Atelier Programmation Orientée Objet Avancée –JAVA-

TP1

ISET Zaghouan

Enseignant

Boukchim_mossaab@yahoo.fr

Premiers Pas

I. 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 (Application Programming Interface) de Java (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\bin** à 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.

II. 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 classe 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 nomrépertoire** 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.

III. L'exécution d'un programme java

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

java NomFichier et dans notre exemple **java HelloWorld**

Les Exercices

Exercice 1 :

On se propose de faire fonctionner un programme Java dont le rôle est d'afficher « J'aime JAVA ! »

Exercice 2 :

On se propose de faire fonctionner un programme Java dont le rôle est d'afficher le premier mot qu'on lui passe comme paramètre d'exécution.

Exercice 3:

Créer un fichier JouerAvecMethode.java contenant la classe JouerAvecMethode

2- Ajouter une méthode void Afficher () qui fait l'affichage d'une chaîne de caractères exemple : « Bienvenu à l'ISET Zaghouan !»

3- Ajouter la méthode main dans laquelle vous faite instancier la classe JouerAvecMethode et référence l'instance ainsi créée à une variable s qu'on déclare comme suit :

JouerAvecMethode s=new JouerAvecMethode () ;

4- Dans la méthode main faite un appel à la méthode Afficher () de l'objet s comme suit :

s. Afficher() ;

Exercice 4:

1-Créer un fichier JouerAvecAttribut.java contenant la classe JouerAvecAttribut

2- Ajouter un attribut String maChaine à la classe

3- Ajouter une méthode void Afficher () qui fait l'affichage de la chaîne de caractères attribut de la classe comme suit System.out.println(this.maChaine);

4- Ajouter la méthode main dans laquelle vous faite instancier la classe JouerAvecAttribut et référence l'instance ainsi créée à une variable s qu'on déclare comme suit :

JouerAvecAttribut s=new JouerAvecAttribut () ;

5- Dans la méthode main faite affecter l'attribut de l'objet s comme suit s.maChaine=''Salut ''

6- Dans la méthode main faite un appel à la méthode Afficher () de l'objet s comme suit

s. Afficher () ;

Les Annexes

Annexe 1 : Préface – Java –

I. Introduction

Dans les années 80, les micros ordinateurs ont obtenu les capacités nécessaires pour supporter des interfaces graphiques. La complexité des applications utilisant des interfaces graphiques est importante car il faut gérer un nombre important d'actions de bas niveau (par exemple, les fenêtres sur l'écran). Il fallait donc décharger le concepteur d'applications de toutes ces difficultés, l'une des raisons qui a favorisé la naissance des outils de développement pour des applications basées sur les interfaces graphiques. La programmation événementielle et les langages de programmation associés sont donc apparus à cette époque.

II. Les types de programmations

Il existe un ensemble de langages de programmation, chacun est spécialisé dans un domaine d'application donné et chacun possède un type spécifique. On distingue :

- ✓ *Programmation structuré ou modulaire* : le programme est vu comme un ensemble d'unités structurées hiérarchiquement. Il existe une interaction entre les modules et on fait généralement distinction entre les données et les traitements.
- ✓ *Programmation orientée objet* : le programme n'est autre qu'une collection d'objets qui communiquent. Un objet et par définition une instance d'une classe dont les composantes peuvent être de deux types : propriétés et méthodes.
- ✓ *Programmation événementielle* : Les composants d'une application événementielle c'est à dire les objets interagissent entre eux et avec l'environnement. Ils communiquent en réponse à des événements. Ces événements peuvent correspondre à une action de l'utilisateur : un click sur un bouton de commande, une écriture dans une zone de texte, un choix dans une case d'option ou une case à cocher, le déplacement d'un objet, ... Ils peuvent aussi être déclenchés par le système : chargement d'une feuille, un top déclenché par l'horloge, ...

III. Java

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

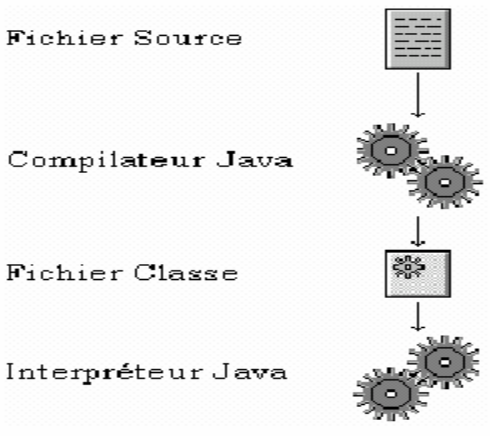
Vers la fin de 1995, le langage de programmation Java (Développé par Sun Microsystems) surgit sur la grande scène d'Internet et obtient immédiatement un énorme succès. Il s'agit d'un langage de conception très performant qui a été adopté par la majorité des fournisseurs. Ses caractéristiques intégrées de sécurité offrent un sentiment de confiance aux programmeurs comme aux utilisateurs des applications. De plus, Java incorpore des fonctionnalités qui facilitent grandement certaines tâches de programmation avancées comme la gestion des réseaux, la connectivité des bases de données ou le développement d'applications multitâches.

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, ...)

1. 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 quelque 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 étendue; 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.

2. Les avantages de Java

Les avantages de ce langage se résument comme suit :

- Sa bibliothèque d'exécution est **indépendante de la plateforme** : en théorie, il est possible d'utiliser le même code pour Windows 95/98/NT, Solaris, UNIX, Macintosh, etc. Cette propriété est indispensable pour une programmation sur Internet ;
- La syntaxe de Java est analogue à celle de C++, ce qui le rend **économique et professionnel**;
- Il est beaucoup plus facile d'obtenir du **code sans erreur** à l'aide de java qu'avec C++ car dans Java sont ajoutées des fonctions destinées à éliminer la possibilité de créer du code contenant les types d'erreurs les plus courants :
    - Les concepteurs de java ont supprimé l'allocation et la libération manuelles de mémoire. **La mémoire dans java est allouée et libérée automatiquement** ;
    - Ils ont éliminé l'arithmétique des pointeurs introduisant du même coup une vraie gestion de tableau. La notion de **référence sur une zone mémoire** remplace avantageusement celle de "pointeur", car elle supprime la possibilité d'écraser toute zone mémoire à cause d'un compteur erroné ;
    - Ils ont éliminé toute possibilité de confusion entre une affectation et un test d'égalité dans une instruction conditionnelle ;
- Ils ont supprimé l'héritage multiple en le remplaçant par une nouvelle notion d'interface dérivée d'Objective C. Les interfaces offrent tout ce qui est obtenu à partir de l'héritage multiple, sans la complexité de la gestion de hiérarchie d'héritage multiple.

3. Caractéristiques de Java

Les caractéristiques fondamentales de Java se résument selon les 13 termes suivants :

- Simple : Java a été conçu de façon relativement proche du langage C++. De nombreuses fonctions compliquées, mal comprises, rarement utilisées de C++ (pointeur, surcharge dynamique, héritage multiple,…), qui semblent apporter plus d'inconvénients que d'avantages, ont été supprimées de Java.
- Fiable : Java a été conçu pour que les programmes qui l'utilisent soient fiables sous différents aspects. Sa conception encourage le programmeur à traquer préventivement les éventuels problèmes, à lancer des vérifications dynamiques en cours d'exécution et à éliminer les situations génératrices d'erreurs... La seule et unique grosse différence entre C++ et Java réside dans le fait que ce dernier intègre un modèle de pointeur qui écarte les risques d'écrasement de la mémoire et d'endommagement des données.
- Orienté objet : Java se concentre sur les objets et sur les interfaces avec ces objets. Java offre de nombreuses classes permettant de définir et de manipuler les objets.
- Distribué : Java possède une importante bibliothèque de routines permettant de gérer les protocoles TCP/IP tels que HTTP et FTP. Les applications Java peuvent charger et accéder à des pages sur Internet via des URL avec la même facilité qu'elles accèdent à un fichier local sur le système.

  Les fonctionnalités réseau de Java sont à la fois fiables et d'utilisation aisée. Java est simple d'utilisation lorsqu'il s'agit de mettre en oeuvre des tâches lourdes, comme l'ouverture d'une connexion avec un socket. De plus, Java rend plus facile l'élaboration des scripts CGI (Common Gateway Interface), et un mécanisme élégant, nommé servlet, augmente considérablement l'efficacité du traitement côté serveur, assuré par Java. De nombreux serveurs Web, parmi les plus courants, supportent les servlets. Le mécanisme d'invocation de méthode à distance (RMI) autorise la communication entre objets distribués.
- Sécurité : Java a été conçu pour être exploité dans des environnements serveur et distribués. Dans ce but, la sécurité n'a pas été négligée. Java permet la construction de systèmes inaltérables et sans virus.
- 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 y avoir d'accès direct à la mémoire. L'accès au disque dur est réglementé dans une applet. Les programmes fonctionnant sur le Web sont soumis 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 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.

- Architecture neutre : Le compilateur génère un format de fichier objet dont l'architecture est neutre. Le code compilé est exécutable sur de nombreux processeurs, à partir du moment où le système d'exécution de Java est présent. Pour ce faire, le compilateur Java génère des instructions en bytecode qui n'ont de lien avec aucune architecture particulière. Au contraire, ces instructions ont été conçues pour être à la fois faciles à interpréter et faciles à traduire en code natif.
- Portable : A la différence du C/C++, on ne trouve pas les aspects de dépendance de la mise en oeuvre dans la spécification. Les tailles des types de données primaires sont spécifiées, ainsi que le comportement arithmétique qui leur est applicable.
- Interprété : Le source est compilé en pseudo code puis exécuté par un interpréteur Java (la Machine Virtuelle Java (JVM)).
- Indépendant de toute plateforme : L'interpréteur Java peut exécuter les bytecode directement sur n'importe quelle machine sur laquelle il a été porté. 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 JVM. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du bytecode.
- Performances élevées : En général, les performances des bytecodes interprétés sont tout à fait suffisantes, il existe toutefois des situations dans lesquelles des performances plus élevées sont nécessaires. Les bytecodes peuvent être traduits à la volée en code machine pour l'unité centrale destinée à accueillir l'application.
- Multithread : Il permet l'utilisation de threads qui sont des unités d'exécution isolées. La JVM elle-même utilise plusieurs threads. Ce qui permet des traitements multitâches

dans une même application. Les avantages du multithread sont une meilleure inter-réactivité et un meilleur comportement en temps réel.

- Dynamique : Java a été conçu pour s'adapter à un environnement qui évolue, et pousse le concept orienté objet à son optimum en permettant l'édition des liens entre modules objets dynamiquement au moment de l'exécution, en particulier dans le cas ou une application s'appuie sur une librairie de classes (une librairie de composants logiciels). Même si cette librairie de classes évolue, il n'est pas nécessaire de modifier ou de recompiler l'application qui y fait appel. Ces nouvelles versions de modules seron prises en compte sans problèmes dès le moment ou elles auront été installées.

Annexe 2 : Préface – L'approche objet –

- ✓ Un objet est une entité aux frontières précises qui possède une identité (un nom).
- ✓ Un ensemble d'attributs caractérise l'état de l'objet.
- ✓ Un ensemble d'opérations (méthodes) en définissent le comportement.
- ✓ Un objet est une instance de classe (une occurrence d'un type abstrait).
- ✓ Une classe est un type de données abstrait, caractérisé par des propriétés (attributs et méthodes) communes à des objets et permettant de créer des objets possédant ces propriétés.

L'encapsulation

L'encapsulation consiste à masquer les détails d'implémentation d'un objet, en définissant une interface. L'interface est la vue externe d'un objet, elle définit les services accessibles (offerts) aux utilisateurs de l'objet.

L'héritage

L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.

Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines.

Le polymorphisme

Le polymorphisme représente la faculté d'une même opération de s'exécuter différemment suivant le contexte de la classe où elle se trouve.

L'agrégation

Il s'agit d'une relation entre deux classes, spécifiant que les objets d'une classe sont des composants de l'autre classe. Une relation d'agrégation permet donc de définir des objets composés d'autres objets.