

Cours/tp n° : 1 Prise en main**Avant-propos :**

Ce premier tp inaugure une série de plusieurs, sur le langage Java. A travers ces Tp vous découvrirez progressivement les bases de Java. Vous ne connaîtrez pas tout sur ce langage(ce n'est pas le but de ces tp), mais vous aurez les briques essentielles et nécessaires à la construction de votre savoir.

L'objectif de ces tp est de vous faire programmer les notions vues en cours d'algorithmie. Le langage de programmation est en fait accessoire, nous aurions pu utiliser le Pascal, le C, le C++...Le choix d'utiliser Java, m'a été dicté par le fait de son orientation réseau et web, ce qui risque d'être votre cœur de métier plus tard. Ce n'est pas le meilleur langage en mode console (mode non fenêtré), mais nous allons l'utiliser ainsi au début pour apprendre les opérations algorithmiques de bases.

Les Tp feront en sorte d'apporter progressivement les éléments nouveaux et éviteront les écueils qui risquent de vous déstabiliser au début. Certains points seront utilisés mais non explicités au départ, c'est volontaire afin de ne pas vous « embrouiller » dès le départ. Ces points là seront approfondis plus tard dans votre progression

Vous êtes maître de votre apprentissage, donc le but du jeu n'est pas de faire le tp le plus rapidement possible, mais de **comprendre et d'assimiler** l'objectif du tp. Donc n'hésitez pas à refaire plusieurs fois le Tp, jusqu'à sa compréhension totale (elle arrivera à un moment donné). Ne vous démoralisez pas si des collègues finissent avant vous un tp, votre tour viendra de les pourrir, mais plus tard ! Donc apprenez la patience et l'humilité, en informatique rien n'est acquis d'avance et vous trouverez toujours plus fort que vous...En temps que maître de votre apprentissage rien ne vous empêche de vous perfectionner en Java de façon autonome en cherchant sur le net d'autres supports plus ciblés et d'en faire profiter vos collègues.

Ces mises en garde étant faites, je vous souhaite beaucoup de plaisir avec Java et bon courage !

Mise en œuvre de java

Le « moteur » de Java est normalement installé sur votre ordinateur, c'est le JDK(Java Development KIT) Sun, sinon vous le trouverez sur <http://java.sun.com>.

Nous allons utiliser un éditeur de texte spécialisé java nommé « Jcreator LE », il a été installé sur votre ordinateur ou vous pouvez le télécharger sur <http://www.jcreator.com>.

Caractéristiques de Java.

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C. Il existe 2 types de programmes en Java : les applets et les applications. Une application autonome (stand alone program) est une application qui s'exécute sous le contrôle direct du système d'exploitation. Une applet est une application qui est chargée par un navigateur Web et qui est exécutée sous le contrôle de celui ci.

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- ✓ Java est interprété : Le source est compilé en pseudo code ou byte-code puis exécuté par un interpréteur Java : La Java Virtual Machine (JVM). Ce concept est à la base du slogan de Sun pour Java :WORA (Write Once, Run Anywhere : écrire une fois, exécuté partout). En effet, le byte-code, s'il ne contient pas de code spécifique à une plate-forme particulière peut être exécuté et obtenir les mêmes résultats sur toutes les machines disposant d'une JVM.
- ✓ Java est indépendant de toute plate-forme :Il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du byte code.
- ✓ Java est orienté objet : Comme la plupart des langages récents, java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java est complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).
- ✓ Java est simple : le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), 'héritage multiple et la surcharge des opérateurs, ...
- ✓ Java est fortement typé : Toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utilisé un cast ou une méthode statique pour la réaliser.

- ✓ Java assure la gestion de la mémoire : l'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector qui restitue les zones de mémoire laissées libres suite à la destruction des objets.
- ✓ Java est sûr : La sécurité fait partie intégrante du système d'exécution et du compilateur. Un programme Java planté ne menace pas le système d'exploitation. Il ne peut pas y avoir d'accès direct à la mémoire. L'accès au disque dur est réglementé dans une applet. Les applets fonctionnant sur le Web sont soumises aux restrictions suivantes dans la version 1.0 de Java : aucun programme ne peut ouvrir, lire, écrire ou effacer un fichier sur le système de l'utilisateur
 - aucun programme ne peut lancer un autre programme sur le système de l'utilisateur
 - toute fenêtre créée par le programme est clairement identifiée comme étant une fenêtre Java, ce qui interdit par exemple la création d'une fausse fenêtre demandant un mot de passe
 - les programmes ne peuvent pas se connecter à d'autres sites Web que celui dont ils proviennent.
 - Java est économe : le pseudo code a une taille relativement petite car les bibliothèques de classes requises ne sont liées qu'à l'exécution.
- ✓ Java est multitâche : il permet l'utilisation de threads qui sont des unités d'exécution isolées. La JVM elle-même utilise plusieurs threads.

Les programmes Java exécutés localement sont des applications, ceux qui tournent sur des pages Web sont des applets. 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.

Il ne faut pas confondre Java et JavaScript. JavaScript est un langage développé par Netscape Communications.

La syntaxe des deux langages est proche car elles dérivent toutes les deux du C++.

Il existe de nombreuses différences entre les deux langages :

	Java	Javascript
Auteur	Développé par Sun Microsystems	Développé par Netscape Communications
Format	Compilé sous forme de byte-code	Interprété
Stockage	Applet téléchargée comme un élément de la page web	Source inséré dans la page web
Utilisation	Utilisable pour développer tous les types d'applications	Utilisable uniquement pour "dynamiser" les pages web
Execution	Exécuté dans la JVM du navigateur	Exécuté par le navigateur
POO	Orienté objets	Manipule des objets mais ne permet pas d'en définir
Typage	Fortement typé	Pas de contrôle de type
Complexité du code	Code relativement complexe	Code simple

Remarques : Dans l'éditeur vous écrirez votre programme qui portera une fois sauvegardé l'extension *.java (monprog.java par exemple). Une fois qu'il sera compilé il s'appellera *.class(monprog.class par exemple). Apportez donc une attention particulière à l'extension et à la casse (Java est sensible aux différences Majuscule/Minuscule), ainsi MonProg.java est différent de monprog.java.

Les types élémentaires

Les types élémentaires ont une taille identique quelque soit la plate-forme d'exécution : c'est un des éléments qui permet à java d'être indépendant de la plate-forme sur lequel le code s'exécute.

Type	Désignation	Longueur	Valeurs	Commentaires
boolean	valeur logique : true ou false	8 bits	True ou false	pas de conversion possible vers un autre type
byte	octet signé	8 bits	-128 à 127	
short	entier court signé	16 bits	-32768 à 32767	
char	caractère Unicode	16 bits	\u0000 à \uFFFF	entouré de cotes simples dans un programme Java
int	entier signé	32 bits	-2147483648 à 2147483647	
float	virgule flottante simple précision	32 bits	1.401e-045 à 3.40282e+038	
double	virgule flottante double précision	64 bits	2.22507e-308 à 1.79769e+308	
long	entier long	64 bits	-9223372036854775808 à 9223372036854775807	

L'affectation

le signe = est l'opérateur d'affectation et s'utilise avec une expression de la forme variable = expression. L'opération d'affectation est associative de droite à gauche : il renvoie la valeur affectée ce qui permet d'écrire :

x = y = z = 0;

Il existe des opérateurs qui permettent de simplifier l'écriture d'une opération d'affectation associée à un opérateur mathématique :

Opérateur	Exemple	Signification
=	a=10	équivalent à : a = 10
+=	a+=10	équivalent à : a = a + 10
-=	a-=	équivalent à : a = a - 10
=	a=	équivalent à : a = a * 10
/=	a/=10	équivalent à : a = a / 10
%=	a%=10	reste de la division
^=	a^=10	équivalent à : a = a ^ 10
<<=	a<<=10	équivalent à : a = a << 10 a est complété par des zéros à droite
>>=	a>>=10	équivalent à : a = a >> 10 a est complété par des zéros à gauche
>>>=	a>>>=10	équivalent à : a = a >>> 10 décalage à gauche non signé

Les comparaisons

Java propose des opérateurs pour toutes les comparaisons :

Opérateurs	Exemple	Signification
>	a > 10	strictement supérieur
<	a < 10	strictement inférieur
>=	a >= 10	supérieur ou égal
<=	a <= 10	inférieur ou égal
==	a == 10	Egalité
!=	a != 10	diffèrent de
&	a & b	ET binaire
^	a ^ b	OU exclusif binaire
	a b	OU binaire
&&	a && b	ET logique (pour expressions booléennes) : l'évaluation de l'expression cesse dès qu'elle devient fausse
	a b	OU logique (pour expressions booléennes) : l'évaluation de l'expression cesse dès qu'elle devient vraie
? :	a ? b : c	opérateur conditionnel : renvoie la valeur b ou c selon l'évaluation de l'expression a (si a alors b sinon c) : b et c doivent retourner le même type

1^{ère} classe

Lancez Jcreator, dans la barre de menu cliquez sur **File** puis sur **New**. Dans la nouvelle fenêtre, cliquez sur l'onglet **Files** et sur l'objet **Java File**. Dans la zone **filename** donnez comme nom de fichier Java le nom de la classe que nous allons créer c'est à dire **bonjour**. La zone location vous permet de définir le répertoire de sauvegarde de vos fichiers.



Remarque : Il vous faut enregistrer le fichier sous le même nom que la classe principale sinon vous aurez une erreur de compilation.

Voici enfin notre premier programme. Il commence par écrire une chaîne de caractères, puis il écrit la date en utilisant la classe **Date** de la bibliothèque standard de Java. Il faut remarquer qu'un style de commentaires supplémentaires est introduit ici : le `/**` qui est un commentaire jusqu'à la fin de la ligne.

Fichier « bonjour.java »
<pre>// bonjour.java import java.util.*; public class bonjour { public static void main(String[] args) { System.out.print("Bonjour, nous sommes le : "); System.out.println(new Date()); } }</pre>

Saisissez-le tel quel et enregistrez le sous **bonjour**.

Voici les explications du code que vous venez de saisir.

```
import java.util.*;
```

Au début de chaque fichier de programme il faut mettre la déclaration **import** pour importer toutes les classes supplémentaires qui seront nécessaires pour le code dans ce fichier. Il faut noter que je dis « supplémentaires » ; C'est parce qu'il y a une bibliothèque de classes qui est automatiquement importée dans tous les fichiers Java : **java.lang**. Étant donné que **java.lang** est implicitement inclus dans tous les fichiers source Java, ces classes sont automatiquement disponibles. C'est le cas de la classe **System** (**System.out.print(...)**). Il n'y a pas de classe **Date** dans **java.lang**, il faut donc déclarer **import java.util.*** de façon à pouvoir utiliser **Date**.

```
public class bonjour
```

Le nom de la classe est le même que celui du fichier. Quand on crée un programme autonome comme celui-là une des classes du fichier doit avoir le même nom que le fichier (le compilateur se plaint si on ne le fait pas). Cette classe doit contenir une méthode appelée **main()** avec la signature suivante :

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

Cette méthode est la méthode principale (**main** in english) de votre programme **bonjour**, c'est ici que vous placerez les actions à faire exécuter à Java. Remarquez comment les accolades marquent bien les blocs de programme importants. Le mot-clef **public** signifie que la méthode est accessible au monde extérieur. Le paramètre de **main()** est un tableau d'objets de type **String**. Le paramètre **args** n'est pas utilisé dans ce programme mais le compilateur Java insiste pour qu'il soit là car il contient les paramètres invoqués sur la ligne de commande.

```
System.out.print("Bonjour, nous sommes le : ");
```

Pour l'instant intéressons-nous à **print()**, qui veut dire « écris ce que je te donne sur la console et ne passe pas à la ligne ». Ainsi, dans tout programme Java on peut dire **System.out.print("quelque chose")** chaque fois qu'on souhaite écrire quelque chose sur la console. **Println()** lui veut dire « écris ce que je te donne sur la console et passe à la ligne ».

```
System.out.println(new Date());
```

La ligne qui écrit la date est assez intéressante : Considérons le paramètre : un objet **Date** est créé (**new Date()**) juste pour transmettre sa valeur à **println()**. Dès que cette instruction est terminée, cette date est inutile et le ramasse-miettes (Garbage collector) peut venir le récupérer n'importe quand. On n'a pas à s'occuper de s'en débarrasser.

Le code étant expliqué, nous allons l'exécuter. Sous Jcreator, cliquez sur le menu **Build** puis sur **Compile File** (Cette phase de compilation créera le fichier *.class qui contient le Byte-code). Si tout se passe bien dans la fenêtre output (la zone inférieure de Jcreator le message suivant apparaîtra :

```
Process completed.
```

Si vous avez un message autre c'est que vous avez fait une erreur de syntaxe, relisez votre code en le comparant avec celui du TP.

Une fois le code compilé cliquez sur le menu **Build** puis sur **execute file**, une console DOS va s'ouvrir et votre programme va s'exécuter sous vos yeux ébahis. Ca y est, vous devenez un programmeur de talent, elle n'est pas belle la vie ?

Il est évident que ce premier programme est vraiment simpliste, mais son but est de vous montrer le fonctionnement de base d'un programme Java et sa structuration spécifique. N'oubliez pas d'indenter (faire des retraits) vos sources, vous y gagnerez en clarté, lisibilité, vous maintiendrez plus facilement vos programmes.

Exercices applicatifs

Le but de ces exercices est de vous familiariser avec le langage, vous travaillez en autonomie, à votre rythme. Il n'y a aucune surprise vous avez vu dans la leçon tous les éléments nécessaires pour résoudre les exercices demandés. Donc courage... !

Exo 1 :

Créez une classe nommée `etatscivil` qui affiche sur la première ligne votre nom votre prénom, sur la deuxième votre date de naissance, sur la troisième votre ville de naissance.

Exo 2 :

En vous aidant du code de la classe `etatscivil` (copier / coller), créez une classe nommée `etatscivil2`. Modifiez le code pour que la ville de naissance soit sur la même ligne que la date. Attention votre code doit comporter le même nombre de ligne que la classe de référence (`etatscivil`).