L3 Informatique – Parcours ISI – UE TML **2012-2013**

TDM Tests fonctionnels-Junit

Le travail sera entièrement développé sous Eclipse, en utilisant ses possibilités de génération automatique de cas de test, classes et opérations manquantes …
Pour chaque exercice :

- Créer un projet java, en cochant « separate folders ».
- Créer deux dossiers de sources : l'un pour le code, l'autre pour les tests.
- Vérifier le setup JUnit et le corriger éventuellement :
  - JUnit 3.8.1 doit apparaitre dans le buildpath du projet : *Project > Properties, Java Build Path,* onglet Libraries ; sinon l'ajouter via *Add Library, JUnit, JUnit 3 ;*
  - La vue JUnit doit apparaître dans *Window > Show View ; sinon l'activer via Window > Show View > Other > JUnit*

1. Exercice 1 :

a) En vous inspirant de la classe Book vue en cours, écrire une classe Produit mémorisant le nom et le prix d'un produit.
Générer la classe de test JUnit correspondante et la compléter.
Tester.
Conseils : Ecrire les tests progressivement, de façon à obtenir la barre verte très vite et à y revenir très vite après un échec …

b) On souhaite utiliser cette classe dans un système d'achats en ligne où l'on constitue un panier en y déposant des produits, en indiquant pour chacun la quantité souhaitée. Le montant total du panier est actualisé à chaque fois.
Ecrire les classes nécessaires.
Générer les classes de test JUnit correspondantes et les compléter.
Tester.
Conseils : voir ci-dessus !

c) Générer la suite de tests JUnit pour le paquetage.
Tester.

2. Exercice 2 :

a) Ecrire une classe Monnaie mémorisant un montant et sa devise.
Le montant doit être un entier positif ; la devise s'exprime par une chaîne de trois caractères ("EUR", "USD", "CHF", etc.).
La classe devra proposer deux opérations : ajouter et multiplier (par un entier). On se limitera dans un premier temps à l'ajout de monnaies dans la même devise (quand on additionne deux Monnaie dans la même devise, le résultat est la somme des deux autres, toujours dans la même devise).
Générer la classe de test JUnit correspondante et la compléter.
Tester.

b) L'opération add doit maintenant lever une exception dans le cas où les deux Monnaie ne sont pas dans la même devise.
Tester cette levée d'exception.

3. Exercice 3 : JUnit et TFD (Test First Development)

On envisage de construire une classe MaDate définissant :

- 3 attributs privés (jour, mois, an), de type int
- les getters et setters correspondants
- un constructeur MaDate (int jour, int mois, int an)
- une opération : lendemain () qui calcule la date du lendemain.

a) Ecrire les tests JUnit correspondants, puis le code pour les satisfaire, au fur et à mesure.

b) Faire de même pour une nouvelle opération : estUneDateValide (int jour, int mois, int an) qui retourne un booléen indiquant si la date correspondante est valide.

Remarque : l'exercice s'apparente à la technique de partitionnement en classes d'équivalence qui sera présentée en détail plus tard.

Travail personnel (en fin de séance ou à la maison) **:**
Définir des tests JUnit sur une ou plusieurs classes Java développées dans le cadre d'UE de programmation en Java.