 (consider a = 10)	0

Opérateurs logiques

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 ! && x < 10)

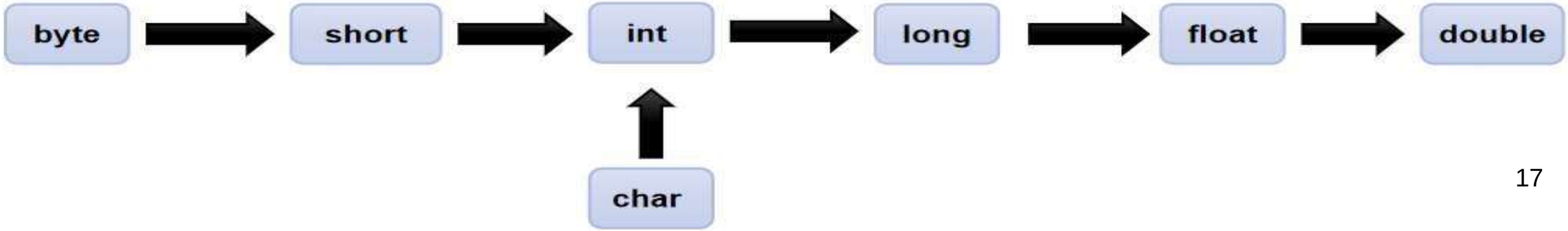
Opérateurs relationnels

Opérateur	Signification	Exemple
>	Supérieur	a>b
<	Inférieur	a<b
>=	Spérieur ou égal	a>=b
<=	Inférieur ou égal	a<=b
==	Egal	a==b
!=	Non égal (différent !)	a!=b

Casting implicite et explicite(1/4)

- Le **casting** ou **transtypage** permet de convertir une variable à partir de son type d'origine, vers un autre type au moment de son utilisation.
- Conversion implicite(automatique) se fait sans risque de perdre l'information vers un type plus large

- Exemple
  - int x=10, long z=x ; // z=10 ;



Casting implicite et explicite (2/4)

- Casting explicite est forcé afin d'éviter la perte d'information vers un type plus étroit

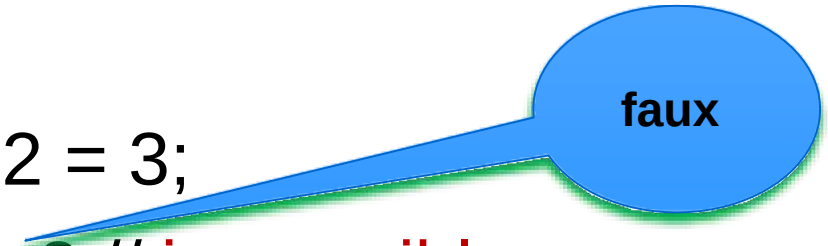
- Exemple :

```
double d=123.789;
float f=(float)(d); // convert double into float f=123.789
```

Casting implicite et explicite (3/4)

- Ecriture du nouveau type entre parenthèse avant la variable à caster.

- Exemple:
- double nbre1 = 10, nbre2 = 3;
- int resultat = nbre1 / nbre2;// impossible



- Solution :
- int resultat = (int)(nbre1 / nbre2);//affiche 3
- System.out.println("Le résultat est = " + résultat);

Casting implicite et explicite(4/4)

☐D'un type float en type int :

- float i = 2.0f;
- int t= (int) i; // =2

☐ D'un type double en int :

- double i = 123.0;
- int z = (int)i; // vaut 123

Structure conditionnelle if (1/2)

- L'instruction if teste une condition booléenne, si le résultat est vrai, le bloc d'instructions suivant la condition est exécuté,sinon il ne l'est pas.

if (condition) bloc1

- Optionnellement, le premier bloc sera suivit du mot clé else et d'un second bloc exécuté seulement si la condition est fausse :

if (condition) bloc1 else bloc2

Structure conditionnelle if (1/2)

```
public class TestIf
{
 public static void main (String[] args)
{
int i = 0;
 if (i < 0) { System.out.println("Ce nombre est négatif
!");
}
else { if (i == 0) System.out.println("Ce nombre est
nul !");
 else System.out.println("Ce nombre est positif !");
 }
}
}
```

Résultat :
Ce nombre est nul

Structure conditionnelle switch (1/2)

- L'instruction switch cherche le résultat d'une expression (de type char ou int) dans une liste de cas et exécute la suite correspondante :

switch (expression)

{case cas1:suite1 case cas2:suite2 ... case cas_n:suite_n}

- Optionnellement le dernier cas peut être le mot clé default si aucun cas ne correspond :

switch (expression)

{case cas1:suite1 case cas2:suite2 ... case cas_n:suite_n

default:suite}

- Pour sortir du switch un break est utilisé à la fin.

Structure conditionnelle switch (2/2)

```
 public class TestSwitch
{
public static void main (String[] args)
   { int a = 1 ;
   switch (a)
    { case 0  : System.out.println("NUL") ; break
;
      case 1  : System.out.println("MOYEN") ;
break;
      case 2  : System.out.println("GRAND") ;
break ;
      default : System.out.println("Inconnu") ;
    }
     System.out.println("Fin") ;
    }
    }
```

Résultat :
MOYEN
Fin

Structure répétitive while (1/3)

- L'instruction while évalue une condition puis répète les instructions du bloc tant que cette condition est vraie

while (condition) bloc

- Exemple

```
public class TestWhile

{ public static void main (String[] args)
 { int i = 0 ;
   while (i<3)
    { System.out.println("i vaut "+i) ;
      i++ ;
   }
  }
}
```

Résultat :
i vaut 0
i vaut 1
i vaut 2

Structure répétitive do while(2/3)

- L'instruction do...while exécute une suite puis évalue une condition et répète bloc tant que condition est vraie

do suite while (condition) ;

- Exemple

```
public class TestDoWhile

{ public static void main (String[] args)
{ int i = 0 ;
do
{ System.out.println("i vaut "+i) ;
i++ ;
}
while (i<3) ;
}
}
```

Résultat :
i vaut 0
i vaut 1
i vaut 2

Structure répétitive for(3/3)

- L'instruction for exécute une instruction1 puis répète les instructions du bloc suivant l'instruction for tant que sa condition est vraie, à la fin de chaque répétition elle exécute son instruction2 :

for (instruction1;condition;instruction2) bloc

- Exemple

```
public class TestFor
{
public static void main
(String[] args)
  { for (int i=0;i<3;i++)
    System.out.println("i vaut "+i)
;
  }
}
```

Résultat :
i vaut 0
i vaut 1
i vaut 2

Lecture en java (1/2)

- Pour récupérer les valeurs saisies au clavier, le langage Java est doté d'une classe spéciale nommée Scanner.
- Pour utiliser cette classe on doit :

- Importer le package (bibliothèque) : import java.util.Scanner ;
- Faire une instanciation : Scanner saisie = new Scanner(System.in) ;
- Guider l'utilisateur avec un message qui s'affiche à l'écran : System.out.println(" Tapez votre message ") ;
- Récupérer la saisie clavier dans une variable :

  String phrase = saisie.nextLine() ;
- Afficher le résultat à l'écran : System.out.println(" vous avez saisie " + phrase) ;

Lecture en java (2/2)

- Pour chaque type de variables une méthode est utilisée:

méthode	Description
nextByte()	lecture des entiers de type byte
nextLong()	lecture des entiers de type long
nextInt()	lecture des entiers de type int
nextFloat()	lecture des entiers de type float
nextDouble()	lecture des entiers de type double
nextLine()	lecture des chaines de caractères

Affichage en java(1/2)

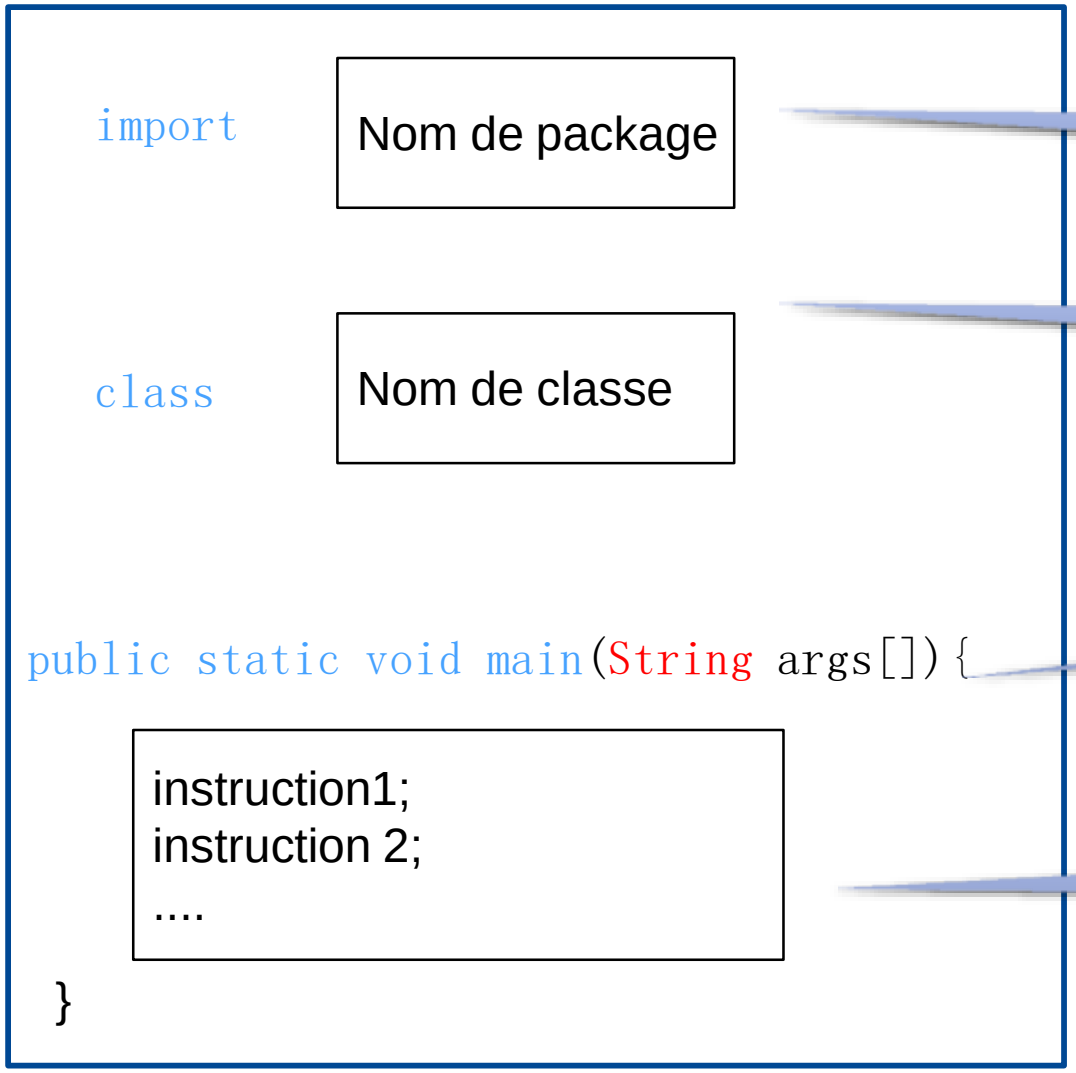
- L’affichage se fait en utilisant les deux fonctions:
- print et printf
- print : permet d’afficher les valeurs sans préciser le format
  - Exemple : System.out .println (argument)
- printf: permet de faire un affichage formaté :
  - contient le texte à afficher
  - le code de formatage : commencent par % suivi par le type de valeur à afficher
  - Exemple :
- System.out.printf(code de formatage, argument)
- On peut ajouter un point décimale et un chiffre pour indiquer le nombre des chiffres après la virgule

Affichage en java(2/2)

- Exemple :
- double vitesse=765,98
- System.out.printf("la vitesse est%.4f KM/H", vitesse)
- vitesse= 765,9800

Type de variable	Code de formatage	Exemple
int	%d	123
double	%f	45.9807
booléen	%b	true ou false
caractère	%c	‘a’
chaine de caractère	%s	"programmer avec java"

Structure d’un programme java



importation des packages

Nom de la classe

Méthode main principale point de départ

Bloc d’instructions à exécuter

Exemple 1

```
import java.util.Scanner;
public class Testage {
public static void main(String[] args) {
System.out.println("Veuillez saisir votre âge !");
Scanner age=new Scanner(System.in);
int tonAge=age.nextInt();
if ( tonAge < 18 ) {
System.out.println("vous êtes mineur"+" Votre âge est : "+tonAge);
}
else{
System.out.println("vous êtes Majeur"+" Votre âge est
: "+tonAge); }

}
}
```

Exemple 2

```
import java.util.Scanner;
public class surface {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Entrez la longueur du rectangle: ");
 double longueur = sc.nextDouble();
 System.out.print("Entrez la largeur du rectangle: ");
 double largeur = sc.nextDouble();
   double surface = longueur * largeur;
   System.out.println("La surface du rectangle est :
"+surface);
}
}
```

- Ecrire un programme Java qui demande à l’utilisateur de saisir un nombre entier N et afficher la somme des nombres de 1 jusqu’a N.

Solution

```
import java.util.Scanner;
 public class sommeDesNpremiersEntiers {
 public static void main(String[] args) {
 Scanner sc=new Scanner(System.in);
 System.out.println("Saisissez la valeur de N");
 int N=sc.nextInt();
 int j=0;
 for(int i=1;i<=N;i++) {
 j=j+i;
 }
 System.out.println("La somme des " + N +" premiers nombres est : "+j);
}
}
```

Conventions d’utilisation

- Les noms de classes commencent par une majuscule (ce sont les seuls avec les constantes) : Voiture, Object

- Les constantes sont en majuscules et les mots sont
  - séparés par le caractère souligné « _ » :
  - UNE_CONSTANTE

- Il est préférable d’utiliser des noms pour les classes et des verbes pour les méthodes : getName(), afficherSomme(), calculerPrix()

Les commentaires

- Le but des commentaires est d'expliquer les parties du programme et le rendre plus facile à comprendre
- Java permet d'ajouter les commentaires sous 3 formes :
- Sur une seule ligne en utilisant //

• Exemple :

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

- Sur plusieurs lignes en utilisant /*

• Exemple :

```
/* cela est un commentaires Sur plusieurs lignes */
```

- Dans javadoc en utilisant /**

• Exemple :

```
/** cela est un commentaire javadoc sur plusieurs lignes*/
```

Fin du chapitre 2