TP de Programmation Orientée Objet N°3
ILOG1 *

5 janvier 2005

1 Pour les débutants en programmation par contrat...

L'objectif essentiel de ce TP est de se familiariser avec l'environnement JML pour la programmation par contrat et de définir complètement une classe (y compris assertions et commentaires).

1.1 Travail à faire pendant la séance de TP

1.1.1 Utilisation de l'environnement JML

1. Téléchargez le fichier `PileSimple.tgz` à l'adresse `http://marc.champesme.free.fr/POO/source/JML/PileSimple.tgz`.
2. Décompressez/désarchivez le fichier `PileSimple.tgz` à l'aide de la commande :
   `tar xvzf PileSimple.tgz`
   Vous devez normalement obtenir dans le répertoire courant, un nouveau répertoire nommé `PileSimple` et contenant les deux fichiers `PileInt.java` et `PileTest.java`. Placez vous dans le répertoire `PileSimple` à l'aide de la commande : `cd PileSimple`.
3. Compilez les deux classes avec la commande `javac`, puis exécutez le programme à l'aide de la commande `java` (note : ce programme contient un bug, c'est fait exprès !). Notez le message d'erreur d'affiché par le programme et, en vous aidant des informations contenues dans ce message d'erreur, repérer dans les classes `PileInt` et `PileTest` les méthodes en cours d'exécution au moment où l'erreur s'est produite. En fonction des informations recueillies, déterminez précisément l'événement qui a déclenché l'interruption du programme.
4. Effacez les fichiers `.class` (`rm *.class`) présents dans le répertoire courant et refaites exactement les mêmes actions qu'à la question précédente en utilisant cette fois :
   – la commande `jmlc` (à la place de `javac`) pour compiler
   – la commande `jmlrac` (à la place de `java`) pour exécuter le programme

   Note : les commandes `jmlc` et `jmlrac` admettent exactement les mêmes options que les commandes standards qu'elles remplacent (`javac` et `java`).
5. En fonction des observations faites précédemment, proposez une correction du programme.
6. Modifier le programme (les assertions de la classe PileInt ne doivent pas être modifiées) afin que l'exécution du programme déclenche une violation de post-condition ou une violation d'invariant. Décrivez précisément ces modifications en indiquant l'assertion provoquant l'interruption du programme (le cas échéant, indiquez la méthode à laquelle se rapporte cette assertion).
7. Générez la documentation incluant les assertions de la classe `PileInt` à l'aide de la commande :
   `jmldoc -author -d api PileInt.java`

*M. Champesme – Département d'Informatique – Institut Galilée

Note : la commande `jmldoc` admet exactement les mêmes options que la commande standard qu'elle remplace (`javadoc`). Les options `-author -d api` sont donc aussi utilisables avec `javadoc` : l'option `-author` sert à faire apparaître dans la documentation produite, le contenu du champ `@author` spécifié dans le commentaire de la classe et l'option `-d api` permet de placer les fichiers html produits dans le répertoire spécifié (dans ce cas, le répertoire nommé `api`).

1.1.2 Définition complète d'une classe

Définissez une version "mutable" (classe `MutPoint`) et une version "immutable" (classe `ImmutPoint`) de la classe `Point` décrite dans le sujet de TD (y compris les méthodes `equals`, `clone`, `hashCode` et `toString` et y compris l'ensemble des assertions pour toutes les méthodes). Définissez ensuite une classe `TestPoint` permettant de tester ces deux classes.

1.2 Travail à rendre

- Les réponses aux questions posées dans la section 1.1.1.
- Les trois fichiers : `MutPoint.java`, `ImmutPoint.java` et `TestPoint.java`. Les deux classes `MutPoint` et `ImmutPoint` doivent être entièrement commentées et contenir toutes les assertions que vous jugerez appropriées.

2 Pour ceux qui connaissent déjà la programmation par contrat...

2.1 Travail à faire pendant la séance de TP

L'objectif de ce TP est de définir complètement (i.e. y compris assertions et commentaires) plusieurs classes interagissant entre elles et avec deux classes données.

En utilisant les 2 classes `Dessin` et `Polygone` dont le code source est disponible à l'adresse : `http://marc.champesme.free.fr/POO/coursTD/LicenceILOG12004-2005/TP3/` définissez les classes `Point`, `Segment` et `Cercle` de manière cohérente avec, d'une part, l'énoncé du TD[1] et, d'autre part, les classes `Dessin` et `Polygone` (vous ne devez apporter aucune modification à la classe `Polygone`). Modifiez la classe `Dessin` de façon à pouvoir tester l'affichage de segments de polygones et de cercles.

2.2 Travail à rendre

Les fichiers : `Point.java`, `Segment.java`, `Cercle.java` et `Dessin.java`.

3 Note sur le travail à rendre

- Même lorsque le travail a été fait en groupe, chaque étudiant doit rendre sa propre version du travail demandé.
- Pour chaque classe inscrivez votre nom dans le commentaire de la classe en utilisant le marqueur javadoc `@author`.
- Le travail demandé doit être rendu de préférence, par courrier électronique à l'adresse `mailto:Marc.Champesme@lipn.univ-paris13.fr` en précisant dans le sujet (ou objet) du message : "POO" (pour programmation orientée objet), "CR TP3" (pour compte rendu du TP n°3) et votre nom.

[1] y compris les méthodes `equals`, `clone`, `hashCode` et `toString` et y compris l'ensemble des assertions pour toutes les méthodes