Les tests unitaires

1. Généralités

Un test permet de vérifier un programme par exécution de celui-ci.

Il est important de tester les programmes pour détecter au plus tôt les dysfonctionnements. Plus un dysfonctionnement est réparé tardivement plus sa correction est coûteuse.

On peut bien sûr faire des tests manuels mais ils sont long et fastidieux à produire et par conséquent ils sont souvent négligés. Il convient donc d'automatiser ces tests.

Dans ce document, ce sont les tests unitaires qui sont abordés. Le rôle des tests unitaires consiste à tester une petite partie d'une application. En programmation objet, on peut considérer que l'unité à considérer est la classe.

2. Mise en œuvre des tests unitaires

On peut bien sûr écrire les tests unitaires après avoir écrit le code des classes, mais il est possible d'écrire les tests unitaires en même temps que l'écriture de l'application voire même avant.

Par ailleurs, les tests unitaires apportent aux personnes qui sont amenées à lire le code d'une application des informations sur la manière dont doit fonctionner le code. On peut donc considérer les tests comme faisant partie de la documentation.

À chaque fois que l'on modifie son code, il faut prendre soin de refaire les tests pour voir si des régressions sont apparues.

Une régression est un problème qui survient après un changement dans le code source. Ce changement peut être dû à la correction d'un bug, à l'ajout d'une nouvelle fonctionnalité, à un réusinage du code *(refactoring)*, etc.

3. Le *framework* JUnit 4

JUnit est un *framework* (un *framework* est un ensemble de bibliothèques, de conventions, de manières de travailler, etc.) de rédaction et d'exécutions de tests unitaires. JUnit appartient à la famille xUnit qui décline les tests unitaires pour différents langages (PHPUnit pour PHP, NUnit pour .net, etc.). C'est ce *framewrok* que nous utiliserons dans sa version 4.

En Java, l'unité sur laquelle va porter les tests est la classe. Il est possible de prendre comme habitude d'écrire une classe de test à chaque fois que l'on écrit une classe. Par convention, le nom des classes de test commence par le nom de la classe que l'on veut tester et est suivi par *Test*. Par exemple, *CalculTest* est le nom de la classe de test associée à la classe *Calcul.*

Junit 4 utilise les annotations de Java. Ces annotations commencent par le caractère @. Une annotation fournit une information sur un élément du code source (classe, méthode, paramètre, etc.). Cette information peut être utilisée pendant la compilation ou pendant l'exécution.

On connaît déjà l'annotation *@Override* pour signaler qu'une méthode redéfinit une méthode d'une classe mère. Cette information est prise en compte par le compilateur qui signale une erreur si par exemple la méthode annotée ne redéfinie pas une méthode existante.

On peut créer ses propres annotations mais cela dépasse le cadre de cet enseignement.

Pour déterminer quelles sont les méthodes de test, Junit 4, quant à lui, recherche les méthodes avec l'annotation *@Test* dans n'importe quelle classe.

4. Exemple

Voir le projet NetBeans *DemoTestsUnitaires2.*

Un cas de test *(CalculTest* par exemple) regroupe plusieurs tests dans une classe.

Une méthode d'assertion (*Assert.assertEquals* par exemple) permet de savoir si une affirmation est respectée ou non. C'est ce qui permet de savoir si un test est réussi ou non.

Une suite de tests (*DemoTestSuite*) regroupe plusieurs cas de tests. Lorsque une suite de tests est exécutée tous les tests de la suite sont exécutés.

5. Création de tests avec NetBeans

NetBeans permet de faire des tests avec JUnit (si vous avez spécifiez cette option au moment de l'installation de NetBeans*).*

Pour écrire une classe de test : clic droit sur le paquetage des sources > nouveau > autre > unit tests > JUnit ; puis dans la fenêtre choisir les options voulues.

Exécutez le fichier pour lancer les tests. Un panneau s'affichera qui présentera les tests réussis en vert et les tests en échec en rouge. On peut voir aussi des informations sur la sortie standard.

6. Exercice

Écrivez une classe *Convertisseur* qui va permettre de convertir des distances d'une unité de mesure dans une autre. Les unités a prendre en compte sont : kilomètre, hectomètre, décamètre, mètre, décimètre, centimètre, millimètre. Ces unités de longueur pourront être définies en utilisant des constantes.

Il faut pouvoir spécifier une unité d'origine, et une unité cible avant d'opérer une conversion. Ainsi, on doit pouvoir par exemple convertir 453,1 décamètres en décimètres.

Il faut aussi disposer d'une méthode qui prend en paramètre une longueur à convertir et qui renvoie la longueur convertie.

Écrivez les tests unitaires qui permettent de tester la classe *Convertisseur*.

7. Différences entre JUnit 3 et JUnit 4

JUnit 3 n'utilise pas les annotations.

Les classes de cas de tests doivent hériter obligatoirement de la classe *TestCase*.

Le nom des méthodes de tests doit obligatoirement commencer par *test*.

La classe représentant une suite de tests doivent hériter obligatoirement de la classe *TestSuite.*