VERSION ENSEIGNANT

Programmation dirigée par les tests (TDD : Test-Driven Development)

Exercice 1

Question 1 Chaque groupe de deux étudiants doit traiter les sujets dans la liste ci-dessous. Prendre deux sujets avec un étudiant dans le rôle du testeur et l'autre programmeur, prendre les deux autres sujets en inversant les rôles des étudiants.

1. Ecrivez un programme permettant de calculer toutes les racines carrées des nombres compris entre A et B, A et B étant deux nombres entiers tels que $A < B$. Le résultat doit etre retourner dans un tableau d'entiers.
   (Etudiant 1 = Programmeur ; Etudiant 2 = Testeur)
2. Ecrivez un programme permettant de retourner une matrice de taille $M \times N$ remplie par des nombres aléatoires compris entre A et B. Les valeurs M, N, A et B doivent être passées en paramètres.
   (Etudiant 1 = Testeur ; Etudiant 2 = Programmeur)
3. Ecrivez un programme permettant de calculer le nombre de mots palindrômes dans une phrase passée en paramètre. Un palindrôme est un mot que l'on peut lire dans les deux sens.
   (Etudiant 1 = Programmeur ; Etudiant 2 = Testeur)
4. Ecrivez un programme qui, prend en entrée trois variables a,b et c qui représentent les côtés d'un triangle, et retourne :
   - valeur 1 pour **Triangle isocèle**.
   - valeur 2 pour **Triangle équilatéral**.
   - valeur 3 pour **Triangle scalène**.
   - valeur 4 pour **pas un triangle**.

   (Etudiant 1 = Testeur ; Etudiant 2 = Programmeur)

Programmeur

1. Écrivez le squelette de la classe principale, et commentez-le.
2. Écrivez le code des méthodes de la classe principale, et commentez-le.

Testeur

1. Écrivez l'intégralité de la classe test. Cette classe doit comprendre :
   - Des assertions comme vu précédemment en S3.
   - Des tests vérifiant que les exceptions sont bien levées quand elles doivent l'être (par exemple : de mauvais paramètres sont passés dans l'appel).
   - Des tests vérifiant que les boucles s'effectuent dans des temps raisonnables.
   - Formatez correctement les sorties de vos tests en utilisant les annotations `@Before` et `@After`.

1 Score

	Etudiant 1	**Etudiant 2**
Programme 1		
Programme 2		
Programme 3		
Programme 4		

Exercice 2

On considère un terrain caractérisé par une hauteur et une largeur (classe `Field`) sur lequel se trouvent un certain nombre d'objets possédant un poids et des coordonnées (classe `FieldObject`). Certains des objets peuvent se déplacer dans les limites du terrain en dépensant du carburant (classe `Moveable`). Parmi les objets mobiles, on distingue des robots (classe `Robot`), qui peuvent transporter ou déposer les autres objets, dans la limite d'une certaine charge totale.

Objectif

Dans ce TP, nous vous fournissons les squelettes de ces quatre classes, que vous devez compléter :

http://www.lirmm.fr/ lazaar/ENS/IUT/TP-Junit.zip

Nous ne dirons rien de plus sur le programme dans ce sujet. En revanche, nous vous fournissons un jeu de 83 tests Junit, qui vous montrent les comportements attendus, et qui devront vous guider dans l'écriture du code.

L'objectif est de compléter les méthodes du squelette de sorte que tous les tests réussissent.

Indications

Le squelette fourni contenant de l'héritage, il peut être nécessaire de redéfinir certaines méthodes dans les sous-classes.

Nous vous suggérons de compléter les classes dans l'ordre `Field` , `FieldObject` , `Moveable` , `Robot` .

La méthode la plus délicate est la méthode `goTo` , il peut être judicieux de ne s'y frotter qu'une fois à l'aise avec la mécanique de ce TP.

Mise en route

Sous Eclipse, créez un nouveau projet java, et importez-y les fichiers du répertoire TP-Junit/ fourni.