TP1 - Premiers pas en Java

Septembre 2007

Objectif

Les objectifs de ce premier TP sont la mise en place de l'environnement de travail en java et l'étude des fondements du langage. C'est-à-dire d'une part la compréhension de la compilation et de l'exécution de code java, l'arborescente classique d'une application et la génération de la documentation et d'autre part la maitrise de la création de classes et d'objets.

1 Une première exécution

1.1 Mettre en place l'environnement

□ La première chose à faire est de fixer quelle distribution du jdk va être utilisée, pour cela fixer la valeur de la variable d'environnement JAVA_HOME pour indiquer son emplacement (`/opt/jdk1.6.0`).
□ Ensuite, on ajoute le répertoire `$JAVA_HOME/bin` à la valeur de la variable `PATH` pour que les commandes de bases soient accessibles.
□ Vérifier que cela fonctione en exécutant les commandes `java -version` et `javac -help`

Il est impératif que **toutes** les documentations suivantes soient ouvertes dans votre navigateur web :

- la documentation sur les API :
  `http://java.sun.com/javase/6/docs/api/`
- la documentation sur les outils Java fournis par Sun :
  `http://java.sun.com/javase/6/docs/`

1.2 Compiler et exécuter une application

□ A la racine de votre compte ajouter un répertoire `SSI3` (ce répertoire sera appelé *répertoire de travail*). A l'intérieur de votre répertoire de travail ajouter le répertoire `TP` (appelé répertoire de *projet*). A l'intérieur de votre répertoire de projet ajouter les répertoires standards `src`, `build`, `doc`, `dist` et `lib`.

□ Dans le répertoire `src`, créer l'arborescence correspondant au paquetage :
`fr.univtln.`*`login`*`.tp.tp1`. A l'intérieur de ce paquetage ajouter le programme Java `PremierProgramme.java` (il est sous le fichier de ce TP sur le Wiki).
□ Ouvrir et modifier ce programme pour que l'instruction `package` soit correcte.
□ `A partir de votre répertoire de projet`, compiler le programme avec la commande suivante :

```
javac -sourcepath src -d build
src/fr/univtln/login/tp/tp1/PremierProgramme.java
```

□ Regarder dans le répertoire `build`. Pour exécuter le programme, depuis votre répertoire de projet :

```
java -classpath build
fr.univtln.login.tp.tp1.PremierProgramme Pierre
```

1.3 Générer la documentation

1.3.1 Précision sur la forme

La commande `javadoc` produit de la documentation en partant de commentaires particuliers insérés dans le code source des classes (/** . . . */). On peut ainsi documenter les paquetages, classes ou interfaces, variables d'instance, méthodes, . . .

Les commentaires peuvent contenir du texte simple et des balises HTML de mise en forme de texte (¡I¿ italique, ¡B¿ caractère gras, . . .). On peut utiliser la balise `<code>` pour inclure du code dans les commentaires. Des balises spéciaux à javadoc qui commencent par le caractère @ (@author, @version, @param,. . .). Les commentaires doivent être placés juste avant ce qu'ils commentent.
□ Lire cette page :
`http://java.sun.com/javase/6/docs/technotes/tools/solaris/javadoc.html`.

De plus, il est rappelé que le langage Java est entouré d'un ensemble de bonnes pratiques. Cette page présente les habitudes dans le cadre du langage java.
□ Lire cette page : `http://g.oswego.edu/dl/html/javaCodingStd.html`.

1.3.2 Génération

A partir de maintenant, **tous vos programmes seront commentés**.
□ Génerer la documentation depuis votre répertoire de projet avec la commande :
`javadoc fr.univtln.tp.tp1 src/fr/univtln/bruno/tp/tp1/PremierProgramme.java -d doc`

1.4 Générer et exécuter l'archive jar

□ Génerer l'archive de votre projet depuis votre répertoire de projet avec la commande :
`cd build; jar cvf ../dist/tp.jar fr; cd ..`
□ Exécuter le programme en ajoutant le jar au classpath et en indiquant la classe exécutable avec la commande suivante :
`java -classpath dist/tp.jar fr.univtln.`*`login`*`.tp.tp1.PremierProgramme Pierre`
Il est possible de rendre le jar " exécutable " en créant un fichier `manifest` qui indique en particulier quelle est la classe exécutable.
□ Copier le fichier `monManifest` dans le répertoire de projet et regarder son contenu.
Recréer le fichier `.jar` en précisant le manifeste à ajouter :
□ cd build; jar cvfm ../dist/tp.jar ../monManifest fr; cd ..
Puis exécuter directement l'archive :
□ `java -jar dist/tp.jar Pierre`

1.5 Automatiser le processus avec Apache Ant

Comme vous l'avez vu la compilation d'un projet complet en Java est une tâche fastidieuse. C'est pourquoi, il est conseillé d'utiliser un outil qui automatise ce travail : Ant (`http://ant.apache.org/`). Ant utilise un fichier de configuration qui indique les tâches à accomplir, vous trouverez un exemple de ce fichier sur le wiki `build.xml`.
□ Copier ce fichier dans votre répertoire de projet et regarder son contenu. Adapter son contenu pour qu'il indique des chemins corrects.
□ Créer une variable d'environnement `ANT_HOME` qui indique le répertoire d'installation de ant (`/usr/share/ant`). Ajouter `ANT_HOME/bin` dans le `PATH`.
Pour obtenir l'ensemble des tâches possibles avec un fichier de construction :
□ `ant -projecthelp build.xml`
Pour lancer l'exécution et si nécessaire compiler et générer l'archive :
□ `ant run`

1.6 Utilisation de l'environnement Eclipse

Nous allons maintenant utiliser l'environnement de développement eclipse :
`http://www.eclipse.org`. Comme " workspace " vous pouvez choisir votre espace de travail (c'est là que eclipse stocke ses informations de configuration).
□ Créer un nouveau projet à partir d'un fichier ant (cf. menu `file`).
□ Modifier les propriétés du nouveau projet (clic droit sur le projet). Dans "Java compiler" indiquer la compatibilité avec Java5. Dans les propriétés des chemins vérifier que les répertoires des sources et des binaires sont corrects.
□ Dans les propriétés du paquetage `JRE_LIB` indiquer où se trouve l'archive de la Javadoc.
□ Pour lancer une tâche ant, utiliser le menu de droite.
□ Vous pouvez maintenant éditer vos programmes Java.

2 Les concepts de base de Java

2.1 Création et instanciation d'une classe

Créer une classe Java exécutable nommée `Test` qui appartient au paquetage `tp1`. La classe `Test` devra afficher `Hello`.

Créer une classe `Personne` **sans constructeur explicite** qui décrit une personne ayant un nom, un prénom, un age et un salaire. Créer les accesseurs correspondants à ces attributs. Vous vérifierez qu'un salaire ne peut pas être négatif. Instancier une personne `p1` dans la classe `Test`, mettre à jour ses informations et les afficher (`Pierre Truc est agé de 30 ans et gagne 2000e`).

Ajouter des constructeurs dans la classe `Personne` qui initialise le nom et le prénom et éventuellement l'age et la profession. Modifier cette classe pour que le nom et le prénom ne puisse plus être modifiés après l'instanciation.

Ajouter un attribut qui indique l'année de naissance et les accesseurs associés. Modifier la méthode que retourne l'age. L'année courante s'obtient avec `java.util.Calendar.getInstance().get(Calendar.YEAR)`.

Ajouter une méthode `comparerSalaire()` qui prend en parametre une personne et retourne -1, 0 ou 1 selon que celle à un salaire supérieur, égal ou inférieur à l'instance courante. Créer une deuxième personne `p2` née la même année que `p1` mais qui gagne plus. Afficher le résultat de la comparaison des salaires de `p1` et `p2`.

Ajouter une méthode `comparerSalaire()` qui prend en paramètre deux personnes et qui retourne -1, 0 ou 1 selon de le salaire de la première est inférieur, égal ou supérieur à celui de la seconde.

Attention `comparerSalaire(Personne)` **et** `comparerSalaire(Personne, Personne)` **sont-elle des méthode de classe ou d'instance ?** Ajouter un attribut de classe `totalDesSalaires` qui indique le total des salaires des personnes et un accesseur. Modifier les méthodes nécessaires pour le salaire total soit maintenu à jour automatiquement.

Afficher le salaire total après la création de chaque personne.

2.2 Les méthodes de base

2.2.1 toString()

Afficher directement l'objet `p1` dans la classe `Test`. Ajouter une méthode : `String toString()` à la classe personne qui retourne une chaine de caractère qui décrit la personne. Exécuter `Test`.

2.2.2 equals()

On souhaite considérer que deux personnes sont " égales " dans notre application, si elles sont nées la même année. Pour définir une relation d'égalite d'objets en Java on redéfinit la méthode `Boolean equals(Object)`. Vérifier que `p1` et `p2` sont égales.

2.3 Définition de classes locales

Pour représenter le fait qu'une Personne à un cerveau ajouter une définition de la Classe Cerveau **à l'intérieur** de la définition de la classe Personne. Ajouter un attribut cerveau à personne et instancier un cerveau à chaque personne lors de sa création. Compiler et exécuter `Test`. Regarder dans le répertoire `build` les classes qui ont été compilées. Essayer de créer une instance de Cerveau diretement dans la classe Test.

3 Pour finir...

Vous allez maintenant créer une classe `Entreprise` qui permet de réprésenter des entreprises qui comportent au maximun Personnes `MAX_EMPLOYES`.

Vous ajouterez les méthodes pour créer une entreprise, ajouter des employés et afficher le nombre d'employé, un employé précis et tous les employés (en utilisant le foreach de Java5).

Les entreprises doivent appartenir à l'une des trois catégories suivantes: {PUBLIQUE, PME, GRAND_GROUPE}. Modifier la classe `Entreprise` pour représenter cela en utilisant un type enuméré et ajouter trois méthodes `isPUBLIQUE()`, `isPME()`, `isGRAND_GROUPE()` qui retournent un booléen.

Ajouter une méthode `miseEnForme()` qui passe le nom de toute les personnes d'une entreprise en majuscule.

Modifier votre classe `Test`, pour l'on puisse lui passer en paramètre le nom et le type d'une entreprise, puis la liste des personnes qui la compose.

```
java Test maBoite PME Pierre Durand 1950 2000 Paul Dupond 1960 3000 Marie Martin 1975 2500
```