Prof. Burkhart Wolff
wolff@lri.fr

T. Balabonski, F. Gara, W. Mebrek
blsk@lri.fr, wafaa.mebrek@telecom-sudparis.eu,
fatmagara@gmail.com

TD 3 - Test boîte noire

Semaine du 7 octobre 2019

Exercice 1 (Test fonctionnel de sequence.)
On reprend le cahier de charge du Bank-scenario du TD2, les formalisations des invariants et des contrats des opérations.

Questions

1. Developer un scenario de test abstrait pour un client, le systeme, et les operations `deposit`, `balance` `schedule_transfer` and `exec_transfer`.
2. Developer un scenario de test abstrait pour deux clients, et les operations `deposit`, `balance` `withdraw` pour les deux clients acceder leurs comptes cheques.
3. Developer un scenario de test abstrait pour deux clients, et les operations `deposit`, `balance` `withdraw` pour les deux clients acceder leurs comptes d'epargne.

Exercice 2 (Test fonctionnel)
Une société vend deux produits A et B au prix unitaire de 5 € pour A et de 10 € pour B. Une commande comprend une certaine quantité du produit A et une certaine quantité du produit B. Le coût d'une commande est la somme totale des prix unitaires des produits commandés, à laquelle on applique une réduction selon les règles suivantes :

— Si la somme totale est supérieure ou égale à 200 €, on applique une réduction de 5%, si elle est supérieure ou égale à 1000 €, la réduction est de 20%. Ces deux réductions ne sont pas cumulables et portent sur la somme totale.
— La société souhaitant encourager la vente de A, on applique, sur le prix obtenu grâce à la règle précédente, une réduction supplémentaire de 10% si la commande comprend au moins 45 produits A.

1. Donnez un ensemble de tests pour le calcul du coût total d'une commande. Pour chaque test :
   — expliquez le cas particulier visé par le test (objectif de test) ;
   — donnez la formule du résultat attendu ;
   — donnez des valeurs concrètes et le résultat correspondant attendu.
2. Complétez votre jeu de tests par une analyse aux limites.

Exercice 3 (Option)
On considère une classe `Tableau` permettant de stocker des entiers dans un tableau trié. Le nombre d'éléments du tableau est stocké dans un attribut `taille` et la longueur du tableau dans un attribut `capacite`. La taille est toujours supérieure ou égale à 0, la capacité toujours strictement positive, et la taille inférieure ou égale à la capacité. En particulier, on ne peut pas appeler le constructeur de la classe `Tableau(int capacite)` avec un argument négatif ou nul.

```
public class Tableau {
    private int tab[];
    private int taille;
    private int capacite;

    public Tableau(int capacite) { }

    public boolean inserer(int val) { }

    public boolean supprimer(int val) {
        int i = 0;
        while(i < taille && tab[i] != val) {
            i++;
        }
        if(i == taille) {
            return false;
        }
        for(int j = i; j < taille-1; j++) {
            tab[j] = tab[j+1];
        }
        tab[taille-1] = 0;
        taille = taille-1;
        return true;
    }
}
```

On dispose de deux méthodes `inserer` et `supprimer`. Si le tableau n'est pas plein, la méthode `inserer` ajoute l'élément passé en argument au tableau en conservant l'ordre et renvoie *true*. Si le tableau est plein, elle renvoie *false*. La méthode `supprimer` enlève une occurrence de l'élément passé en argument et renvoie *true*. Elle renvoie *false* si l'élément n'était pas présent.

On cherche à tester la méthode `supprimer` de cette classe, de manière « interne », c'est-à-dire en ayant accès à tous les attributs.

1. Proposer un ensemble de tests pour la fonction `supprimer`.
2. Pour les modifications suivantes du programme, dire si la modification introduit une faute et lorsque c'est possible, donner un cas de test permettant de révéler la faute.
   (a) On modifie la condition du *if* par `i == taille-1`.
   (b) On modifie la condition d'arrêt de la boucle *for* par `j < taille`.
   (c) On modifie l'initialisation de la boucle *for* par `j=i+1`.
   (d) On change `taille` par `capacite` dans la condition du *while* et du *if*.

(e) On supprime la ligne `tab[taille-1] = 0`.

(f) On fait les modifications (d) et (e) ensemble.

3. On teste maintenant cette méthode depuis l'extérieur de la classe. De quels observateurs aurait-on besoin idéalement pour pouvoir détecter autant de fautes qu'avec les tests internes ?

4. On ajoute les méthodes suivantes à la classe `Tableau` :

```
int taille()
int capacite()
boolean present(int val)
```

Quels aspects de l'implémentation ne peut-on plus observer ? Peut-on les tester ? Comment ?

5. Pour deux des tests proposés en question 1, construire la suite d'appels de méthodes (le scénario de test) permettant d'exécuter ce test, depuis l'initialisation des objets jusqu'à la vérification du résultat obtenu.