Devoir de programmation objet
DEUG MIAS 2ème année

Marc Champesme
Marc.Champesme@lipn.univ-paris13.fr
Departement d'Informatique
Institut Galilée

18 mai 2004

Tous les devoirs doivent être remis à votre chargé de TD en main propre, dans le casier DEUG MIAS 2ème année à l'entrée du couloir B300 ou par e-mail à l'adresse mailto:marc.champesme@lipn.univ-paris13.fr, **au plus tard le vendredi 4 juin 2004**. Tout devoir rendu après cette date sera considéré comme non fait.

1 Objectif du devoir

Le but est de mettre en pratique les concepts de la programmation orientée objet et plus particulièrement :

- implémenter des classes en respectant rigoureusement leur interface (i.e. commentaires et contrat associé à chaque classe) ;
- utiliser correctement l'héritage et le polymorphisme ;
- écrire les commentaires d'une classe.

2 Présentation du sujet

Il s'agit d'implémenter un ensemble de classes représentant des figures géométriques simples, en respectant l'interface donnée pour ces classes. L'interface des classes à implémenter est consultable à l'adresse :
http://marc.champesme.free.fr/POO/devoir/MIAS2-2003-2004/classesGeo/jmldoc/
et peut-être téléchargée à l'adresse :
http://marc.champesme.free.fr/POO/devoir/MIAS2-2003-2004/classesGeo.jmldoc.tgz

3 Travail demandé

Le devoir doit être réalisé individuellement : chaque étudiant devra rendre sa propre version du logiciel ainsi qu'un rapport décrivant le travail réalisé.

3.1 Corriger les assertions des classes

Les assertions des classes, telles qu'elles sont données, contiennent des erreurs qui rendent impossible d'en donner une implémentation conforme. Parmi celles-ci certaines sont décrites en annexe **??**. Votre première tâche sera d'apporter les corrections nécessaires aux assertions des classes à implémenter. Chaque correction apportée aux assertions des classes devra être décrite dans le rapport que vous rendrez en accompagnant cette description d'une référence explicite à l'annexe **??**.

D'autres erreurs non décrites dans l'annexe existent très probablement, si vous en trouvez il vous est possible de les corriger, sous réserve que vous décriviez très précisément en quoi consiste ces erreurs, quelles classes et méthodes sont concernées, et que vous justifiiez en quoi ces erreurs rendent l'implémentation impossible.

En dehors de ce cadre, toute modification non justifiée des assertions sera considérée comme une erreur.

3.2 Implémenter les classes

Vous devez donner une implémentation complète et conforme aux assertions pour chacune des méthodes de chaque classe, à l'exception de la classe `GMath` dont l'implémentation vous est fournie.

Votre implémentation sera jugée correcte, si l'exécution d'un programme de test arbitraire (mais satisfaisant toutes les pré-conditions) ne provoque aucune violation d'assertion.

> Aucune méthode ne doit être ajoutée dans les classes à implémenter : la documentation jmldoc des classes implémentées doit-être strictement identique à la documentation fournie (sous réserve des corrections apportées aux assertions dans le cadre précédemment défini à la section **??**).

3.3 Commenter la classe `PointGeo`

Vous devrez aussi écrire des commentaires complets pour la classe `PointGeo`, en décrivant de manière aussi précise que possible le rôle et le fonctionnement de cette classe et de chacune de ses méthodes. Vos commentaires devront, en particulier, s'efforcer de décrire avec des mots aussi précis que possible les informations fournies par les assertions (vos commentaires devront donc être plus détaillés que ceux qui sont donnés pour les autres classes).

4 Bonus

En plus du travail demandé ci-dessus, jusqu'à 4 points de bonus (i.e. hors barème) pourront être attribués pour la réalisation d'une (ou plusieurs) des tâches décrites ci-après.

4.1 Affichage des figures

Réaliser un programme permettant d'afficher les figures. Pour ce travail, vous pourrez vous aider du logiciel disponible à l'adresse :

`http://marc.champesme.free.fr/POO/source/LogicielDessin.tar.gz`

Ce travail sera évalué selon les critères suivant :

- affichage de l'ensemble des figures (y compris des figures ayant subies des rotations, translations ou homothéties)
- affichage conforme aux propriétés géométriques des formules
- qualité du code : commentaires, assertions, respect des consignes sur la visibilité des variables...

4.2 Réalisation d'une interface graphique

Réalisation d'un logiciel interactif de dessin basé sur les classes implémentées. Pour ce travail, vous pourrez vous inspirer des travaux d'étudiants disponibles aux adresses :

`http://marc.champesme.free.fr/POO/devoir/MIAS2-2003-2004/devoirEtudiant1.src.tgz`

et

`http://marc.champesme.free.fr/POO/devoir/MIAS2-2003-2004/devoirEtudiant2.src.tgz`

Ce travail sera évalué selon les mêmes critères que le travail précédent en prenant en compte, en plus, la richesse en fonctionnalités et l'ergonomie du logiciel (i.e. facilité d'utilisation).

5 Evaluation

La qualité de la réalisation sera évaluée selon les critères suivants :

- Complétude : l'ensemble des méthodes et des classes ont-elles été implémenter ?
- Conformité par rapport aux assertions : l'implémentation est elle conforme au contrat ? La conformité de votre implémentation sera évaluée à l'aide de programmes de test permettant de tester chaque méthode sur des données représentant un large éventail de cas ;

- Qualité des commentaires de la classe `PointGeo` ;
- Qualité du rapport rendu ;
- Qualité des corrections apportées aux assertions : pour les erreurs non mentionnées an annexe **??**, une attention particulière sera portée à l'argumentation justifiant qu'il s'agit bien d'une erreur d'assertion.

6 Documents à rendre par chaque étudiant

Chaque étudiant rend le code source des classes implémentées et un rapport tel que décrit ci-dessous :

Votre rapport doit être présenté et rédigé de manière à permettre une compréhension rapide, complète et précise de l'ensemble du travail réalisé. Ce rapport devra contenir en particulier :

- Pour chaque classe, la liste complète de toutes les méthodes dont l'implémentation est demandée en mentionnant pour chacune si :
  - l'implémentation est complète
  - l'implémentation a été faite mais n'est pas conforme aux assertions (précisez autant que possible la raison pour laquelle vous n'avez pas réussi à donner une implémentation correcte, en particulier si vous pensez qu'il s'agit d'une erreur dans les assertions)
  - l'implémentation n'a pas été faite
- Pour chacune des erreurs mentionnées en annexe **??**, une description complète et exhaustive de toutes les modifications apportées aux assertions en précisant bien chaque classe et chaque méthode dont les assertions ont été modifiées pour corriger cette erreur.
- Pour chacune des erreurs portant sur les assertions et non mentionnées dans l'annexe **??**, une description précise, détaillée et argumentée de la raison pour laquelle il s'agit d'une erreur d'assertion et des modifications apportées aux assertions afin de la corriger.

A Erreurs à corriger dans les assertions des classes

La correction des erreurs décrites ci-dessous peut nécessiter, pour chaque erreur, des modifications dans plusieurs classes.

A.1 Classe `GMath`

La méthode `GMath.epsEquals(...)` n'est pas adaptée (i.e. test trop strict) quand les valeurs à comparer sont inférieures à `epsilon`.

A.2 Sous-classes de `Figure` autres que `Segment`

Les sous-classes de `Figure` autres que `Segment` permettent de définir des figures dont des points caractéristiques successifs sont identiques, alors que la classe `Segment` impose que les deux points d'un segment soit distincts. Cela rend impossible l'implémentation de certaines méthodes (par exemple, la méthode `getSegment(int)` de la classe `LignePolygonale`)

A.3 Classe `PointGeo`

Les post-conditions de la méthode `homothetie(...)` de la classe `PointGeo` sont contradictoires dans le cas ou `(k == 1)` et `this.equals(centre)`.

A.4 Méthodes à paramètres de type `double`

Les variables de type `double` peuvent prendre la valeur spéciale `Double.NaN` (i.e. “Not a Number”). Les assertions des méthodes à paramètres de type `double` ne prennent pas en compte cette possibilité ce qui rend leur implémentation impossible du fait des propriétés très particulière de cette valeur (par exemple, l'expression `Double.NaN == Double.NaN` est toujours fausse).

A.5 Classe `Segment`

Il manque un parenthesage sur la 2eme post-condition de la méthode `epsDroiteContient(...)`.

Pour la méthode `estEpsPerpendiculaireA()`, si un segment est exactement vertical (resp. horizontal) et l'autre approximativement horizontal (resp. vertical), alors, dans ce cas, les segments ne sont pas reconnus comme approximativement perpendiculaires. Il manque par conséquent la possibilité de pouvoir évaluer si un segment est approximativement horizontal (resp. vertical).

A.6 Classe `Polygone`

Pour le constructeur prenant un paramètre de type `LignePolygonale`, il manque une pré-condition permettant d'établir l'invariant : `getNbPoints() >= 3`