IUT de Paris Descartes – Base de la Programmation Objet

Travaux Pratiques – Sujet n°3

Objectifs

- Définir des attributs de classe
- Définir des constructeurs
- Définir et vérifier des préconditions
- Développer des tests unitaires

Une course de relais 4×100m

Nous devons simuler une course de relais 4 x 100m. Nous commençons par développer une classe `Athlète` représentant un participant d'un relais.

Pour simplifier, un athlète sera identifié uniquement par son numéro de dossard. Bien entendu deux athlètes distincts ne doivent pas porter le même numéro de dossard.

1. Créez une classe `Athlète` disposant d'un constructeur sans paramètre. Faites en sorte que le numéro de dossard de l'athlète soit déterminé automatiquement. Complétez la classe par une méthode `toString()`.
2. Étant encore au premier stade du développement, la mise en place de tests unitaires va nous permettre de détecter les erreurs au plus tôt.

   Avant tout chose, vous devez configurer *eclipse* pour que les clauses `assert` soient effectivement testées lors de l'exécution des tests (cela pourra nous être utile plus tard). Sélectionnez `Fenêtre` → `Préférences` → `Java` → `JUnit` dans le menu et cochez l'option '`Add -ea to VM arguments ...`'. Notez bien que cette configuration est valable pour tous les projets de votre espace de travail.

   Créez un nouveau cas de test `JUnit` (`Fichier` → `Nouveau` → `JUnit Test Case`). Dans la boîte de dialogue qui s'affiche, assurez vous que la version 4 de `JUnit` est sélectionnée. Normalement, *eclipse* propose un nom par défaut. Si ce n'est pas le cas, nommez votre classe `AthlèteTest`. Assurez vous que la classe devant être testée est bien la classe `Athlète`. La boite de dialogue suivante, vous propose de sélectionner les méthodes que vous voulez tester. Choisissez le constructeur. Étudiez le code produit par *eclipse* et, en particulier, le '`import static`' ainsi que la directive `@Test`.

   À présent, vous devez remplacer l'instruction ' fail ("Not yet implemented")' par des instructions de test. Vous vérifierez (en employant `AssertTrue` ou `AssertFalse`) que les dossards de deux athlètes nouvellement créés sont distincts (les chaînes de caractère les représentants doivent être différentes). Exécutez le cas de test et observez les résultats. Proposez un deuxième cas de test, vérifiant que dix athlètes nouvellement construits sont distincts deux à deux.

   La documentation de la classe `org.junit.Assert` peut vous être utile (`http://junit.org/javadoc/latest/org/junit/Assert.html`)
3. Lors d'une course 4×100m, les participants doivent se transmettre un témoin. Ajoutez un champ `témoin` indiquant si l'athlète dispose du témoin. Modifiez le constructeur en conséquence et ajoutez les méthodes `prendLeTémoin()`, `rendLeTémoin()`, `aLeTémoin()` ainsi que `passeLeTémoinA(Athlète a)` dont le rôle est de transmettre le témoin à l'athlète `a`. Pour chacune de ces méthodes, déterminez ses préconditions et vérifiez-les en employant `assert`.

4. Ajoutez un cas de test relatif à la prise du témoin. Vous vérifierez qu'un athlète nouvellement créé ne dispose pas du témoin, qu'il en dispose effectivement si on lui donne et que le passage de témoin est effectif.
5. La course d'un athlète doit être simulée. Le temps de course sera tiré aléatoirement entre 9 et 14 secondes. Ajoutez un champ `dernierTemps` stockant le dernier temps de course de l'athlète ainsi qu'une méthode `court()` lui affectant une valeur aléatoire (la méthode statique `Math.random()` retourne un réel aléatoire compris entre 0 et 1) et un accesseur à cette valeur. Pensez à modifiez le constructeur en conséquence et à vérifier les préconditions là où c'est nécessaire. En particulier, un athlète ne peut courir que s'il dispose du témoin.
6. Ajoutez un cas de test relatif à la course d'un athlète. Vous vérifierez que son temps de course est bien compris entre 9 et 14 secondes.
7. Les athlètes d'une même équipe sont ordonnés de façon à se passer le témoin un à un. Pour ce faire, chaque athlète doit connaître celui à qui il doit passer le témoin. Ajoutez un champ `suivant` désignant cet athlète, modifiez le constructeur si nécessaire et ajoutez une méthode permettant de positionner la valeur de ce champ.
8. Dans notre simulation, les athlètes ont des réflexes un peu surprenants. En effet, dès qu'il reçoit le témoin, un athlète se met immédiatement à courir. Après sa course, il transmet aussitôt le témoin à l'athlète suivant s'il en connaît un. Dans le cas contraire, il rend le témoin. Modifiez la méthode `prendLeTémoin()` en conséquence et relancez la vérification des tests.
9. Le test associé à la méthode de la prise du témoin échoue. Modifiez cette méthode de façon à ce qu'un athlète n'ayant pas de suivant dans le relais, conserve le témoin après sa course. Relancez l'exécution des tests et assurez vous que les résultats sont positifs.
10. Imaginez et programmez de nouveaux cas de test pour la classe.