INFORMATIQUE INDUSTRIELLE

Licence SEICOM – I.U.T. de Nantes

TP 1 : Initiation à la programmation orientée objet avec l'atelier de développement Eclipse

Module M2-1 édition 2012

1 OBJECTIFS
Etre capable de : Repérer, dans une application existante, les objets utilisés ;
Manipuler un objet, ses différents membres en prenant en charge leur encapsulation ;
Définir et utiliser un nouvel objet instancié d'une classe existante ;
Comprendre l'utilisation de techniques associées aux objets : agrégation.
Utiliser java en ligne de commande.

Q.1. Les travaux à faire apparaitre dans votre CR seront repérés ainsi.

2 La classe Etudiant
Lancer un terminal et vérifier que le JDK est accessible. Si non, utiliser le script "CMD_avec_JDK_Java.bat" dans Y:\CommunGEII\Licence SEICOM.

Q.1. Comment avez vous testé la présence du JDK ?
Javac ? commande non reconnue
lancement .bat ? javac reconnue

Copier les fichiers du TP1 (Etudiant.java, html...) dans votre répertoire de travail.
Q.2. À partir de la documentation de la classe Etudiant, donnez la représentation UML de cette

Classe Etudiant
-int nbAbs -String nom -String prenom +Etudiant()
+Etudiant(String n, String p) +ajouteAbsence() +equals(Etudiant e):bool +fullName():String +getNbAbs():int +justifierAbsence() +shortName():String +toString():String

Quelle est la classe mère de la classe Etudiant ?
La classe object

Ouvrez le fichier Etudiant.java avec un éditeur de texte (Blocnote, Notepade++,...).
Compilez la classe Etudiant. Executez le main() de cette classe.
Q.3. Donnez les commandes DOS utilisées.
Javac Etudiant.java
java Etudiant

Retirez (=mettre en commentaire) la méthode "toString()" de Etudiant. Ré-éxécutez le main().
Q.4. Expliquez la différence d'affichage en vous aidant de la documentation de Object.toString().
Lien util :http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html#toString%28%29
la méthode transforme une chaine de caractère en une autre. Additionne 2 chaines ici les initiales

Q.5. Dans la méthode main(), expliquez la ligne "System.out.println(p2);" en vous aidant des pages de la documentation de System.out.println(Object) et String.valueOf(Object).

Liens utils:
System.out.println(Object)->
http://docs.oracle.com/javase/7/docs/api/java/io/PrintStream.html#println%28java.lang.Object%29
String.valueOf(Object)->
http://docs.oracle.com/javase/7/docs/api/java/lang/String.html#valueOf%28java.lang.Object%29

Affiche à l'écran l'objet p2 : fait appel d'abord à la méthode valueOf(p2) qui fait appel a la méthode toString de la classe object.

Q.6. Expliquez la différence entre le résultat du "==" et du ".equals" (lignes 96 à 98 du code)entre les étudiants p2 et p3.
== compare les 2 adresses des objets qui sont différentes et renvoi donc false alors que .equals compare le contenue des 2 objets qui sont identiques et renvoi donc true.

En comparant votre diagramme de classe et le code fourni, vous remarquerez qu'il manque un attribut "nbAbs" représentant le nombre d'absences d'un étudiant.
Q.7. Ajouter cet attribut et modifez les constructeurs et les méthodes ajouteAbsence(), getNbAbs() et justifierAbsence().
Cf code

Lors de l'affichage d'un étudiant (grâce à System.out.println(.) par exemple), nous aimerions avoir ce type de sortie
si l'étudiant a des absences : [PJ] Patrick Jane (2 abs)
et s'il n'en a pas : [PJ] Patrick Jane

Q.8. Modifier la fonction toString() pour obtenir le bon format.
Cf code

3,La classe Module

Un module est représenté par son nom, son sigle, un nombre d'étudiants maximum pouvant assister au cours. La documentation complète de cette classe ainsi qu'un début d'implémentation vous sont fournis.

Q.9. En vous aidant des explications sur tableaux dans le poly de cours (slide 86) compléter le constructeur de la classe Module.
Cf code

Q.10. Completer la classe Module pour avoir toutes les méthodes décrites dans la documentation Module.html disponible.
Cf code

Q.11. Tester avec le main() existant.
OK

On aimerait ajouter ce scénario ci-dessous dans le main() de la classe Module (en plus des tests existants) :
• creer un autre module POO / M21 avec 20 étudiants au max
• ajouter john et bob au module
• afficher M21
• bob a une absence de plus ! (=> lui ajouter une absence)
• ré-afficher les modules M14 et M21
Q.12. Compléter le main en conséquence.
Cf code

Q.13. Expliquer pourquoi les absences de bob apparaissent dans le module M14 et dans le module M21.

Car c'est le même objet « bob » de type Etudiant qui est affiché dans le module m14 et m21.(Il n'y a pas 2 objets « bob » qui ont été créés)

Q.14. Modifier la fonction ajoute(Etudiant) pour ne pas qu'un même étudiant puisse s'inscrire deux fois au même module. Faut-il modifier quelque chose dans la classe Etudiant ? Si oui faites le bien sûr !
Non il n'est pas nécessaire de modifier quelque chose dans la classe Etudiant, Une modification dans la méthode ajoute() suffit. Il existe sans doute d'autre façon de faire.