TP : autour de JUnit

JUnit est disponible à `//junit.org` (prendre une version > 4.0).

Exercice 1.
*On définit une classe Money permettant de représenter une certain montant (*amount *dans une devise donnée (*currency*). Le montant doit être positif, et la devise s'exprime comme une chaîne de trois caractères définie par la norme ISO qui va bien (ici, seulement EUR (euro), USD ($), CHF (franc suisse), GBP (livre sterling)). Quand on additionne deux Money de même devise, le résultat a pour montant la somme des deux autres montants.*

```
1  class Money {
2      private int fAmount;
3      private String fCurrency;
4
5      public Money(int amount, String currency);
6      public int amount();
7      public String currency();
8      public Money add(Money m);
9      public Money add(int namount, String ncurrency); // namount negatif possible
10     }
11 }
```

Questions :

1. *En partant de l'en-tête ci-dessus, écrire une classe MoneyTest qui teste si Money est correcte. Bien faire attention à indiquer le besoin et scénario testé pour chaque test. Vous penserez à ajouter des tests de robustesse (bien les séparer des autres). Pensez à garder votre ensemble de tests concis.*
2. *Écrire une classe Money implantant l'en-tête donné et passant tous vos tests.*
3. *Écrire une classe MoneyBag qui prend en compte plusieurs devises différentes et offre une méthode de normalisation du MoneyBag (un seul couple (devise,montant) pour les devises), un test d'égalité et deux méthodes `add` et `subb` (avec un Money et un MoneyBag).*
4. *Écrire les tests pour tout ça. Vérifier que votre programme les passe avec succès.*

Exercice 2 (Couverture structurelle)**.** *Cobertura et Emma sont des logiciels libres permettant de calculer la couverture structurelle d'un jeu de test. Un plugin Eclipse existe aussi pour Emma (EclEmma).*

1. *Installer l'un de ses trois outils de calcul de couverture.*
2. *Reprendre l'exemple de l'exercice 1 et rejouez les tests avec Emma et/ou Cobertura. Calculez les différentes couvertures obtenues. Complétez les tests pour arriver à 100%. Regardez un peu toutes les options / paramètres de l'outil.*
3. *Idem avec l'exercice 7 du TD.*

Exercice 3 (mutations)**.** *Nous allons utiliser l'outil mujava (ou sa version plugin eclipse muclipse) permettant de faire du test de mutations en Java.*

1. *installer mujava ou muclipse*
2. *prendre le programme de l'exercice 8 du TD.*
   (a) *générer les mutants correspondants à ROR et ABS combien y a-t-il de mutants ?*
   (b) *reprendre le jeu de test fait en TD (mutations). Quel est le score de couverture de mutation obtenu ?*
   (c) *compléter pour obtenir 100% de score de mutation*