Eclipse, javadoc et implémentation de la bibliothèque

L'objectif de ce TP est triple :
- vous familiariser avec l'environnement de développement Eclipse,
- apprendre à générer de la documentation pour vos projets Java grâce à l'outil javadoc.

1 Eclipse : quelques manipulations de base

Eclipse est un environnement de développement très pratique et gratuit, écrit en Java. C'est un environnement dédié à Java, mais aussi à d'autres langages grâce à la notion de plugin que nous n'aborderons pas ici.
Nous vous indiquons ici quelques manipulations de base avec Eclipse. Pour ceux qui connaissent déjà Eclipse, parcourez quand même cette partie et assurez-vous que vous connaissez toutes les manipulations spécifiées.

1.1 Lancer Eclipse

Sous Unix ou Linux : Avant de lancer Eclipse, assurez-vous que vous avez bien une variable d'environnement JAVA_HOME positionnée vers le répertoire d'installation de Java 1.6 (`echo $JAVA_HOME`). Si ce n'est pas le cas, positionnez-là (`export JAVA_HOME=chemin`[1]) et rajoutez cette ligne dans votre .bashrc, ce qui permettra de ne pas avoir à retaper cette ligne à chaque TP.
Lancez Eclipse avec la commande `eclipse &`. Vous êtes invité à préciser votre workspace. Le workspace correspond à l'espace de travail d'Eclipse, c'est en fait un répertoire où seront notamment stockées vos sources. Vous pouvez avoir plusieurs workspaces si vous le souhaitez.

1.2 Création d'un projet

2 solutions :

1. Dans le package explorer (partie gauche de la fenêtre) : clic droit → new → project
2. Dans le menu : File → New → Project

Un wizard s'ouvre.
- Choisissez Java Project (Next)
- Nommez le projet (par exemple TP1). Eclipse va créer dans le workspace un répertoire de ce nom.
- Choisir Create Project in workspace
- Dans la partie intitulée JRE, vérifiez que la JRE est bien 1.6. Sinon il faudra tôt ou tard régler le problème !
- Choisir Create separate source and output folders. Cela signifie que vous souhaitez séparer vos sources (les .java) des .class. Eclipse va créer dans votre projet un répertoire src et un répertoire bin.(Next)
- Vous arrivez dans des configurations plus fines qu'il n'est pas nécessaire de modifier pour l'instant. (Finish)

1. où chemin est le chemin menant au répertoire d'installation de Java

Dans l'explorateur de gauche, on voit le nouveau projet, et dedans un répertoire src. On voit également la référence à la JRE.

1.3 Création d'une classe

Dans l'explorateur de gauche, sur l'icône src, faire un clic droit → new → class. S'ouvre un wizard "New Java Class".
- ne pas changer le source folder qui doit être correct,
- donner le nom du package dans lequel devra être votre classe (par exemple : `tp`),
- nommer la classe (Livre par exemple)
- enlever la superclasse Object : c'est implicite et cela évite de le faire apparaître dans le code
- ne pas toucher à la partie interfaces (c'est inutile pour le moment)
- cocher la case `main` si vous souhaitez mettre un `main` dans la classe.

Celà crée dans l'explorer le package, avec la classe dedans. Dans l'outline (à droite), s'ouvre un explorateur sur la classe faisant apparaître ses propriétés. Au centre, se trouve le code source de la classe. Vous pouvez contrôler ce qui a été généré (notez qu'il y a un embryon d'annotation pour la génération de la documentation dont nous parlerons plus loin). Notez un commentaire TODO qui signale qu'il faut complèter le code de la méthode main. Les TODO sont générés automatiquement ou ajoutés par le programmeur. L'ensemble des tâches à faire peut être visualisé (menu Windows → Show views → Tasks).

1.4 Quelques manipulations du code Java

Vous pouvez maintenant écrire le code de la classe. Vous remarquerez qu'il est directement indenté. Si vous copiez du code non indenté ou si vous voulez rafraîchir l'indentation : sélectionner la partie concernée, clic droit → source → correct indentation (ou crtl +I).

1.4.1 Eclipse corrige quelques erreurs de compilation

Créer un attribut v de type Vector :
- une petite croix apparaît en début de ligne pour signaler un problème (qui est explicité en passant la souris sur la croix)
- une petite lumière signale une solution
- cliquer sur la lumière (clic gauche) et choisir la solution appropriée (ici : import Vector)

1.4.2 Complétion sémantique

Définir une méthode d'ajout d'un élément dans le vector v. Notez que la compilation se fait au fil de la saisie, et vous signale les erreurs en les soulignant en rouge. Quand on saisit un délimiteur ouvrant (ex. { ou "), il ajoute automatiquement le délimiteur fermant. Dans le corps de votre méthode, tapez `v.` : la liste des méthodes définies pour l'objet v est proposée ; il suffit de choisir celle qui nous intéresse en double-cliquant dessus.

1.4.3 Génération des accesseurs et des constructeurs

Créer un attribut public int i. Sur le code, faire un clic droit → source → generate getters and setters. Les attributs sont proposés dans un wizard, où on choisit les attributs et les accesseurs désirés. Notez que si vous générez les accesseurs sur v, ces accesseurs n'ont pas vraiment d'intérêt.
On peut générer des constructeurs à partir des attributs. Sur le code, faire un clic droit → source → generate constructors using fields. Sélectionner dans le wizard qui apparaît les champs que l'on veut passer en paramètre au constructeur. Pour l'instant, cocher la case *omit call to default constructor super()*.

1.4.4 Renommage

Il existe une fonction de renommage des attributs et méthodes, classes, etc. Par exemple, on peut renommer v. pour cela, dans l'outline, on fait un clic droit sur l'élément (v) → refactor → rename. Cela renomme toutes les occurrences de l'élément dans le code.

1.5 Exécution d'un programme

Pour exécuter le programme (qui contient un `main`), choisir menu run → run as → java application (ou utiliser l'icône avec la flèche verte).

1.6 Utilisation de classes hors projet Eclipse

Pour utiliser des classes qui sont en dehors du projet et en dehors d'une archive .jar, c'est un peu compliqué. Dans le projet, faire un clic droit → *Properties* → *Java Build Path.* Sélectionner l'onglet *Librairies* et choisir *Add Class Folder*, puis *Create New Folder.* Choisir un nom pour le répertoire puis *Advanced.* Là, cocher l'option *Link to Folder in the file system* et choisir le répertoire parent contenant celui du package, puis terminer par OK.

2 La documentation

Vous pouvez vous créer des documentations au format html pour vos programmes. Pour cela, il suffit de commenter votre code à l'aide d'un format de commentaire particulier, puis d'utiliser l'outil javadoc fourni avec le JDK. La documentation de la javadoc est disponible à cette adresse :
http ://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/javadoc.html.

2.1 Formattage des commentaires

On peut ajouter des commentaires de documentation dans le code source, juste au dessus de n'importe quelle classe, interface ou méthode, et n'importe quel constructeur ou attribut.

Format des commentaires :

```
/**
 * Commentaire classique d'une documentation simple
 * qui s'étend sur deux lignes
 */
```

On peut aussi n'utiliser qu'une seule ligne :

```
/** This comment takes up only one line. */
```

Les commentaires placés dans le corps des méthodes sont ignorés. On met un seul commentaire par élément à commenter.
Un commentaire est composé d'une description principale suivie de tags. La description principale commence à partir du délimiteur /** et continue jusqu'à la section de tags.
Voilà un commentaire bien formé :

```
/**
 * Ceci est la description principale de ce commentaire.
 * @see java.lang.Object
 */
```

Il y a 2 sortes de tags, les tags en ligne et les tags en bloc. Les tags en bloc apparaissent sous la forme @tag et les tags en ligne sous la forme {@tag}. Les tags en bloc apparaissent en début de ligne de commentaire (après une étoile) dans une section de tags alors que les tags en ligne apparaissent ... en ligne, c'est-à-dire au milieu d'une ligne de commentaire.
Exemple de tag en ligne (@link) et en bloc (@deprecated) :

```
/**
 * @deprecated  As of JDK 1.1, replaced by {@link #setBounds(int,int,int,int)}
 */
```

Les commentaires sont écrits en HTML. Par exemple :

```
/**
 * This is a <b>doc</b> comment.
 * @see java.lang.Object
 */
```

La première phrase de chaque commentaire doit être une phrase de résumé pour l'entité commentée. Cette première phrase se termine avec le premier point rencontré. Elle apparaîtra dans la partie résumé de la page de documentation.

2.2 Les principaux tags

- @author name : permet de spécifier l'auteur du code. On peut mettre plusieurs tags @author pour un même élément, ou séparer les noms par des virgules.
- @code duCode : permet que le code apparaisse avec un format de code et pas de texte normal. De plus, si du code ressemble à une balise HTML, la balise n'est pas interprétée.
- @docRoot : permet de référer la page racine de la documentation.
- @deprecated : permet de signaler des éléments qui ne devraient plus être utilisés (classique lors d'une évolution de logiciel)
- @exception nomClasse description : permet de spécifier les exceptions levées. Synomyme de @throws.
- @link package.class#member label : permet de référer un autre élément de documentation.
- @linkplain : idem que @link mais le label est en texte simple et pas en police de code.
- @literal texte : permet d'écrire du texte non interprété comme du HTML.
- @param nomParam commentaire : permet de documenter les paramètres des méthodes et constructeurs.
- @return commentaire : permet de documenter ce qui est retourné par une méthode.
- @see reference : comme @link, mais au lieu d'être en ligne, celà ajoute dans la documentation générée une section `see also` avec le lien.
- @since texte : permet de préciser depuis quelle version l'élément existe.
- @throws nomClasse description : synonyme de @exception
- @version texte : permet de spécifier la version.

2.2.1 L'outil javadoc

Pour générer la documentation, on appelle simplement l'outil javadoc avec en paramètre le fichier java à traiter (ou *.java ..). Il existe bien sûr des options permettant de placer la documentation où on le souhaite, etc. Par défaut, la documentation est placée dans le répertoire courant. Il est donc recommandé d'utiliser l'option -d qui permet de spécifier où générer la documentation.

2.3 Javadoc et Eclipse

Eclipse vous assiste dans l'écriture des commentaires : à la création d'éléments à l'aide de wizards, ou quand vous tapez /** puis entrée au dessus d'un élément, Eclipse vous génère les tags qui lui semblent utiles.
Eclipse vous assiste aussi dans l'écriture de tags : quand vous tapez @ dans un commentaire de documentation, la liste des tags disponibles s'affiche, il n'y a plus qu'à choisir.
Vous pouvez générer la documentation d'un projet via project →generate javadoc.