java : Langage et Environnement

1. Introduction

Rava est une nouvelle technologie conçue par Sun Microsystems l'un des premiers fabriquants des stations de travail Sun. Depuis son apparition, à peine plus d'une année, java est devenu l'un des pilliers de la programmation dans le monde; grâce à l'animation qu'autorisait ce langage et surtout grâce à sa relation directe avec le web. Cependant, le web ne permet que la distribution de l'information statique à travers le réseau, il rend donc ses clients passifs. Avec l'avènement de java, les pages web sont devenues interactives et intelligentes 11].

Les fonctionnalités de java ne se limitent pas seulement à l'animation et au web, mais c'est un langage à part entière qui peut rendre de grands services dans de nombreux domaines d'application n'ayant rien avoir avec le web.

Le développement de java s'est inscrit dans le cadre d'un projet très vaste dont l'objectif étant de produire des composants logiciels devant répondre aux contraintes habituelles (robustesse, fiabilité, portabilité, transparence réseau,...) En plus, ces composants devraient aussi pouvoir se balader sur le réseau. Ce dernier point a des conséquences importantes sur l'architecture de l'environnement associé au langage. Sachant qu'il était hors de question de faire voyager du code non compilé, il fallait se résoudre à émuler. Cette solution peut sembler irréaliste mais il suffit de bien choisir le processeur à émuler. Un processeur abstrait a donc été décrit par Sun et ses spécifications sont libres.

A ce niveau, il ne s'agit plus de compiler un programme pour ce processeur abstrait. On obtient ainsi un exécutable indépendant de toute architecture. Il suffit ensuite d'émuler ce processeur pour pouvoir exécuter le programme sur n'importe quelle architecture réelle.
Il est important de remarquer que cette idée n'est pas nouvelle et surtout qu'elle peut être utilisée avec n'importe quel langage. En fait, le choix de départ de Sun pour le langage était C++. Ce choix a été abandonné. Sun a donc choisi de simplifier et de moderniser C++ pour en arriver au langage java.

2. Les **origines de Java**

Les origines de Java remontent à 1990, alors que le World Wide Web n'existait pas encore et que l'Internet ne bénéficiait pas de la publicité et de la couverture actuelle. L'idée est née dans l'esprit de trois chercheurs, James Gosling, White Fiels Diffle et Bill Joy [5].. Ils définissent quelques principes de base pour un nouveau projet: construire un environnement de petite taille pour des machines destinées au grand public. lin 1991, Gosling présente Oak, langage qui sera rebaptisé Java en Décembre 1995, après avoir été adapté à Internet. Depuis le 23 Mai 1995, jour ou Sun a officiellement présenté Java, le langage a beaucoup fait parler de lui; d'abord par sa relation avec Internet puis par ses qualités intrinsèques qui en font un langage à part entière. Java est un langage orienté objet; son grand atout est de permettre le développement d'applications de taille réduite, lisible sur toute plate-forme.

3. **Caractéristiques** du **langage**

Java tel qu'il est défini par Sun est un langage *"orienté objet, simple, distribué, interprété, robuste, sécurisé, neutre d'architecture, portable, hautement performant, mullithread et dynamique"*[3]. Cette énumération décrit les caractéristiques d'un langage capable de délivrer des applications dans un environnement hétérogène distribué indépendamment de toute plate-forme où les aspects de performance et de sécurité sont pris en compte. Nous allons décrire chacune de ces propriétés en montrant les éléments de Java qui permettent de les supporter.

Java est orienté objet: Java adopte les principes fondamentaux des langages objets. Il respecte l'abstraction, l'encapsulation, l'envoi de messages, l'héritage et le polymorphisme. Ce langage est très proche d'"Objective G" et de "SmallTalk" et sa syntaxe est similaire à celle de C++. Java n'utilise pas l'héritage multiple mais offre un mécanisme d'interfaces qui permet de le simuler. L'avantage majeur de la programmation orientée objet est que les objets, méthodes et classes créés peuvent être réutilisés. Cela permet d abaisser les coûts de développement de façon non négligeable.

Java est plus ***simple*** *que* ***C++*** *:* Contrairement à C++, qui autorise, sans l'imposer, la programmation objet, Java pousse le programmeur à utiliser ce paradigme. Il est "plus orienté objet" en quelque sorte. En omettant plusieurs concepts complexes, tels que les pointeurs (il est impossible d'accéder à une zone mémoire) Java offre plus de simplicité. Il utilise pour la gestion de la mémoire le paradigme du ramasse-miettes (Garbage collector), pour décharger le programmeur des problèmes d'allocation et de libération de la mémoire.

Java est distribué : Les appels aux fonctions d'accès réseau et les protocoles Internet **(TCP/IP)** les plus utilisés tels que HTTP, FTP, TELNET sont intégrés dans Java. Ils permettent d'exécuter des processus à distance. Une application client-serveur peut donc être écrite très simplement sous Java.

Java est portable et interprété : Le code Java est le même pour tous les ordinateurs. Lorsqu'un programme Java est compilé, il est transformé en une sorte de pseudo-assembleur qui n'est propre qu'aux compilateurs Java. Par la suite, c'est la machine virtuelle (décrite dans la suite de l'article) qui interprète ce pseudo-assembleur pour pouvoir utiliser Java sur n'importe quelle architecture. Actuellement, des environnements ont été écrits pour: Windows 95, Windows NT. Solaris 2.3 et +. Irix. OS/2 Warp. Aix., Unixware 2.0, HP/UX, MVS, Mac OS 7.5 pour les PowerMacs 68020. 68030, 68040.

Java est robuste : Un programme Java ne peut pas planter votre machine car l'environnement d'exécution de Java gère les erreurs que les bugs ou les accès non autorisés à la mémoire peuvent occasionner, et affiche un message précisant la nature de l'erreur.

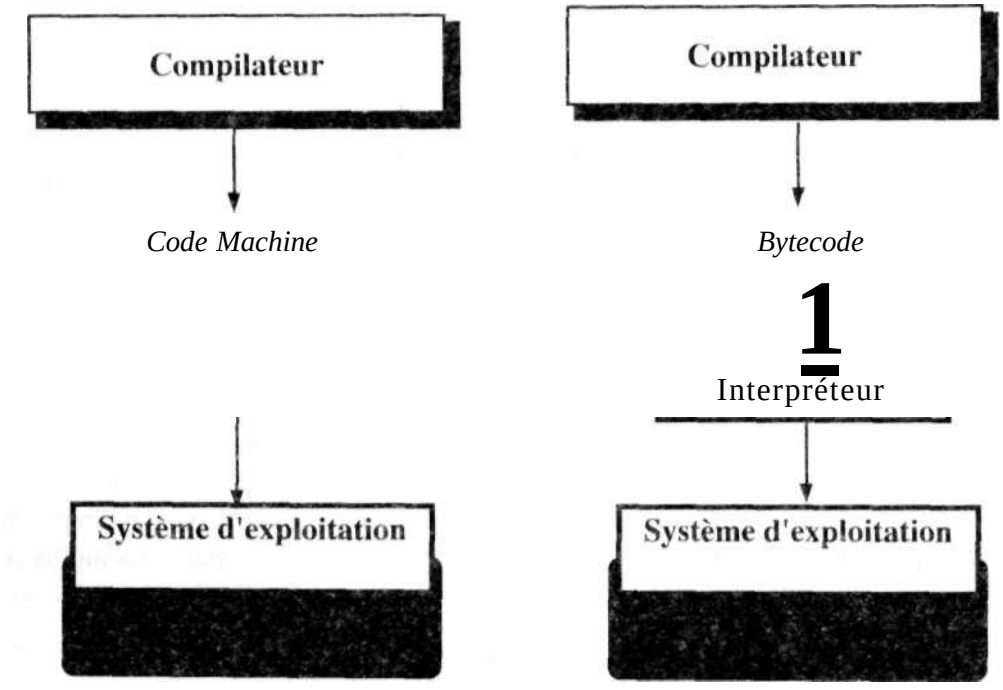
Java est sécurisé : Le problème majeur des programmes circulant sur le réseau est la sécurité. Java intègre dés sa conception plusieurs mécanismes de sécurité qui visent à rendre les programmes fiables. La sécurité recouvre la protection contre les virus, la gestion automatique de la mémoire et toute sorte de vérification de programmes. La sécurité dans Java est d'une importance majeure vu que ce langage est conçu pour implémenter des applications sur le Web. De par son architecture, il est courant de télécharger une application Java distante pour son exécution locale. Il est donc important pour l'interpréteur de s'assurer qu'une application n'a pas été altérée depuis sa compilation. Un mécanisme de contrôle complet a été incorpore afin de vérifier l'authenticité du code.

Java autorise les multi-threads : Java permet l'exécution parallèle de différentes tâches (threads) de façon autonome. Il inclut le support des threads multiples allégeant considérablement l'écriture de programmes qui utilisent ces fonctionnalités. Java gère aussi le mécanisme des moniteurs, qui permet de synchroniser ces "threads".
Java est dynamique : L'édition de iien entre les modules objets se fait dynamiquement au moment de l'exécution. Cela permet, dans le cas où une librairie de classes évolue, de ne pas avoir à modifier ou recompiler l'application qui lui fait appel.

4. La Machine Virtuelle de Java

L'Internet est multiple. De nombreux systèmes d'exploitation cohabitent, de nombreuses versions de logiciels, d'architecture matérielles. Le maître mot semble être hétérogénéité. Java se doit d'être dépendant de ces couches qu'il ne maîtrise pas. C'est pourquoi, le compilateur Java, contrairement aux compilateurs traditionnels, ne génère pas dès instructions machines spécifiques mais un programme en "Bytecode" (figure 1), qui peut être décrit comme un langage machine pour un processeur virtuel qui n'a pas d'existence physique. Ce code objet compilé peut alors être exécuté par un interpréteur Java qui n'est qu'un émulateur de processeur virtuel : la machine Virtuelle java. Cet interpréteur

transforme ce Bytecode (formai de bas niveau représentant la séquence d'instructions à exécuter) en un langage machine de l'ordinateur lorsqu'une application Java est téléchargée. Il suffit donc qu'il existe des interpréteurs différents selon l'ordinateur et k système d'exploitation pour pouvoir utiliser des programmes Java sur n'importe quelk architecture, ce qui permet donc à Java d'être puissant, flexible et portable [5|.



Environnement Compilé **Environnement Interprété de Java**

5. Les Spécificités du langage Java

Cette section décrit succinctement les spécificités du langage Java, incluant les concept orientés objets et les mécanismes importants qu'il renferme.

5.1 Objets, Classes, Interfaces et Packages

Comme tous les langages orientés objet, le langage Java offre les notions de classt d'objet et de dérivation. Un programme Java crée et manipule un ensemble **d'objets**qu interagissent entre eux par l'envoi de messages. Le résultat de ce message est l'invocatio d'une méthode qui exécute une action donnée ou modifie l'état d'un objet. A travers ce interactions, le programme Java peut implémenter une interface graphique, exécuter un animation, ou encore émettre ou recevoir des informations à travers le réseau[2|.

Les **classes** de Java forment le software de base pour supporter les caractéristiques d tout objet. Une classe Java définit les données de chaque instance, les données qu

partagent tous les objets de ce type et les méthodes rattachées à cette classe.
Toutes les classes de Java, exceptée la classe *Object,* héritent d'une autre classe; si aucune superclassc n'a été spécifiée, la classe *Object* est supposée sa superclasse.

La sous-classe hérite l'état et le comportement de sa super-classe. Parfois, pour des raisons de sécurité ou de conception, une classe ou une méthode ne doit pas être héritée. Il est possible dans Java, de l'assurer en qualifiant cette classe ou méthode du mot clé: final. Java introduit également la notion de classes ou méthode abstraite. Ces classes ne sont jamais instanciées et les méthodes ne peuvent pas être surdéfinies.

Contrairement à C++, Java ne supporte pas l'héritage multiple: une classe Java ne doit hériter que d'une seule superclasse. L'héritage simple facilite la conception. Cependant, il est restrictif, en particulier lorsqu'un comportement similaire a besoin d'être dupliqué à travers différentes branches de la hiérarchie des classes. Java résoud ce problème en utilisant le concept d'interfaces. Une interface spécifie une collection de méthodes non implémentécs pouvant être partagées par plusieurs classes. Bien que la hiérarchie des classes est simple, celle des interfaces peut être multiple.

Pour organiser les objets et éviter un conflit de noms, les programmeurs peuvent regrouper les objets dans des librairies. Dans Java, une librairie est appelée: package. Le concept de packages dans Java permet de regrouper les classes et les interfaces reliées entre elles. Le programmeur peut créer ses propres packages en incluant les classes et les interfaces. L'environnement de développement de Java offre différents packages. Les classes et les interfaces contenues dans ces packages peuvent être réutilisées.

5.2 Les Librairies de JAVA

Le système complet JAVA inclut un grand nombre de librairies de classes et de méthodes utiles et de méthodes. Ces librairies sont :

java.lang - la collection des types de base (type de langage) qui sont toujours importés dans toute unité de compilation. C'est là où se trouvent les déclarations d'Object (la racine de la hiérarchie des classes), Class, Threads et Exceptions.

Java.io - L 'équivalent de la librairie Standard I/O, familière sur la plupart des systèmes UNIX.

Java.net - Fournit les supports pour les sockets, les interfaces Telnet et les URLs.

Java.util - Cette librairie contient des classes telles que Dictionary, HashTable et Stack, ainsi que des techniques d'encodage et de décodage, et les classes Date et Time.

Java.awt - (Abstract Windowing Toolkit) Fournit au programmeur toutes les primitives nécessaires a ta gestion de l'interface graphique utilisateur. Cette librairie contient des classes d'interfaces de base telles que les événements, les couleurs, les fontes, et les contrôles tels que les boutons ou barres de menu.

5.3 Exceptions **de Java**

.hua supporte un mécanisme de reprise sur erreurs en utilisant les exceptions. Une exception est un événement qui survient lors d'un cas de figure anormal durant l'exécution d'un programme, interrompant ainsi le flot normal des instructions et exécuian: les éventuelles toutines de traitement[3J.

Les exceptions sont des objets de Java, elles sont insianciées de la classe Exception. Différents types d'erreurs peuvent causer des exceptions, ces erreurs peuvent être aussi bien de simples erreurs de programmation, telles que la tentative d'accès en dehors des limites d'un tableau, que de sérieux problèmes hardware. Le système tic base de Java définit un ensemble d'exceptions utilisées par le système pour signaler les conditions d'erreurs. Quand une erreur apparaît dans un programme, une exception peut être appelée. Certaines exceptions sont prédéfinies par le système mais peuvent être redéfinies à volonté.

5.4 Le multithreading en JAVA

Une activité est appelée un 'thread' en Java. Plusieurs 'threads' peuvent permettre d'effectuer des tâches de fond pendant que le programme continue d'interagir avec l'utilisateur. Java permet de les synchroniser **facilement[2].**

Les supports de threading intégrés à JAVA donnent aux programmeurs la possibilité d'exploiter tous les avantages de l'interfaçage graphique interactif. La librairie JAVA fournit une classe Thread, qui inclut un grand nombre de méthodes pour lancer un thread, l'exécuter, .'arrêter, ou encore regarder son statut.

JAVA supporte le multithreading au niveau du langage et via le support du système run-time ainsi que les objets threads. Au niveau du langage, les méthodes d'une classe déclarées en *synchronized* ne sont pas exécutées en même temps. De telles méthodes sent exécutées sous contrôle des moniteurs.

5.5 Les méthodes **natives**

Java offre la possibilité d'implémenter éventuellement une méthode (méthode d'instance ou méthode de classe) dans un autre langage de programmation tel que C ou C++. Cette méthode est appelée **méthode native.** L'avantage des méthodes natives est d'implémenter des traitements spéciaux non prévus par les librairies Java, tels que l'interfaçage à un nouveau périphérique (exemple: l'interfaçage à un microphone n'est

pas encore traité dans Java). L'inconvénient des méthodes natives est que leur utilisation réduit le facteur portabilité du code[2).

6. Environnement de développement Java

Le Kit de Développement Java JDK fourni par Sun comprend :

Les différents *Packages* de TAPI Java de Sun

Package	Classes présentes
Java applet	Classes de base pour les applets
Java, awt	Classes d'interface graphique AWT
Java. awt. image	Classes de gestion des images AWT
Java.awt.peer	Classes d'interfaçages aux environnements natifs
Java.io	Classes d'entrées/sorties (flux, fichiers...)
Java.lang	Classes de support du langage
Java.nel	Classes de support réseau (URL, *sockets...)*
Java.util	Classes d'utilitaires (vecteur, *hashtablc.)*

Les outils de développement

Package	Classes Présentes
Java	le compilateur, il traduit un fichier source Java *java en un fichier compilé en pseudo-code *.class;
Java	la Machine Virutuelle Java, permet l'exécution d'une application Java
Appletviewer	Comme son non l'indique, est un "viewer", il prend un fichier HTML en argument et exécute les applets Java qu'il y trouve
Jdh	le débogueur pour la mise au point des programmes Java.

Les autres *Packages*

Package	Classes Présentes
Java Server **API**	Permet la création de "Servlets", petits programmes s'exécutant sur le serveur (par exemple, on peut envoyer une Servlet sur en serveur pour effectuer une demande de service plus ou moins complexe).
Java Entreprise API	Vient se rajouter avec notamment Java **RMI** qui permet à deux objets Java d'invoquer des méthodes à travers le réseau.
JDBC	Java DataBase Connectivity. qui es! l'interface d'accès aux bases de données
Java Beans **API**	est une API pour composant logiciel, permettant à une applet de s'interconnecter avec des composants tels que OLE/COM/ActiveX de Microsoft;

7. JAVA et les autres langages de programmation

Il existe des centaines de langages disponibles pour les développeurs afin d'écrire des programmes permettant de résoudre des problèmes dans des domaines précis.

Les langages au niveau des Shells, par exemple, sont des langages interprétés de haut-niveau. Ils travaillent au niveau du système sur des objets qui sont des fichiers et des processus plutôt que des structures de données Les langages interprétés sont facilement portables. Leur principal inconvénient est leur performance; ils sont en général beaucoup plus lents que tout autre langage machine natif ou bytecode interprété. Cela a peu d'importance si le programme est court et s'il est exécuté rarement.

Ensuite viennent les langages tels que Perl, qui ont beaucoup de caractéristiques communes avec JAVA telles que la robustesse, le comportement dynamique et la neutralité de l'architecture. L'évolution de Perl a mené ce langage vers l'adoption de caractéristiques orientées-objet, sécurisées, et il présente des caractéristiques communes avec JAVA.

Au dernier niveau viennent tes langages compilés, tels que C ci C++, grâce auxquels il est alors possible de développer des projets dans des domaines extrêmement variés, et de haute performance. Java est environ 20 fois plus lent qu'un programme écrit en C. Cela peut suffir pour certaines applications mais pas pour toutes. Cependant des travaux sont

en cours chez. Sun pour écrire des run-time plus puissants.
Le langage JAVA est en t'ait un compromis entre les langages de hailt-niveau, les langages scripts portables, mais lents, et les langages compilés, non portables mais rapides [4]..

8. Conclusion

Pour les développeurs, Java dispose de nombreux atouts, qui n'avaient jamais été réunis dans un seul langage auparavant. Le grand atout de Java consiste en fait à regrouper dans un seul langage les caractéristiques les plus intéressantes des autres langages. Le caractère universel de Java en fait un langage de choix dés que l'on vise plusieurs plateformes. Les codes Java étant indépendants de l'architecture cible, les applications Java sont donc particulièrement bien adaptées à des environnements réseaux hétérogènes tel qu'Internet.

En fait, Java est la révolution annoncée de la programmation Client/Serveur et de l'Internet. Quoi de plus pratique, pour une entreprise qui possède un parc informatique diversifié, que de pouvoir faire communiquer des ordinateurs qui, autrefois, n'échangeaient des données qu'au prix de logiciels coûteux ? Ajoutons à cela qu'en réseau local, les applications Java peuvent être installées sur un serveur, et mises à la disposition des employés sans pour autant qu'ils aient à installer quelque chose sur leur ordinateur personnel... D'où des coûts de support et de mise à jour réduits...

^Z^ZUZIZZ, Références Bibliographiques $_m$^$_m$

[l].The Java Languagc : An Ovcrview Sun Microsystems 1995
http://litsim.ccntrc.edu/java/docs/java-ovcrview

[2].Thc Java Tutorial: Object Oricntcd Programming for thc Internet
M. Campione, K. Walrath
1997
http://java.sun.com/nav/read/Tutorial

[3].Teach yourself Java in 21 days
L. Lcmay, C. L. Pcrkins
1996
http://www.lne.com/Wcb/Java

[4].The Java White Paper
http://arrakis.ucd.ic/~paulmc/java

[5].Thc Java Saga
D. Bank
1996
http://www.hfd.hr/Studcnt/clanci/java.html

[6].Java: la renaissance du client-serveur
Mai, 1996
http://www.mygale.org/09/yboutin/java.html

[7|.Java: la révolution logicielle
Institut Prométhéus. 1996
http://www.promcthcus.eds.fr/actu/article