TP 1 : Débuter en Java

DUREE : 6 HEURES

Objectifs:

- S'initialiser avec la programmation en Java
- Découvrir les techniques de base de la programmation avec Java
- Installation de l'environnement de travail.
- Création du première applet Java et le premier programme Java.
- Familiarisation avec la syntaxe du langage Java.

Avant de programmer en Java

1. Le langage de programmation Java

Java est un environnement de programmation orienté objets adapté à la distribution d'applications sur Internet et s'intégrant au Web.

Il se compose de 4 éléments :

- un langage de programmation
- une machine virtuelle (JVM)
- un ensemble de classes standards réparties dans différentes API
- un ensemble d'outils (jdb, javadoc, ...)

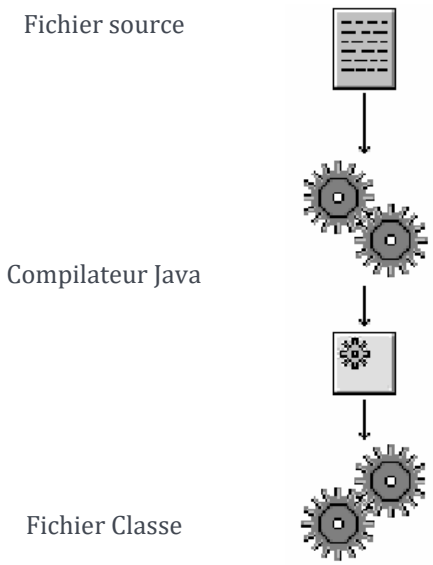
2. Interprétation d'un programme Java

Tout d'abord, Java simplifie le processus de développement; quelle que soit la machine sur laquelle le codage est réalisé, le compilateur fournit le même code.

Ensuite, quel que soit le système utilisé par les utilisateurs, cet unique code est directement opérationnel.

En effet, la compilation d'une source Java produit du pseudo-code Java qui sera exécuté par tout interpréteur Java sans aucune modification ou recompilation.

Cet "interpréteur" est la "machine virtuelle Java". De plus en plus, cette machine virtuelle utilise un compilateur JIT ("Just In Time") qui transforme, au moment de l'exécution, le pseudo-code en code natif (code machine pour un type d'ordinateur précis) afin d'obtenir la vitesse d'exécution maximale.



Le langage Java peut être utilisé pour créer des modules de code référencés au sein d'une page html et exécutés par un navigateur compatible Java(ou un simple interpréteur Java), on parle alors d'Applets.

Ces modules de code ont rendu Java populaire car ils permettent à un créateur de site d'enrichir le contenu de son site de modules dynamiques et/ou interactifs qui tourneront à l'identique quel que soit la machine et le système utilisé par le visiteur de ce site.

Java permet également de créer des applications autonomes qui peuvent se substituer à des applications développés en langage compilé. Pour ces applications, l'API Java apporte un ensemble très riche de classes répondant à de nombreux besoins et pouvant être étendues; cette unique API simplifie la création et le déploiement des applications, en effet cette application s'exécutera sur tout système en utilisant l'aspect visuel de ce système.

3. Les Types de programmes Java

Les programmes peuvent être des applications ou des applets :

- Une application est un programme exécuté localement.
- Une applet est une application exécutée par l'intermédiaire d'un navigateur Web qui définit automatiquement un cadre de travail. Ces cadres sont définies par le code HTML accompagnant l'applet.

Les différences entre une applet et une application sont :

- Les applets n'ont pas de bloc main() : la méthode main() est appelée par la machine virtuelle pour exécuter une application.
- Les applets ne peuvent pas être testées avec l'interpréteur mais doivent être intégrées à une page HTML, elle-même visualisée avec un navigateur sachant gérer les applets Java, ou testées avec l'applet viewer.

4. Les techniques de base de la programmation en Java

- N'importe quel éditeur de texte peut être utilisé pour éditer un fichier source Java ;
- Il est nécessaire de compiler la source pour le transformer en J-code ou byte-code Java qui sera lui exécuté par la machine virtuelle ;
- Il est préférable de définir une classe par fichier. Le nom de la classe publique et le fichier qui la contient doivent être identiques (il faut respecter la casse) ;
- Pour être compilé, le programme doit être enregistré au format de caractères Unicode : une conversion automatique est faite par le JDK si nécessaire.

4.1. Avant de programmer en Java

Le JDK de Sun (Java Development Kit) est l'outil essentiel pour programmer en Java. Il permet la compilation, le débogage et l'exception d'applications et d'applets Java. Il comprend également des fonctions avancées comme les signatures numériques, la création d'archives et de documentation ou l'intégration de code natif C.

La documentation de JDK est en fait l'API (ApplicationProgramming Interface)deJava (le détail de toutes les fonctions du langage).

Après avoir installé le JDK, il faut définir un chemin de recherche des exécutables « PATH».

Pour le faire, Lancer une session DOS et taper :

`Set Path=C:\JDK\bin` ou bien directement `Path=C:\JDK\bin` (Donne la valeur C:\JDK\bin à la variable d'environnement path)

Ou encore `Set Path=%Path%;C:\JDK\bin` (Ajoute le chemin C:\JDK\binà la valeur de la variable d'environnement path)

Ceci ne fonctionne que si C:\JDK\est le répertoire d'installation du JDK. Si ce n'est pas le cas, remplacer-le par votre vrai répertoire.

4.2. La compilation d'un code source

Pour compiler un fichier source il suffit d'invoquer la commande javac avec le nom du fichier source avec son extension .java : `javac NomFichier.java`

Le nom du fichier doit correspondre au nom de la classe principale en respectant la casse même si le système d'exploitation n'y est pas sensible.

```
Public class HelloWorld {
//code
}
```

Le nom du fichier sera HelloWorld.java

Une fois le fichier enregistré, ouvrir une fenêtre DOS et aller au répertoire où le fichier est enregistré (Rappel: utiliser cd nom répertoire pour monter d'un niveau et cd.. pour descendre).

Pour le compiler taper : `Javac HelloWorld.java`

Suite à la compilation, le pseudo-code Java est enregistré sous le nom HelloWorld.class, ce fichier est compilé et sera interprété par la machine virtuelle.

4.3. L'exécution d'un programme java

4.3.1. L'exécution d'une application

Une classe ne peut être exécutée que si elle contienne une méthode main() correctement définie. Pour exécuter un fichier contenant du bytecode, toujours dans le même répertoire (dans lequel

est créé le fichier .class), il suffit d'invoquer la commande java avec le nom du fichier source avec ou sans son extension .class : java NomFichier.

Dans notre exemple java HelloWorld

4.3.2. L'exécution d'une applet

Il suffit de créer une page HTML pouvant être très simple :

Exemple

```
<HTML>
<TITLE> test applet Java </TITLE>
<BODY>
<APPLET code=« NomFichier.class » width=270 height=200>
</APPLET>
</BODY>
</HTML>
```

Il faut ensuite visualiser la page créée dans l'applet viewer ou dans un navigateur32 bits compatible avec la version de Java dans laquelle l'applet est écrite.

4.3.3. Le premier Programme Java

Dans cette partie, on se propose de faire fonctionner un programme Java dont le rôle est d'afficher « Hello World ! ».

Il suffit de suivre les étapes suivantes :

1- Copier le programme suivant dans un éditeur de texte (exp NotePad)

```
public class HelloWorld {
public static void main(String[] args) {
System.out.println("Hello World!");
}
}
```

2- Enregistrer le fichier sous le répertoire à créer "c:\POO" sous le nom HelloWorld.java

3- Lancer une session Dos et effectuer les opérations équivalentes à celles décrites dans la fenêtre suivante :

4- Vérifier dans le même répertoire (c:\POO) la création du fichier HelloWorld.class (utiliser la commande Dir).

4.3.4. La première Applet Java

Soit le fichier suivant, à enregistrer sous le répertoire "c:\BEJava" avec le nom "HelloWorldApplet.java" :

```
import java.applet.*;
import java.awt.*;
public class HelloWorldApplet extends Applet {
public void paint(Graphics g) {
g.drawString("Hello world!", 50, 25);
}
}
```

Ensuite, copier le code HTML suivant dans un éditeur de texte et enregistrer-le sous le nom "Hello.HTML"

```
<HTML>
<HEAD>
<TITLE>Un Simple Programme</TITLE>
</HEAD>
<BODY>
<APPLET CODE="HelloWorldApplet.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

Compiler "HelloWorldApplet.java" pour créer "HelloWorldApplet.class" puis lancer AppletViewer Hello.html

On peut également Changer la fonte du texte à afficher. Le code de l'applet appropriée est :

```
import java.awt.Font;
import java.awt.Graphics;
public class HelloApplet extends java.applet.Applet
{ protected Font  font;
public void init(){
super.init();
font = newFont("TimesRoman", Font.BOLD, 24);
}
public void paint(Graphics graphics) {
graphics.setFont(font);
graphics.drawString("Hello world!", 10, 25);
}
}
```

Travail demandé

1ère Partie :

Installation de JDK et JRE.

2ème Partie :

Exercice 1 :

Exécuter l'applet « HelloWorld »

Exercice 2 :

Compiler les codes suivants puis corriger-les :

1. Déclaration de variables

```
public class test1 {
public static void main(String args[]) {
int i = 0;
for (int i = 0; i < 5; i++) { System.out.print(i + ", "); }
System.out.print("\n");
} }
```

2. Types de données : float et double

```
public class test2 {
public static void main(String args[]) {
float a = 3.0;
double b = 4;
float c;
c = Math.sqrt(a * a + b * b);
System.out.println("c = " + c);
} }
```

3. Types de données : byte et int

```
public class test3 {
public static void main(String args[]) {
byte b = 42;
char c = 'a';
short s = 1024;
```

```
int i = 50000;
float f = 5.67f;
double d = .1234;
double resultat = (f * b) + (i / c) - (d * s);
System.out.print((f * b) + " + " + (i / c) + " - " + (d * s));
System.out.println(" = " + resultat);
byte b2 = 10;
byte b3 = b2 * b;
System.out.println("b3 = " + b3);
}
}
```

4. Conversion de types

```
class test4 {
public static void main (String args[]) {
short x = 5 , y = 15;
x = x + y ;
System.out.println("x = " + x);
}
}
```

5. Passage de paramètres

```
public class test5 {
public static void main(String args[]) {
String jour_semaine[];
int jour = Integer.parseInt(args[0]);
jour_semaine = new String[7];
jour_semaine[0] = "dimanche";
jour_semaine[1] = "lundi";
jour_semaine[2] = "mardi";
jour_semaine[3] = "mercredi";
jour_semaine[4] = "jeudi";
jour_semaine[5] = "vendredi";
jour_semaine[6] = "Samedi";
System.out.println("Moi, je préfère le " +
jour_semaine[jour]);
}
}
```

Tester la commande java test5 7

Corrigez la classe test5 pour éviter ce genre de problème.

Exercice 3:

Ecrire un programme qui affiche le nom ainsi que le nombre de jours dans un mois donné, par exemple

Commande : java test6 10

Résultat : Le mois de novembre a 30 jours

Exercice 4 :

Ecrivez un programme Java qui lit un nombre et indique s'il est positif, négatif ou s'il vaut zéro et s'il est pair ou impair.

Exemple d'exécution:

```
Entrez un nombre entier: 5
Le nombre est positif et impair

Entrez un nombre entier: -4
Le nombre est négatif et pair

Entrez un nombre entier: 0
Le nombre est zéro (et il est pair)
```