Chapitre I
Notions de base et outils de travail

Objectifs

- Connaître les principes fondateurs et l'historique du langage Java
- S'informer des principales caractéristiques du langage Java
- Connaître l'environnement de développement de Java

1. LANGAGE JAVA EN BREF

Le langage de programmation Java peut être considéré comme l'une des avancées les plus significatives du monde informatique durant les 20 dernières années. Selon plusieurs études statistiques, il est depuis longtemps le langage le plus utilisé et choisi par la communauté des programmeurs. Sa popularité vient de plusieurs facteurs. Java est d'abord et avant tout le premier langage de programmation à pouvoir s'exécuter sur presque n'importe quel type de plateforme, c'est-à-dire sur n'importe quel type d'ordinateur (ou plus précisément de processeur, Motorola, Intel, etc.) et n'importe quel système d'exploitation (Linux, Unix, Windows, etc.). Il a été le premier langage de programmation à offrir la possibilité d'utiliser des applets qui représentent des programmes pouvant être exécutés de manière sécurisée et dynamique dans un environnement distribué et multi-plateforme. D'autre part, java est un langage de programmation orienté-objet avec une syntaxe simple et familière. Il est également fourni avec un ensemble très riche de plusieurs librairies (de mathématique, de développement graphiques, de traitement de fichiers, d'accès aux bases de données, de réseau, d'utilitaires, etc., etc.) libérant ainsi le programmeur à re-écrire des fonctions communes et usuelles et lui permettant de se concentrer sur la résolution de son problème spécifique. Par ailleurs, avec sa caractéristique de portabilité, Java est ouvert sur le monde de l'Internet, ce qui en fait un des langages les plus appropriés pour le développement des applications web. Actuellement, et toujours selon les statistiques, l'architecture JEE (Java Entreprise Edition) basée essentiellement sur le langage Java, reste l'une des plus populaires auprès des compagnies informatiques pour le développement des applications web.

2. HISTORIQUE DE JAVA

Le langage de programmation Java est né en 1991. Une équipe de recherche, nommée "Green team" de la société Sun Microsystems, travaillait dans le cadre d'un projet appelé "Oak" (en référence, selon la légende, à un arbre planté devant la fenêtre de leur bureau) dans le but était d'établir un pont de communication entre les nombreux périphériques de consommateurs. Cette équipe désirait créer un nouveau langage de programmation qui permettrait aux périphériques grand public équipés de différents processeurs de partager les mêmes avancées logicielles.

Ce projet a voué à l'échec et l'équipe se devait de trouver un autre débouché pour son nouveau langage de programmation. Durant la même période, l'Internet prenait son essor et l'équipe s'était aperçue que le langage Oak était parfaitement adapté au développement de composants multimédia pour Internet et à l'amélioration des pages web. Ces petites applications, appelées applets, ont constitué la première utilisation du langage Oak et les programmeurs ont alors adopté ce qui est devenu le langage de programmation Java.

En 2009, la société Oracle a acquis l'entreprise Sun Microsystems. On peut désormais voir apparaître le logo Oracle dans les documentations du langage Java.

3. CARACTERISTIQUES DU LANGAGE JAVA

Les principales caractéristiques du langage Java sont les suivantes :

Portabilité : Dans le monde informatique il existe plusieurs systèmes d'exploitation et plusieurs types d'ordinateurs (ou de processeurs). La combinaison "processeur/système d'exploitation" est généralement appelée une "plateforme". Les informaticiens ont toujours cherché à développer des programmes qui peuvent s'exécuter sur n'importe quelle plateforme, autrement dit développer des programmes qui soient portables, c'est-à-dire pouvoir porter ces programmes d'une plateforme à une autre sans aucun travail supplémentaire. Cependant, tous les langages de programmation qui étaient offerts sur le marché informatique n'assuraient pas cette portabilité. Prenons par exemple le cas d'un programme écrit avec le langage VB (Visual Basic), ce dernier ne peut s'exécuter ni sur Linux, ni sur MacOS mais seulement sur une plateforme dont le système d'exploitation est Windows. Le langage C ou C++ oblige le programmeur à recompiler un programme écrit sur une certaine plateforme pour le porter sur une autre. Rappelons qu'un compilateur est une application qui convertit le code du programme (écrit en C, VB ou autre) en code binaire (ou code machine). Un fichier binaire est spécifique (dépendant) à la plateforme sur laquelle il a été créé et est appelé un exécutable, pouvant être exécuté par un utilisateur final.

Un programme écrit en langage Java peut s'exécuter sur n'importe quelle plateforme, par exemple sous le système d'exploitation Solaris avec un processeur SPARC, sous MacOS avec un processeur Motorola et sous Windows avec un processeur Intel, sans aucune modification. Les programmes Java sont également compilés à l'aide d'un compilateur de la technologie Java. Cependant, le format résultant d'un programme Java après compilation est du **bytecode** indépendant de la plateforme et non du code binaire spécifique. Une fois créé, le bytecode est interprété par un interpréteur de bytecode appelé **machine virtuelle Java (JVM^{TM})**. Cette dernière correspond à un programme qui reconnaît le bytecode indépendant de la plateforme et peut l'exécuter sur n'importe quelle plateforme.

Orienté Objet : Dans les années durant lesquelles l'équipe "Green Team" avait commencé le développement de Java, la programmation procédurale (ou modulaire) avait déjà montré ses limitations et faiblesses au profit de la programmation Orienté objet. Il est évident donc que le langage Java se devait d'être également orienté objet. La programmation modulaire avait comme principe de regrouper les données dans une même entité et de les séparer des traitements. L'idée principale de l'évolution vers

la programmation orientée-objet est de continuer à séparer les données et les traitements. Cependant, les traitements directs qui s'opèrent sur une certaine structure de données devraient être directement liés à cette structure. Ainsi, l'idée est de considérer comme faisant partie d'une même structure, appelée Classe, non seulement les données mais également les fonctions qui s'y rattachent. Le code devient logiquement découpé en petites entités cohérentes et devient ainsi plus simple à maintenir et plus facilement réutilisable, étant intrinsèquement modulaire.

Distribution : Java est un langage distribué car il fournit la prise en charge des technologies réseau distribuées, telles que RMI, CORBA ou URL. De plus, la technologie java permet de télécharger du code sur Internet et de l'exécuter sur un ordinateur personnel. En effet, La plate-forme Java fut l'un des premiers systèmes à offrir le support de l'exécution du code à partir de sources distantes. Ceci est possible par l'utilisation d'applets. Une applet peut fonctionner dans le navigateur web d'un utilisateur, exécutant du code téléchargé d'un serveur.

Simplicité : Depuis le début, l'équipe "Green Team" s'était fixée comme objectif de proposer un langage simple, principalement en termes de syntaxe. Cette simplicité revenait principalement à éviter certaines complexités des langages C et C++ étant donné que ces derniers étaient très utilisés avant l'apparition de Java. A titre d'exemple, et contrairement à ces deux langages, Java utilise une syntaxe très simple dans la gestion et la manipulation des pointeurs. Le deuxième exemple de simplicité concerne la gestion de la mémoire. Java, contrairement à ces deux langages, a proposé un nouveau concept appelé "ramasse miettes" qui soulage le programmeur à surveiller et à avoir à supprimer lui-même les objets qui ne sont plus référencés. D'autres exemples de simplicité seront présentés dans la suite de l'ouvrage.

Multitâche : Java assure le multitâche en donnant la capacité à un programmeur de faire exécuter simultanément plusieurs tâches. Une tâche, appelée thread dans la terminologie java, est une partie de code représentant un flot d'instructions s'exécutant en concurrence avec d'autres flots d'instructions. Les avantages principaux du multitâche sont des performances plus élevées en matière d'interactivité et un meilleur comportement en temps réel, bien que ce dernier soit en général dépendant du système. Notons que Java intègre certains threads de manière transparente (ramasse miettes, horloge, chargement des images ou des sons), mais permet également au programmeur de développer ses propres threads de façon simple.

Sécurité : Les programmes écrits avec le langage Java sont sécurisés car ce dernier utilise des mesures de sécurité telles que l'interdiction de manipuler la mémoire à l'aide de pointeurs, l'interdiction pour les programmes distribués de lire et d'écrire sur le disque dur d'un ordinateur ou encore la vérification de la validité du code de tous les programmes Java.

4. LICENCE ET EDITIONS DU LANGAGE JAVA

Le langage Java est libre et open source (depuis novembre 2006) et sa licence à permis l'éclosion d'un grand nombre d'outils libres dans les domaines les plus variés. Il se décline sous plusieurs éditions vu sa popularité et sa portabilité sur divers types de plateformes :

- Java SE (Standard Edition) : Cette édition est destinée aux applications pour poste de travail. C'est l'édition utilisée dans le cadre de ce document.
- Java EE (Entreprise Edition) : Cette édition est spécialisée dans les applications d'entreprise. Elle contient pour cela un grand nombre d'API et d'extensions.
- Java ME (Mobile Edition) : Cette édition concerne les applications mobiles.

5. INSTALLATION DE JAVA

Pour pouvoir utiliser le langage Java, il faut télécharger l'édition standard (ou poste de travail) à partir de l'adresse (Noter que l'adresse web peut avoir changé depuis la publication de ce document) :

http://www.oracle.com/technetwork/java/javase/downloads/

L'édition standard propose deux possibilités de téléchargement :

- Télécharger la JRE (Java Runtime Environment – Environnement d'exécution Java): Il s'agit de l'environnement minimal pour permettre à des programmes compilés en Java de s'exécuter sur votre ordinateur. Cet environnement ne comprend donc que la machine virtuelle Java JVM et permet d'exécuter des fichiers bytecode.

- Télécharger la JDK (Java Development Kit – Kit de développement Java): Il s'agit d'un kit qui comprend non seulement la machine virtuelle Java JVM mais également un environnement de développement simple, c'est-à-dire un environnement qui permet de développer des programmes simples en fournissant des commandes de compilation, d'exécution, de génération automatique de la documentation, de génération automatique d'archives, etc. C'est cette version qui est utilisée dans le cadre de cet ouvrage.

Une fois le téléchargement terminé, il suffit de lancer le programme d'installation et de suivre les instructions.

6. AUTRES RESSOURCES

Plusieurs autres ressources existent sur le langage Java. Plus particulièrement, les lecteurs sont invités à se documenter auprès des références officielles suivantes :

- Reportages sur la technologie Java, historique de la technologie Java et de ses auteurs - http://www.oracle.com/technetwork/java/javase/overview/
- Didacticiel Java destiné aux programmeurs avec plusieurs exemples - http://docs.oracle.com/javase/tutorial/
- Un livre destiné aux étudiants et comprenant de nombreux exemples de code et exercices – Deitel and Deitel, *Java: how to program*, Prentice-Hall, 2004
- Un livre destiné aux non programmeurs – Farrell, Joyce, *Java Programming : Comprehensive*, Course Technology, 1999

Chapitre II
Premières applications en Java

Objectifs

- Etre capable d'écrire des programmes simples en Java
- Etre capable d'utiliser les instructions d'entrées/sorties
- Devenir familier avec les types de données primitifs
- Comprendre les concepts de base de la mémoire
- Etre capable d'utiliser les opérateurs arithmétiques, relationnels et logiques
- Comprendre la précédence des opérateurs
- Utiliser la documentation

Nous introduisons dans la première partie de ce chapitre la programmation avec le langage Java en présentant des exemples simples qui illustrent plusieurs caractéristiques du langage. Chaque exemple est analysé ligne par ligne.

Un programme utilise en général des données en entrée et des données en sortie. Par exemple, un programme peut consister à lire en entrée une valeur en mètres et le programme doit la convertir en sortie en valeur en pieds. Chaque donnée manipulée dans un programme doit appartenir à un type donné (entier, réel, etc.).

Dans la deuxième partie de ce chapitre, nous allons étudier les types fournis en Java et détaillons les opérateurs qui permettent de combiner plusieurs données (chacun appartenant à un type) dans une expression.

1. PREMIER PROGRAMME: AFFICHER UNE LIGNE

Nous commençons par considérer une application simple qui affiche une ligne de texte (figure 2.1). Une application est un programme qui s'exécute en utilisant l'interpréteur Java. Pour des raisons pédagogiques, chaque ligne est numérotée. Les numéros de ligne ne font pas partie du programme.

```
1. // prog 2.1: Premier.java
2. // Un premier programme en Java
3.
4. public class Premier {
5.
6.        // Méthode main commence l'exécution d'un programme Java
7.        public static void main(String args[]) {
8.           System.out.println("Premier programme en Java");
9.        } // Fin de la méthode main
10.
11.} // Fin de la classe Premier
```

Figure 2.1 Un premier programme en Java

La ligne 8 représente la ligne qui affiche le texte "Premier programme en Java" à l'écran. Considérons maintenant le programme ligne par ligne. La ligne 1

```
// prog 2.1: Premier.java
```

commence par le symbole //, indiquant que le texte restant de la ligne est un **commentaire**. Ce dernier ne fait pas partie du programme Java qui sera compilé et exécuté par l'ordinateur. Ainsi, il est complètement ignoré par le compilateur et sert uniquement à améliorer la lisibilité et la compréhension du programme.

Un commentaire commençant par le symbole // est appelé commentaire "simple ligne" car le commentaire finit à la fin de la ligne courante. Un commentaire simple ligne peut également commencer au milieu d'une ligne et finir à la fin de la même ligne.

Il existe également des commentaires à lignes multiples. Par exemple, le code suivant

```
/* Ceci est
un commentaire
à lignes multiples */
```

est un commentaire éclaté sur plusieurs lignes. Ce type de commentaire commence par le délimiteur /* et finit par */. Tout le texte qui se trouve entre les deux délimiteurs est ignoré par le compilateur.

La ligne 2

```
// Un premier programme en Java
```

représente également un commentaire qui décrit le but du programme. La ligne 3 est une ligne vide. Les lignes vides et les espaces sont utilisés par les programmeurs pour rendre les programmes plus lisibles. Comme les commentaires, les lignes vides et les espaces sont ignorés par le compilateur. La ligne 4

```
public class Premier {
```

définit un programme appelé Premier. Plus exactement, la ligne 4 définit la classe Premier. Tout programme en Java doit contenir au moins la définition d'une classe. Le **mot clé** class introduit la définition d'une classe en Java et est immédiatement suivi par le nom de la classe (Premier dans notre cas). Un mot clé (ou mot réservé) est un mot réservé à l'utilisation par Java et est toujours épelé en caractères miniscules. Ainsi, le nom d'une classe ou d'une variable ne peut être un mot réservé. Par exemple, public class class définissant une classe portant le nom class ne peut être accepté en Java.

Pour raisons de simplicité, considérons pour le moment que toute classe commence par le mot clé public (Dans le chapitre 6, nous allons discuter d'autres types de classes). Quand vous voulez sauvegarder votre fichier, il est nécessaire de nommer le fichier avec le même nom que celui de votre classe publique suivi de l'extension .java. Par exemple, le programme de la figure 2.1 doit être sauvegardé dans un fichier portant le nom Premier.java.

Revenons maintenant au programme de la figure 2.1. Une accolade ouvrante { (à la fin de la ligne 4) introduit le corps de toute classe. Une accolade fermante correspondante }

(début de la ligne 11) doit terminer toute définition de classe. Remarquons que les lignes 7-9 sont indentées, c'est-à-dire qu'elles sont écrites en retrait par rapport à la ligne 4 et 11 signifiant qu'elles appartiennent au bloc qui commence à la ligne 4 et qui finit à la ligne 11. Cette indentation est l'une des conventions d'espacement et l'une des bonnes pratiques de programmation.

La ligne 5 est une ligne vide insérée pour plus de lisibilité. La ligne 6

```
// Méthode main commence l'exécution d'un programme Java
```

est un commentaire indiquant le but des lignes 7-9. La ligne 7

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

est une composante nécessaire de tout programme Java. En effet, tout programme Java commence son exécution par main. Les parenthèses après main indiquent que main est un bloc de programme appelé **méthode**. Une classe java (et donc un programme Java) contient normalement une ou plusieurs méthodes. Pour une classe Java, seulement une de ces méthodes doit être la méthode main et doit être définie tel qu'illustrée à la ligne 7.

Les méthodes exécutent un ensemble de tâches et peuvent retourner une information lorsque ces tâches sont accomplies. Le mot clé void indique qu'une méthode exécutera un ensemble de tâches mais ne retournera à la fin aucune information. Dans les chapitres suivants, nous discuterons de méthodes qui retournent une certaine information. Pour le moment, simplement imitez le contenu de la ligne 7 pour vos applications Java. L'explication du reste de la ligne 7 sera donnée dans les chapitres ultérieurs. L'accolade gauche { en fin de la ligne 7 initie le corps de la méthode main. Une accolade droite } finit la définition de la méthode (ligne 9). Notez que la ligne contenue dans le corps de la méthode est indentée entre les accolades.

La ligne 8

```
System.out.println("Premier programme en Java");
```

instruit l'ordinateur d'exécuter une action, en l'occurrence afficher la chaîne de caractères contenue entre les doubles quottes. Une chaîne de caractères peut être considérée comme un message. Les espaces blancs dans une chaîne ne sont pas ignorés par le compilateur. System.out est considéré comme étant l'objet de sortie standard du système d'exploitation utilisé. Il peut s'agir soit de l'écran, soit de l'imprimante ou tout autre périphérique de sortie externe. En général, la sortie standard correspond à l'écran. System.out permet aux applications Java d'afficher les chaînes de caractères ainsi que d'autres informations dans la fenêtre de console (ou la fenêtre de commandes) à partir de laquelle le programme est exécuté. La méthode System.out.println affiche une ligne de texte dans la fenêtre de console et passe à la ligne suivante, i.e. que lorsque System.out.println accomplit sa tâche, il positionne automatiquement le curseur au début de la ligne suivante. Par contre, la méthode System.out.print affiche une ligne de texte et le curseur reste positionné sur la même ligne et exactement à la fin du texte en question. Ainsi, l'exécution de la ligne 8 donne le même résultat si cette dernière avait été divisée aux deux instructions suivantes :

```
        System.out.print("Premier programme ");
        System.out.println("en Java");
```

La ligne 8 entière, incluant System.out.println, ses arguments entre les parenthèses (la chaîne de caractères) et le point virgule ";" constitue une instruction. Toute instruction doit finir par un point virgule. Lorsque le compilateur Java exécute l'instruction de la ligne 8, il affiche le message Premier programme en Java sur la fenêtre de console.

Certains programmeurs trouvent des difficultés à lire ou à écrire un programme tout en respectant la correspondance entre les accolades gauche et droite de la définition d'une classe ou d'une méthode. Pour cela, les programmeurs préfèrent inclure un commentaire après l'accolade droite qui finit une classe ou une méthode. Par exemple, la ligne 9

```
    } // Fin de la méthode main
```

spécifie l'accolade droite qui finit la méthode main, et la ligne 10

```
} // Fin de la classe Premier
```

spécifie l'accolade droite fermante de la classe Premier.

2. COMPILER ET EXECUTER UN PROGRAMME JAVA

Maintenant, il est temps de compiler et exécuter notre programme. Commencez d'abord par enregistrer le fichier dans le répertoire bin de votre jdk (la version de java que vous avez installée). En supposant que vous ayez installé Java (version jdk1.5) directement sous la racine C :, le chemin complet de votre fichier est alors : C:\jdk1.5\bin\Premier.java. Afin de compiler un programme Java, il suffit d'ouvrir une fenêtre de console, se positionner sur le répertoire où est enregistré votre fichier et saisir la commande javac <nomDuFichier.java>. Dans notre cas, la compilation du fichier Premier.java (Figure 2.2) serait : javac Premier.java

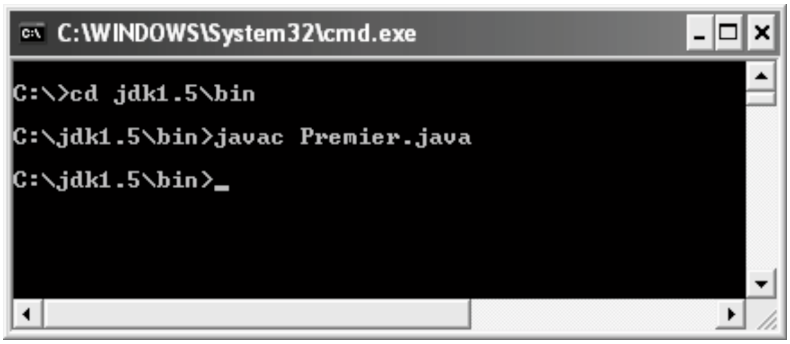

Figure 2.2 La compilation du programme Premier.java

Si le programme ne contient aucune erreur, la commande javac crée un nouveau fichier portant le même nom que le fichier .java mais avec l'extension .class. Dans notre cas, le nouveau fichier créé est nommé Premier.class (Figure 2.3).