

Laboratoire 1 – Introduction au Java
ITI 1521. Introduction à l'informatique II
25-29 Janvier 2021
Du en ligne après une semaine du Lab
/10

Objectifs

- Se familiariser avec le langage Java et les environnements de programmation
- Implémenter des programmes simples
- Conversion de type
- Boucles de contrôle

I. Introduction

Règles concernant l'écriture de programmes Java : Consultez ces règles pour tous vos labs java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html.

I.1. Éditeur de code source

Un éditeur de code source (éditeur de programme) est un éditeur de texte conçu pour éditer le code source des programmes. Ces logiciels ont généralement les caractéristiques suivantes :

- Facilité de la lecture du code et son débogage :
 - o Les éléments du langage tels que les mots réservés (clés), les paramètres et les chaînes de caractères sont affichés à l'aide de polices et de couleurs distinctes.
 - o Indentation du code source : l'éditeur de code source comprend un ensemble de règles afin de formater le texte. Par exemple, si l'on omet les parenthèses englobant les énoncés de la clause else, seul le premier énoncé sera indenté vers la droite.
 - o Correspondance des accolades ((),[]). Lorsqu'on positionne le curseur sur une accolade, l'éditeur met en évidence l'accolade correspondante.
 - o Auto-complétion des mots clés du langage.

Ci-dessous quelques éditeurs de code source populaires :

- Notepad++ (Windows, <http://notepad-plus-plus.org>)
- Sublime Text (Mac OS, Windows, Linux, <http://www.sublimetext.com>)
- TextWrangler (Mac OS, <http://www.barebones.com/products/textwrangler/>)
- TextMate (Mac OS, <http://macromates.com>)
- Emacs (Mac Os, Windows, Unix/Linux, <https://www.gnu.org/software/emacs/>)

L'éditeur de code source Notepad++ est recommandé.

Créer un simple programme « *HelloWorld* ». Un programme « *HelloWorld* » possède une méthode principale et celle-ci affiche la chaîne de caractères « *Hello World !* ». Vous nommerez cette classe *HelloWorld*.

- Copier le code source suivant dans votre éditeur de code source et sauvegarder le contenu dans un fichier sous le nom *HelloWorld.java*.

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

I.2. Compilation et exécution à partir d'une fenêtre de commandes

Ouvrez une fenêtre (shell) de commandes (command line interface). Assurez-vous que le répertoire courant soit celui qui contient le fichier *HelloWorld.java* (Pour le système d'exploitation Windows, utilisez la commande *cd* afin de vous déplacer et *dir* afin de faire défiler la liste des fichiers du répertoire courant).

Compilez le programme HelloWorld.java
javac HelloWorld.java

Lorsque vous aurez réussi à compiler le programme, vous verrez apparaître dans le répertoire le fichier *HelloWorld.class*. Le fichier *.class* contient la représentation en code-octet (byte-code) de votre programme.

Le code-octet est exécuté par un interprète qu'on nomme Java Virtual Machine (JVM). Afin d'exécuter ce programme, tapez ce qui suit dans la fenêtre de commandes.

java HelloWorld

*/*OUTPUT*/*

Hello World !

Le résultat de l'énoncé *println* apparaît sur la sortie (à l'écran). Ici, *java*, c'est la machine Java virtuelle (JVM). La JVM lit le code-octet de votre programme, une instruction à la fois, et exécute les actions nécessaires.

I.3. Environnement de programmation

Parmi les environnements de programmation les plus populaires dans l'industrie (<http://pypl.github.io/IDE.html>), on a **Eclipse** et **Netbeans**. Il est recommandé de vous familiariser avec ces environnements avant vos premières entrevues pour un placement coop ou un emploi. D'autres environnements plus simples, tels **DrJava** et **BlueJ** sont cependant préférables :

- La courbe d'apprentissage pour **Eclipse** et **Netbeans** est considérable.
- Ces environnements très puissants sont particulièrement intéressants pour développer de grandes applications avec de grandes équipes de travail.
- Ces environnements masquent les frontières entre les composantes (éditeur, compilateur, machine virtuelle, etc.)
- Ces environnements sont puissants, ils détectent et corrigent certains types d'erreurs.
- **DrJava** a une console d'interaction permettant l'exécution d'instructions Java sans avoir à créer de classes. (<http://www.drjava.org>)

Refaites maintenant l'exercice ci-dessus à l'aide de **DrJava**. Ouvrir le fichier *HelloWorld.java* à l'aide de DrJava, compiler ce programme, et exécuter le.

II. Langage Java

Les différences majeures entre le langage **Java** et **Python** sont:

- Java est un langage orienté objet. Tout code Java est inséré dans une classe :
 - o Tout programme doit posséder une méthode principale :
public static void main(String[] args).
 - o Java est un langage fortement typé. Il faut explicitement déclarer le type de chaque variable.
 - o Chaque énoncé est terminé par un point-virgule.
 - o Il faut mettre entre accolades les instructions d'un bloc de code.

Voici des ressources conçues pour faire la transition de Python à Java.

- From Python to Java by Ken Lambert
- Java for Python Programmers by Brad Miller
- Java Tutorials — Lesson: Language Basics

II.1. Paramètres sur la ligne de commandes

Il existe une façon simple de saisir des données en Java. On a utilisé la signature qui suit pour déclarer nos méthodes principales :

```
public static void main( String[] args );
```

Lorsque la machine virtuelle exécute un programme,

```
> java HelloWorld
```

Elle consulte la classe et cherche une méthode nommée *main* ayant la signature ci-haut.

La référence **args** est un paramètre de méthode et comme pour tous les paramètres vous pouvez choisir l'identificateur que vous souhaitez. Utiliser **args** parce que c'est d'usage courant sous Unix.

args est une référence vers un tableau de chaînes de caractères. Pour imprimer son contenu, vous allez créer un nouveau programme nommé *Cmdline* ayant une méthode principale. Vous comprendrez mieux ce qui se passe si chaque élément du tableau est imprimé sur sa propre ligne (imprimez les éléments un par ligne). Aussi, afin de bien délimiter le début et la fin de la liste d'arguments, affichez un message au début de la méthode et un message à la fin. Puisqu'on y est, vous devriez aussi imprimer l'index de l'élément.

En Java, l'opérateur «+» permet la concaténation de chaînes de caractères :

```
int i ;
```

```
i = 0 ;
```

```
System.out.println ( "Argument " + i + " est bonjour " );
```

La conversion de la valeur de l'entier en chaîne de caractères se fait automatiquement.

Voici des exemples de résultat :

```
>javac Cmdline.java
```

```
>java Cmdline Bienvenue en ITI1521!
```

```
Start of the program.
```

```
Argument 0 is Bienvenue
```

```
Argument 1 is en
```

```
Argument 2 is ITI1521!
```

```
End of the program.
```

Le système d'exploitation de l'ordinateur passe à notre programme un tableau de chaînes contenant tous les paramètres passés sur la ligne de commandes.

Question 1 : (1 POINT)

Implémentez le programme *Cmdline* décrit ci-dessus.

Question 2 : (1 POINT)

Écrire un programme nommé *Triangle* pour imprimer le triangle de chiffres binaires suivant :

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Question 3 : (1 POINT)

Écrire une méthode *factorial* qui prend un entier *n* en argument et qui calcule et renvoie la valeur de la factorielle de *n* ($n!$).

Écrivez un programme principal qui teste la méthode *factorial* en calculant et affichant la factorielle d'un nombre donné en argument de la ligne de commande.

Exemple de sortie :

```
> java factorial 5
120
```

II.2. Conversion de type

En Java, les chaînes de caractères sont représentées par des objets de la classe `String`, dont les instances sont non mutables. On souhaite pouvoir convertir des chaînes de caractères contenant des nombres en valeur entières. Le faire en utilisant la méthode statique `static int parseInt(String)` de la classe `Integer`.

Ces classes regroupent aussi un ensemble de méthodes utiles pour ce type. Consulter les suivants pour en savoir plus:

- <http://docs.oracle.com/javase/8/docs/api/overview-summary.html>

C'est la documentation des classes et méthode de Java 8.0 ;

Visitez le package `lang` (qui est toujours chargé par défaut dans vos programmes) ;

- <http://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html>

Plus bas dans cette page, vous trouverez la classe *Integer* ;

- <http://docs.oracle.com/javase/8/docs/api/java/lang/Integer.html>

Trouvez maintenant la méthode *parseInt(String s)* ;

Question 4 : (1 POINT)

Construisez la classe *Somme* qui fait la conversion de tous les éléments sur la ligne de commande, de chaîne en entiers, et fait aussi la somme des éléments.

Exemple de sortie :

```
> javac Somme.java
> java Somme 12 13 14 15
La somme est 54
```

II.3. Boucles de contrôle :

Question 5 : (1 POINT)

Implémentez un programme nommé *ForTest* qui calcule la somme des 10 premiers entiers en utilisant la boucle *for*.

Question 6 : (1 POINT)

Implémentez un programme nommé *WhileTest* pour trouver la somme des chiffres qui constituent le nombre entier 123489. Utiliser la boucle *while*.

Question 7 : (1 POINT)

Implémentez un programme nommé *DoWhileTest* pour trouver la somme des chiffres qui constituent le nombre entier 123489. Utiliser la boucle *do while*.

II.4. Utilisation de la documentation en ligne :

Question 8 : (3 POINTS)

Écrire un programme *Dates* pour obtenir et afficher la date courante selon 2 façons différentes:

- en utilisant la classe *Date* (créer un objet de type *java.util.Date*) et afficher le résultat (date et heure) par une expression comme :
Today is Tue Jan 12 19:10:48 EST 2021
- en construisant un objet de type *java.util.Date* comme au premier cas, et un objet de type *java.text.SimpleDateFormat* qui est une variété de *java.text.DateFormat*. Afficher le résultat (date et heure) par une expression comme :
On est le 26 janvier 2021 21:10

Créer et soumettre un fichier zip

Directives

- Créez un répertoire que vous nommerez *lab1_ID*, où vous remplacerez ID par votre numéro d'étudiant.
- Dans ce répertoire, créez 8 sous-répertoires **Q1, Q2, ... et Q8**.
- Dans chaque répertoire, copier les programmes correspondant à cette question. N'y ajoutez que le code source, les fichiers *.java*. Ne soumettez pas les fichiers *.class*.
- Dans le répertoire *lab1_ID*, créez un fichier texte nommé *README.txt*, qui devra contenir votre nom, numéro d'étudiant, ainsi qu'une brève description du contenu :

Nom étudiant :

Numéro d'étudiant :

Code du cours : ITI1521

Section Lab:

Resources

- <https://docs.oracle.com/javase/tutorial/getStarted/application/index.html>
- <https://docs.oracle.com/javase/tutorial/getStarted/cupojava/win32.html>
- <https://docs.oracle.com/javase/tutorial/getStarted/cupojava/unix.html>
- <https://docs.oracle.com/javase/tutorial/getStarted/problems/index.html>
- <https://docs.oracle.com/javase/15/docs/api/java/sql/Date.html>
- <http://www.oracle.com/technetwork/java/javase>
- <http://openjdk.java.net/projects/jdk/>

L'environnement de travail dans lequel nous allons travailler est le suivant :

- Java SE 15 de Oracle
- Editeur Notepad++ (Windows, <http://notepad-plus-plus.org>) ou votre éditeur préféré
- Une fenêtre de type *xterm* pour le travail en dehors de l'éditeur, pour lancer les commandes `javac`, `java`,... (une fenêtre de commande "cmd" sous Windows).
- Dans le navigateur Web que vous préférez ajoutez des marque-pages (ou signets) sur les adresses suivantes pour une aide en ligne indispensable :
 - documentation sur les API (signet vers <http://download.oracle.com/javase/15/docs/api/index.html>)
 - documentation sur les outils Java fournis avec le JDK (signet vers <http://download.oracle.com/javase/15/docs/technotes/tools/index.html#basic>)

Sur votre ordinateur personnel

Pour développer en Java, il faut un JDK.

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

il vous faudra aussi installer la *javadoc* sur votre machine (le lien "Java SE 15 documentation" plus bas sur la même page de chargement du JDK).

- **Drjava**
<http://www.drjava.org>