Utilisation de JUnit pour tester les dictionnaires

JUnit est installé par défaut dans Eclipse.

1 Première classe de test simple pour `OrderedDictionary`

Nous allons ici mettre en place une première classe de test simple pour tester la classe `OrderedDictionary`.

1.1 Création de la classe de test

Dans votre projet :
- clic droit → new JUnit test case
- Choisissez bien JUnit 4 et pas JUnit 3
- Nommez votre classe de test. Cochez la case permettant de générer la méthode `setUp`. Ne sélectionnez aucune autre option.

1.2 Environnement de test

Dans la classe test nouvellement créée, déclarez un attribut de type `OrderedDictionary`. Créez-le dans la méthode `setUp` qui a été générée. Vous noterez que cette méthode est annotée par l'annotation `@Before`, c'est donc une méthode de préambule qui sera appelée avant chaque méthode de test.

1.3 Une première méthode de test

Ajoutez à votre classe une première méthode de test `testAddOneElementToEmptyDico()`. N'oubliez pas d'ajouter l'annotation `@Test` à cette méthode, afin qu'elle soit bien considérée par `JUnit` comme une méthode de test. Dans le corps de cette méthode, écrivez le code permettant d'ajoutez le couple (clef-valeur) de votre choix. Puis, vérifiez que tout s'est bien passé :
- Vérifiez que la taille du dictionnaire est bien 1 (par une assertion du type `AssertEquals(1, dico.size()`)
- Vérifiez que l'élément ajouté existe bien dans votre dictionnaire (par une assertion du type `AssertTrue(dico.containsKey(''clef''))`)
- Vérifiez que vous pouvez bien retrouver l'élément ajouté dans votre dictionnaire.

Exécutez votre test (`run as JUnit application`).

1.4 Fin de la classe de test pour OrderedDictionary

Complétez le test de la classe `OrderedDictionary` en ajoutant autant de méthodes de test qu'il vous semble nécessaire. Corrigez les erreurs que vous détecterez au fur et à mesure.

2 Test des autres dictionnaires

Testez les 2 autres types de dictionnaires. Vous vous rendrez compte que de nombreuses méthodes de test sont identiques à celles de la classe de test pour `OrderedDictionary`. Il pourra être intéressant de créer une super-classe encapsulant ces tests.

Finalement, créez une suite de test permettant d'enchaîner l'exécution des tests des 3 classes de test.

3 Couverture de code

Quand vous pensez avoir terminé vos tests, ré-exécutez-les, cette fois avec `run as Coverage Application`. `Coverage` est un outil provenant du plugin Eclipse `ECLEmma`, qui permet d'analyser le code couvert par les tests, c'est-à-dire de déterminer quelles parties du code sont exécutées lors de l'éxécution des tests, et quelles parties ne le sont pas. On obtient une mesure de la couverture de code. Attention, avoir 100% de couverture de code ne signifie pas nécessairement d'avoir correctement testé votre code.

Observez les parties de votre code couvertes et non couvertes. (Re-)Testez en conséquence.

4 Test d'une autre version de dictionnaire

Utilisez maintenant vos tests pour tester une autre version de dictionnaires que la vôtre, par exemple celle disponible sur la page des supports de cours de cette UE :
http ://www.lirmm.fr/ dony/enseig/IL/TP-1-2-Atester .