

TABLE DES MATIÈRES

INTRODUCTION À LA PROGRAMMATION JAVA.....	2
1. SQUELETTE D'UN PROGRAMME JAVA.....	2
2. CONCEPT DE VARIABLE.....	3
3. CONCEPT DE CONSTANTE.....	3
4. LE TYPAGE.....	3
5. DÉCLARATIONS.....	5
6. LA NOTION D'EXPRESSION.....	6
7. LA NOTION D'AFFECTATION.....	8
LES ENTRÉES / SORTIES EN JAVA.....	10
1. AFFICHAGE DANS LA CONSOLE.....	10
2. AFFICHAGE DANS UNE BOÎTE DE DIALOGUE.....	12
3. SAISIE DES DONNÉES À PARTIR D'UNE BOÎTE DE DIALOGUE.....	14
4. SAISIE DES DONNÉES À PARTIR DE LA CONSOLE.....	17

Introduction à la programmation Java

1. Squelette d'un programme Java

Un programme Java est formé d'**instructions** précisant les différents traitements qui doivent être faits.

Les **applications Java** sont des programmes autonomes qui résident sur la machine qui les exécute. Elles nécessitent que la Machine virtuelle Java (JVM) soit installée et ne peuvent pas être intégrées dans une page Web.

Toute application Java doit posséder une méthode **main**. C'est de là que démarre le programme. Toutes les instructions sont exécutées à partir de ce point.

Modèle de programme pour une application :

Commentaires d'en-tête	<pre>/* * NomFichier.java * Description de ce que fait le programme * Auteur: votre nom * Date: la date */</pre>
Directives	<pre>import ...</pre>
Déclaration de la classe	<pre>public class NomFichier {</pre>
Déclaration des attributs	
Déclaration de la méthode main	<pre> public static void main(String args[]) { // Déclarations des constantes // Déclarations des variables // Traitements (calculs, affichages...) } // fin du main</pre>
Déclaration des autres méthodes	
Fin de la classe	<pre>} // fin de la classe</pre>

2. Concept de variable

Une **variable** est un emplacement en mémoire représenté par un nom (une suite de caractères) que l'on utilise pour conserver une donnée.

- ☞ Il est important de choisir le nom des variables de manière à rappeler la signification de la donnée qu'elle représente.
- ☞ Par convention, un nom de variable commence par une minuscule, chaque nouveau mot commence par une majuscule. On peut inclure des chiffres dans un nom mais on devrait éviter les accents.
- ☞ Une variable est représentée par : un **nom**, une **valeur** et un **type**.

Exemples :

nb2
12

sommeNb
4.5

nomEleve
"Nathalie"

3. Concept de constante

Une valeur **constante** peut être écrite telle quelle dans un programme. C'est une valeur qui ne changera jamais.

Exemples :

- ☞ 0.07
- ☞ 235
- ☞ "Bonjour"

Mais, dans bien des cas, il est préférable de donner un nom **symbolique** aux **constantes**, comme pour les variables. Par convention, les noms de constantes sont en majuscules, chaque nouveau mot étant séparé du précédent par le caractère _.

```
TAUX_TVA = 0.07;  
MESSAGE = "Bonjour";
```

4. Le typage

En Java, comme dans la plupart des langages évolués, toute donnée (*variable* ou *constante*) doit obligatoirement être déclarée avec un **type** donné.

Le **type** d'une donnée indique :

- ☞ le domaine des valeurs qu'elle peut prendre ;
- ☞ les opérations permises sur cette donnée.

En effet, un programme manipule des données qui sont soit de type **numérique** (entier, réel, ...) soit de type **non numérique** (caractères ou chaînes de caractères).

Les types de base du langage Java :

type	équivalent en français	taille en mémoire	exemples de valeurs
char	caractère unicode	16 bits	'a' '\$' 'É'
byte	entier	8 bits	Domaine : [-128, +127]
short	entier court	16 bits	Domaine : [-32768, +32767]
int	entier	32 bits	Domaine : [-2 ³¹ , +2 ³¹ -1]
long	entier long	64 bits	Domaine : [-2 ⁶³ , +2 ⁶³ -1]
float	réel	32 bits	123.5f 25.f ou 25.0f -54.2f 12E+2f (1200)
double	réel double précision	64 bits	123.5 -54.26589 0.5 ou .5
boolean	booléen (oui, non)	8 bits	true false
String	chaîne de caractères		"Montréal, Qc" "9155 St-Hubert"
On notera que String commence par une majuscule car ne n'est pas un type de base (ou type <i>primitif</i>) du langage Java.			

Exercice :

Pour chacun des nombres suivants, indiquez si le nombre est un entier, un réel ou s'il est invalide pour le compilateur Java :

	nombre	entier	réel	invalide (pourquoi ?)
a	234			
b	123.			
c	2E+4			
d	-17.2			
e	0.0			
f	-.25			
g	12.5E+2			
h	-100			
i	-112.5			
j	12 E23			
k	12E-3			
l	12E -3			
m	- 100			
n	0.125E			

5. Déclarations

Toute donnée (*variable ou constante*) doit être **déclarée une et une seule fois**, avant d'être utilisée.
La déclaration permet au programme :

- ☞ de réserver l'espace mémoire ;
- ☞ d'initialiser la donnée.

Syntaxe :

Déclaration d'une variable :

type nom;

Déclaration et initialisation d'une variable :

type nom = valeur;

Déclaration et initialisation d'une constante symbolique :

final type nom = valeur;

Exemples de déclarations et d'initialisations valides en Java :

```
int i = 3;
char appreciation = 'A';
char delete = '\377';
long somme = 456L;
double val = -87.56;
float cumul = 76.3f;
double large;
double veryLarge = 657E+234;
int nombreEtudiants = 2,
    moyenneClasse;
boolean passe = true;
float moyExamens,
    moyLabs = 0.0f;
String nom,
    prenom = "Julie";
final double TAUX_TVQ = 0.075;
final char APOSTROPHE = '\'';
final char NEWLINE = '\n';
```

Exercice :

Pour chacune des déclarations suivantes, indiquez si elle est valide ou non pour le compilateur Java :

déclaration	valide	invalidé
<code>final double TVQ = 0.07;</code>		
<code>boolean b;</code>		
<code>double a2 = 5.2;</code>		
<code>int k = 5;</code>		
<code>double somme = 5;</code>		
<code>char c = '7';</code>		
<code>String a = "a";</code>		
<code>char c = "7";</code>		
<code>int f = 2.5;</code>		
<code>String vide = "";</code>		
<code>String lettres = 'abc';</code>		

6. La notion d'expression

La combinaison d'opérateurs et d'opérandes constitue **une expression** dont le résultat est une valeur.

- ☞ Un **opérateur** permet de faire un calcul.
- ☞ Un **opérande** peut être soit une constante, soit une variable.

Les différents opérateurs arithmétiques :

opérateur	exemple	remarques	valeur si $x = 7$ et $y = 2$
+	$x + y$	Addition	9
-	$x - y$	Soustraction	5
*	$x * y$	Multiplication	14
/	x / y	Division <u>Attention</u> : Si les opérandes sont entiers, il s'agit de la division entière	3
%	$x \% y$	Modulo (reste de la division)	1

La priorité des opérateurs :

Lors de l'évaluation d'une expression :

- 1 - On doit traiter les parenthèses en premier lieu
- 2 - La priorité des opérateurs intervient ensuite
- 3 - Pour les opérateurs de même priorité, l'évaluation se fait de gauche à droite

Table de priorité des opérateurs arithmétiques :

opérateurs		priorité
()	Parenthèses	1
+	Opérateur unaire d'addition	2
-	Opérateur unaire de soustraction	
*, /, %		3
+	Opérateur binaire d'addition	4
-	Opérateur binaire de soustraction	

Voir la table de priorité des opérateurs en annexe.

Exemple :

ordre : $2 * 2 + (2 + 1) * (2 + 1) / 3$
 3 6 1 4 2 5

Exercices :

Évaluez les expressions suivantes :

expression	résultat
$3.7 + (8 / 2) * (8 / 2)$	
$13 + 4 * 2 - 3 * (8 \% 3 + 5)$	
$-(5 * 2) * (5 * 2) * 10 + 18 / (4 + 8 / 2 / 2)$	
$12 + 3 * 4 / 3$	
$24 + 36 / (6 * 3 / (7 + 2))$	
$3 + 5 * 6 / 3 - 5$	
$1.0 / (1.0 / 2)$	
$1.0 / 1.0 / 2.0$	
$2 * 3.14 * \text{Math.pow}(4, 2)$	
$2 + \text{Math.sqrt}(8.0 / 2.0)$	

Expliquez la différence entre les 4 opérations suivantes :

opération	différence
$14 / 3$	
$14.0 / 3.0$	
$14.0 / 3$	
$14 \% 3$	

7. La notion d'affectation

L'**affectation** est une opération fondamentale dans les langages de programmation. C'est elle qui permet de placer une valeur en mémoire. Par contre, il faut faire attention au type des variables des chaque coté de l'assignation (un réel ne peut pas être conservé dans un entier).

Exemples :

notation conceptuelle	notation en Java	explications
$x \leftarrow 3$	<code>x = 3;</code>	Conserve la valeur 3 à l'emplacement désigné par x
$y \leftarrow 4$ $x \leftarrow y$	<code>y = 4;</code> <code>x = y;</code>	Conserve la valeur de la variable y (4) à l'emplacement désigné par x
$y \leftarrow 4$ $x \leftarrow y * 2$	<code>y = 4;</code> <code>x = y * 2;</code>	Conserve la valeur de l'expression y * 2 (8) à l'emplacement désigné par x

Exercices :

Indiquez quelles sont les valeurs des variables entières **i**, **j** et **k** après l'exécution de chacune des instructions suivantes : *(les variables conservent leur valeur d'une ligne à l'autre)*

instruction	valeur de i	valeur de j	valeur de k
<code>int i = 13;</code>			
<code>int j = 5;</code>			
<code>i = j + 1;</code>			
<code>i = i % 5;</code>			
<code>j = j * i + 2;</code>			
<code>int k = 1;</code>			
<code>k = (i * 7 + 3) / 5;</code>			

Sachant que **a = 4**, **b = 6**, **c = 2** et **d = 3**, calculez les expressions suivantes :

expression	résultat
<code>- a * ((2.0 + b) / c) * 2.5 * c</code>	
<code>(a * (b + d) - 2) * 3</code>	

Il est possible d'utiliser les opérateurs spéciaux pour simplifier l'écriture d'expressions courantes : des opérateurs d'affectation d'addition, de soustraction, de multiplication, de division et de modulo, ainsi que des opérateurs unaires de post-incrémentation et de post-décrémentation.

Exemples :

opérateur	exemple d'utilisation	remplace l'expression
<code>+=</code>	<code>somme += nombre;</code>	<code>somme = somme + nombre;</code>
<code>-=</code>	<code>total -= 2;</code>	<code>total = total - 2;</code>
<code>*=</code>	<code>resultat *= valeur;</code>	<code>resultat = resultat * valeur;</code>
<code>/=</code>	<code>nombre /= facteur;</code>	<code>nombre = nombre / facteur;</code>
<code>%=</code>	<code>nombre %= 4;</code>	<code>nombre = nombre % 4;</code>
<code>++</code>	<code>compteur++;</code>	<code>compteur = compteur + 1;</code> <i>ou</i> <code>compteur += 1;</code>
<code>--</code>	<code>nbFois--;</code>	<code>nbFois = nbFois - 1;</code> <i>ou</i> <code>nbFois -= 1;</code>
<p>Dans le cas des deux derniers opérateurs illustrés, leur exécution se produit une fois qu'on a terminé d'utiliser la variable dans l'expression courante.</p> <p>Ainsi, pour l'expression <code>total = nombre++ - 5</code>, la valeur de nombre sera augmentée de 1 seulement après que le résultat de l'expression <code>nombre - 5</code> ait été affecté à la variable <code>total</code>.</p>		

Exercices :

Indiquez quelles sont les valeurs des variables entières **a**, **b** et **c** après l'exécution de chacune des instructions suivantes : *(les variables conservent leur valeur d'une ligne à l'autre)*

	instruction	valeur de a	valeur de b	valeur de c
a	<code>a = 5;</code>			
b	<code>b = 3;</code>			
c	<code>c = a + b++;</code>			
d	<code>a--;</code>			
e	<code>c -= 10;</code>			
f	<code>a += b;</code>			
g	<code>a /= b--;</code>			
h	<code>a = b++ + c;</code>			
i	<code>b *= c--;</code>			
j	<code>a /= 2;</code>			
k	<code>b %= c;</code>			
l	<code>a = b = c;</code>			

Les entrées / sorties en Java

1. Affichage dans la console

Syntaxe :

```
// Affiche une ligne de texte
System.out.println("Mon message");

// Ne génère pas de saut de ligne après le message
System.out.print("Mon message");

// Affiche le message sur deux lignes
System.out.println("Bienvenue\n dans mon programme");

// Affiche le texte résultat = suivi de la valeur de la variable somme
System.out.println("résultat = " + somme);
```

Séquence de contrôle de Java :

\n	Nouvelle ligne
\t	Tabulation horizontale
\\	Barre oblique inverse (back slash)
\"	Guillemet

Exemples :

instructions Java	affichage à la console
System.out.println("Bonjour");	Bonjour
System.out.println("Bonjour Natalie");	Bonjour Natalie
System.out.println("Bonjour\nNatalie");	Bonjour Natalie
int moyenne = 80; System.out.println("Votre moyenne est " + moyenne);	Votre moyenne est 80
int moyenne = 80; System.out.println("Votre moyenne est\t" + moyenne);	Votre moyenne est 80

\n fait un retour à la ligne

\t fait une tabulation

Exercice :

Complétez le tableau suivant :

l = une ligne vide b = un espace
t = une tabulation

affichage	instructions
	<code>System.out.println("*****");</code> <code>System.out.println("***");</code>
	<code>System.out.print("*****");</code> <code>System.out.print("***");</code>
bbb127	
l l Bonjour l l	
* ***	<i>// en une seule instruction</i>
	<code>double moyenne = 67.4;</code> <code>System.out.println("La moyenne est\t" +</code> <code>moyenne + " %");</code>
	<code>char lettre = 'A';</code> <code>System.out.println("Le caractère " +</code> <code>lettre);</code>

2. Affichage dans une boîte de dialogue

Pour effectuer l'affichage dans une boîte de dialogue, nous pouvons utiliser la méthode graphique *showMessageDialog* de la librairie *javax.swing.JOptionPane* de java. Voir les exemples d'utilisation des boîtes de dialogue en annexe.

Syntaxe :

```
JOptionPane.showMessageDialog(null, "Mon message");
```

Exemple :

```
import javax.swing.*;
public class Exercice2 {

    public static void main(String[] args) {
        // déclaration des variables
        String texte = "tout le monde";

        // lecture des données

        // calcul


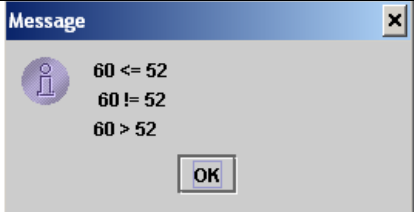
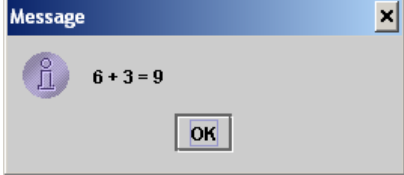
        // affichage des résultats
        JOptionPane.showMessageDialog(null, "Bonjour " + texte);

        // fin du programme
        System.exit(0); // nécessaire avec JOptionPane
    }
}
```



Exercice :

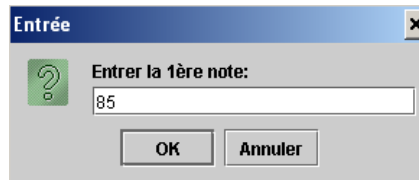
Complétez le tableau suivant :

affichage	instructions
	
	
	<pre>int nbr1 = 6; int nbr2 = 3; int somme;</pre>

3. Saisie des données à partir d'une boîte de dialogue

Pour effectuer la saisie d'une donnée, nous pouvons utiliser la méthode graphique `showInputDialog` de la librairie `javax.swing.JOptionPane` de java.

Cette méthode permet d'afficher une boîte de dialogue contenant un message d'invite et un champ (zone de texte) pour la saisie. Voir les exemples d'utilisation des boîtes de dialogue en annexe.



Notez que cette méthode retourne toujours une chaîne de caractères (type `String`) qui doit donc être convertie au besoin.

Exemple 1 saisie d'un nombre entier :

```
import javax.swing.*;
public class Exemple1 {

    public static void main(String[] args) {

        // déclaration des variables
        String saisie;           // variable de type chaîne de caractères
        int age;

        // lecture des données en entrée
        saisie = JOptionPane.showInputDialog("Entrez votre âge");
        age = Integer.parseInt(saisie);

        // traitement sur la donnée
        age = age + 10;

        // affichage du résultat dans la console
        System.out.println("Votre âge dans dix ans sera " +
                           age + " ans");

        // fin du programme
        System.exit(0);
    }
}
```

Exemple 2 saisie d'un nombre décimal :

```

import javax.swing.*;
public class Exemple2 {

    public static void main(String[] args) {

        // déclaration des variables
        String saisie;           // chaîne de caractères saisie par l'utilisateur
        double taille;

        // invite l'utilisateur à entrer sa taille en mètres
        saisie = JOptionPane.showInputDialog(
            "Entrez votre taille en m");
        taille = Double.parseDouble(saisie);

        // afficher le résultat dans une boîte de dialogue
        JOptionPane.showMessageDialog(null,
            "votre taille est " + taille + "m");

        // fin du programme
        System.exit(0);
    }
}

```



De façon analogue, on peut convertir la chaîne saisie dans les différents types offerts par Java :

type de la variable val	instruction pour convertir la chaîne saisie
byte	val = Byte.parseByte(saisie);
short	val = Short.parseShort(saisie);
int	val = Integer.parseInt(saisie);
long	val = Long.parseLong(saisie);
double	val = Double.parseDouble(saisie);
float	val = Float.parseFloat(saisie);
char	val = saisie.charAt(0);

Exercice :

Algorithme qui calcule la facture d'un client avec les taxes, puis affiche le coût avant taxes et le coût total. Les constantes TAUX_TPS et TAUX_TVQ contiennent les taux applicables.

```
lire prixUnit, qte
calculer coutAvantTaxes ← prixUnit * qte
calculer coutTotal ← coutAvantTaxes * (1 + TAUX_TPS) * (1 + TAUX_TVQ)
afficher coutAvantTaxes et coutTotal
```

Complétez le programme suivant en utilisant les boîtes de dialogue pour la saisie et l'affichage :

```
/*
 * Facture.java
 * Programme qui calcule la facture d'un client avec les taxes,
 * puis affiche le coût avant taxes et le coût total
 * Auteur   :
 * Date     :
 */

import javax.swing.*;
public class Facture
{
    public static void main (String args[])
    {
        final double TAUX_TPS = 0.05,    // les constantes pour les taux
                  TAUX_TVQ = 0.075;

        double prixUnit,                  // prix unitaire du produit acheté
               coutAvantTaxes,            // coût avant taxes
               coutTotal;                 // coût total avec taxes

        int qte;                          // quantité du produit acheté

        String saisie;                    // chaîne pour la lecture

        // lire prixUnit et qte

        // calculer coutAvantTaxes <- prixUnit * qte

        // calculer coutTotal <- coutAvantTaxes * (1 + TAUX_TPS) * (1 + TAUX_TVQ)

        // afficher coutAvantTaxes et coutTotal

        System.exit(0);
    }
}
```


4. Saisie des données à partir de la console

Les programmes qui affichent des résultats à la console lisent généralement les données à la console.

Exemple :

```
/*
 * Lecture.java (MODE CONSOLE)
 * Ce programme fait la lecture à la console du nom et de l'âge de l'utilisateur
 * pour ensuite afficher ces informations
 * Date : août 2007
 */

import java.io.*;
public class Lecture {

    public static void main (String args[]) throws IOException {

        int age;
        String nom;
        String saisie;

        BufferedReader clavier = new BufferedReader(
            new InputStreamReader(System.in));

        // lecture des données
        System.out.print("\nVotre nom : ");
        nom = clavier.readLine();

        System.out.print("\nVotre âge : ");
        saisie = clavier.readLine();
        age = Integer.parseInt(saisie);

        // affichage des résultats
        System.out.println("\nBonjour " + nom +
            ", vous êtes donc âgé(e) de " + age + " ans.");

        System.out.println("Puissiez-vous vivre encore longtemps!");
    }
}
```

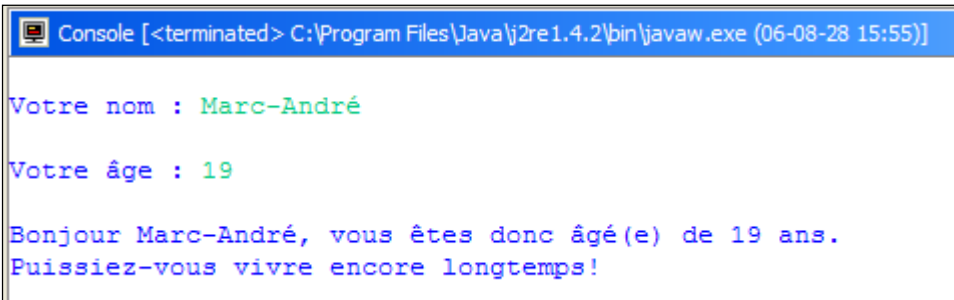
Pour importer la librairie qui contient les classes nécessaires à la lecture

Déclaration d'un tampon de lecture associé à la console

Lire la ligne de type String et la placer dans la variable nom

Extraire l'entier de la ligne lue et le placer dans la variable age

Résultats obtenus sur la console :



```
Console [<terminated> C:\Program Files\Java\j2re1.4.2\bin\javaw.exe (06-08-28 15:55)]

Votre nom : Marc-André

Votre âge : 19

Bonjour Marc-André, vous êtes donc âgé(e) de 19 ans.
Puissiez-vous vivre encore longtemps!
```