Le langage Java

M. Berthouzoz
Cours préparé par L. Nerima
Université de Genève

Les commentaires

- Trois styles :

Style	Définition
`/* commentaire */`	Tous les caractères compris entre /* et */ sont ignorés
`// commentaire`	Tous les caractères compris entre les // et le retour de la ligne sont ignorés
`/** commentaire */`	Tous les caractères compris entre /** et */ sont ignorés par le compilateur mais peuvent être traités par des outils de documentation automatique

- Bonnes habitudes : commenter les programmes !
- Exemple :

  int zoom = 2 // valeur par défaut du zoom au démarrage

- Exemple de mauvais commentaire :

  int zoom = 2 // zoom à 2

Les identificateurs

- Exemples d'identificateurs valides en Java :

  $valeur_system

  dateDeNaissance

  ISO9000

- Exemples d'identificateurs non valides en Java :

```
ça          // ç comme premier caractère
9neuf       // 9 comme premier caractère
note#       // # pas au-dessus de 0X00C0 (U+00C0)
long        // mot réservé
```

Les mots réservés

abstract	boolean	break	byte	byvalue[a]
case	catch	char	class	const[b]
continue	default	do	double	else
extends	false	finally	float	for
goto[c]	if	implements	import	instanceof
int	interface	long	native	new
null	package	private	protected	public
return	short	static	super	switch
synchronized	this	throw	throws	transient
true	try	void	while	

a. Mot réservé mais pas utilisé actuellement.
b. Idem.
c. Idem.

La déclaration des variables

- Toutes les variables doivent être déclarées avant d'être utilisées
- Le type des variables doit être précisé à la déclaration (typage fort)
- Le type précède l'identificateur de la variable
- Lors de la déclaration il est possible d'initialiser la variable (mais pas obligatoire)
- Les variables peuvent être déclarées n'importe où (p.ex. dans un bloc d'instructions)
- Portée des variables : la variable est visible à l'intérieur du bloc où elle est déclarée

L’affectation des variables

- L’opérateur d’affectation (assignation) est le symbole =

```
j=2;        // expression d’assignation d’un littéral
i=j*3;      // expression d’assignation d’une expression
```

Les types de base (1)

Booléens (*boolean*)

```
boolean ok;
boolean p, q, r;
boolean resultatAtteint = false;
boolean continuer = true;
```

Nombres entiers (*byte*, *short*, *int* ou *long*)

```
int nbrDeMois = 12;
int nbrDeDoigts = 012;          // octal, = 10 en décimal
int DixHuit = 0x12              // hexadécimal, = 18 en décimal
```

Les types de base (2)

Nombres réels (*float* ou *double*)

```
double r;
double s = 0.01
double e = 2.5E-3;
```

Caractères (*char*)

```
char c = ‘a’;
char q = ‘4’;
char a = ‘\n’;
```

Définir une constante

- Avec le mot-clé *final*

  final int NbreDeRoues=4;

  final double pi=3.14159;

- La valeur de *NbreDeRoues* et de *pi* ne pourra pas varier (erreur à la compilation)

Les tableaux (1)

- Les tableaux sont déclarés en post-fixant le type ou la variable par [] :

```
int i[ ];                    // vecteurs d'entiers
int[ ] j;                    // autre notation pour le même type que i
```

Pour déclarer un tableau de dimension > 1 :

```
float f [ ] [ ] [ ];         // tableau de réels de dimension 3
```

- Les vecteurs ne sont pas contraints par des bornes au moment de leur déclaration

Les tableaux (2)

- L'allocation de l'espace nécessaire au vecteur se fera explicitement au moyen d'une méthode *new*
- On peut cependant allouer de l'espace au tableau lors de la déclaration :

```
int k[ ] = {1, 1, 2, 3*t, 5, 8, u-13, 21};
double[ ][ ] e1 = {{0.0, 1.0}, {-1.0, 0.0}};
```

- Allocation de l'espace avec *new* :

```
int[ ] m = new int [10], int[ ] n = new int [10];
int x = 45, y = 19, int j = 0;
m[0] = x
n[j] = y
```

Les chaînes de caractères

- Suites de caractères entourées par des guillemets simples ou doubles
- Le type est *String*
- En Java, le type *String* est une classe à part entière (prédéfinie)
- Ce n'est **PAS** un tableau de caractères

Les opérateurs numériques (1)

- Opérateurs unaires

Opérateur	Action	Exemple
-	Négation	`i=-j;`
++	Incrémentation de 1	`i++;`
--	Décrémentation de 1	`i--;`

Les opérateurs numériques (2)

- Opérateurs binaires

Opérateur	Action	Exemple
+	Addition	i=j+k
+=		i+=2; //i=i+2
-	Soustraction	i=j-k;
-=		i-=j; //i=i-j
*	Multiplication	x=2*y
*=		x*=x; //x=x*x
/	Division (tronquée si les arguments sont entiers)	i=j/k;
/=		x/=10; //x=x/10
%	Modulo	i=j%k;
%=		i%=2; //i=i%2

Les opérateurs relationnels

Opérateur	Action	Exemple
<	Plus petit que	`x<i;`
>	Plus grand que	`i>100;`
<=	Plus petit ou égal à	`j<=k;`
>=	Plus grand ou égal à	`c>='a';`
==	Egal à	`i==20;`
!=	Différent de	`c!='z';`

Les opérateurs logiques

Opérateur	Action	Exemple
`!`	Négation	`!p;`
`&`	ET	`p & (i<10)`
`|`	OU	`p | q`
`^`	OU exclusif	`p ^ false`
`&&`	ET évalué	`p && q && r`
`||`	OU évalué	`p || q || r`
`!=`	Négation assignée	`p!=p;`
`&=`	ET assigné	`p&=q // p= p & q`
`|=`	OU assigné	`p|=q // p= p | q`
`?:`	Si alors sinon	`(i<10)?(j=2):( j=3)`

Précédence des opérateurs

Précédence des opérateurs (de la plus grande à la plus petite) Les opérateurs de même niveau sont évalués depuis la gauche				
.	[]	()		
++	--	!	~	instanceof
*	/	%		
+	-			
<<	>>	>>>		
<	>	<=	>=	
==	!=			
&				
^				
&&				
||				
?:				
=	op=			
,				

Précédence des opérateurs – Exemple

- Prenons : x=z+w-y/(3*y%2)
- L'ordre de lecture (de gauche à droite) est le suivant :

  = + - / ()
- Avec la table de précédence, l'ordre d'**exécution** est le suivant :

  () / + - =
- À l'intérieur des parenthèses (), nous avons * puis %, par ordre de lecture et par ordre d'exécution

Les structures de contrôle (1)

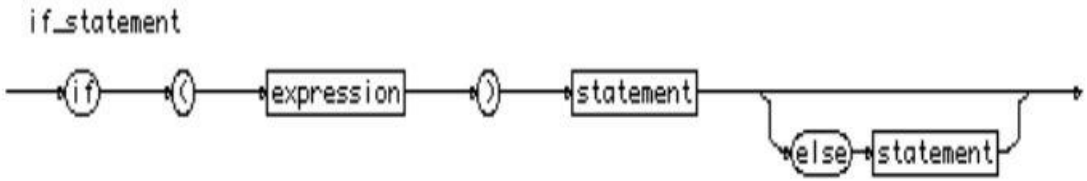
- Le bloc d'instructions est délimité par des accolades { }

- Exemple :

```
{
   int a, b = 10, c = 3, d = 5, j = 0, i;
   i = j;
   System.out.println(" a = " + a);
   a = b / c;
   System.out.println(" a = " + a);
   d *= 5;
   j++;
   System.out.println(" d = " + d + " " + "j = " + j);
}
```

Les structures de contrôle (2)

- Instruction *if* :



- Exemple :

```
if (i == 1){
   s = "i est égal à 1";
   i = j;
}
else{
   s = "i est différent de 1";
   i = 1515;
}
```

Les structures de contrôle (3)

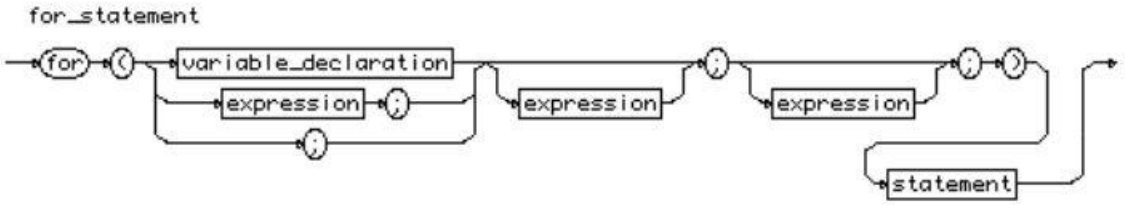
- Boucle *while* :

- Exemple :

```
class TestWhile{
     public static void main (String args[]) {
        int n=100, somme_i=0, j=1;
        // boucle calculant la somme de la série S = 1 + 2 + … + n
        while (j<=n) {
           somme_i+=j;
           j++;
          }
        System.out.println("boucle  1:" + somme_i);
        // même somme mais boucle écrite de manière condensée
        somme_i=0; j=0; // attention: il faut que j=0 (et non 1)
        while (++j<=n ) somme_i+=j;
        System.out.println("boucle  2:" + somme_i);
     }
  }
```

Les structures de contrôle (4)

- Boucle *for* :



- Exemple :

```
class LabelInst{
  public static void main (String args[]){
   boucle1:
        for (int i=0;i<10;i++){
             for (int j=0;j<10;j++){
               System.out.println(i+"/"+j);
             }
        }
   }
}
```

Les structures de contrôle (5)

- Boucle *do… while* :

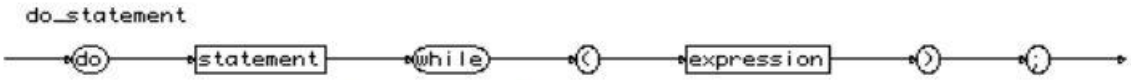


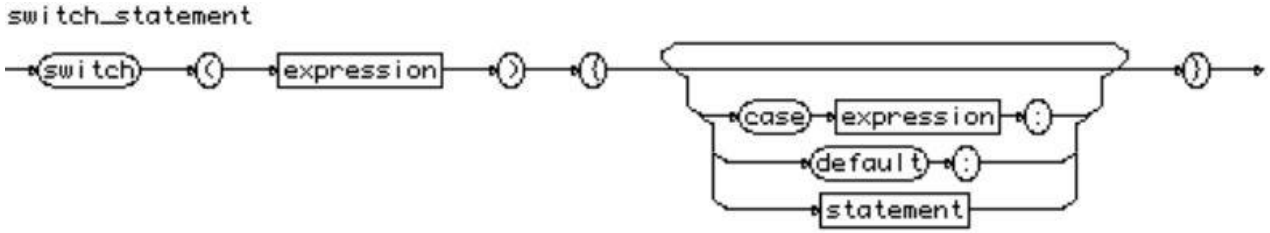
Diagramme 5.6 do_statement

- L'instruction est exécutée au moins une fois
- Exemple :

```
class TestDo{
   public static void main (String args[]) {
        int n=100, somme_i=0, j=1;
        do {
           somme_i+=j;
           j++;
        } while (j<=n);
        System.out.println("boucle 1:"+somme_i);
   }
}
```

Les structures de contrôle (6)

- Instruction *switch* :



1. Évalue l'expression du *switch*
2. Compare le résultat avec chaque expression des *case*
3. S'il y a égalité, exécute les instructions du *case*
4. Continue avec l'exécution des instructions des *case* suivants

Si on veut qu'un seul *case* soit exécuté, tous les *case* doivent terminer par *break*

Les structures de contrôle (7)

- Instruction *switch* – Exemple

```
class TestSwitch{
   public static void main (String args[]) {
        int i=3;
        switch (i) {
               case 1: System.out.println("I"); break;
               case 2: System.out.println("II"); break;
               case 3: System.out.println("III"); break;
               case 4: System.out.println("IV"); break;
               case 5: System.out.println("V"); break;
               default: System.out.println("pas de
   traduction");
               }
        }
}
```

L'instruction *break* (1)

- Pour interrompre l'itération

```
class TestBreak{
   public static void main (String args[]) {
       bloc1:
          for (int i=0;i<10;i++){
             for (int j=0; j<10;j++){
                System.out.println(i+"/"+j);
                // quitte la boucle locale (for j…)
                if (j==1) break;
                // quitte le bloc complet
                if (i==2) break bloc1;
             }
          }
       }
}
```

L'instruction *break* (2)

- Exécution du programme *TestBreak*

0/0
0/1
1/0
1/1
2/0

L'instruction *continue* (1)

- Pour passer directement à l'itération suivante

```
class TestContinue{
   public static void main (String args[]) {
        bloc3:
           for (int i=0;i<10;i++){
              for (int j=0; j<10;j++){
                 // passe au tour suivant de la boucle locale
                 if (j>1) continue;
                 // passe au tour suivant de for i…
                 if (i>2) continue bloc3;
                 System.out.println(i+"/"+j);
              }
           }
        System.out.println("en dehors des boucles");
   }
}
```

L’instruction *continue* (2)

- Exécution du programme *TestContinue*

0/0
0/1
1/0
1/1
2/0
2/1
en dehors des boucles