Les tests unitaires – TP2
Automatiser des tests

Objectif

L'objectif du TP est d'automatiser les tests effectués dans le TP 1. Il s'agit encore de tester la méthode ***prixAPayer*** de la classe ***TarifSpectacle.*** Ou dit autrement d'écrire un test unitaire automatique pour la classe ***TarifSpectacle*** et donc pour la méthode ***prixAPayer*** .

ETAPE 1 – Préparation de la classe de test et du jeu de tests

1) Selon les conventions présentées dans le cours quel sera le nom de la classe permettant d'effectuer des tests unitaires de ***TarifSpectacle*** ?

2) Toujours selon les conventions quel sera le nom de la méthode permettant de tester la méthode ***prixAPayer*** ? Quelle sera l'entête de cette méthode de test ?

3) Créer la classe de test, avec la méthode de test (corps vide) et une fonction *main* dont le rôle sera d'appeler la méthode de test.

4) Définir dans la classe de test un jeu de test (ou jeu d'essai) correspondant à celui que vous aviez établi dans l'étape 3 du TP 1 (stratégie *All pairs*, avec seulement des données d'entrée correctes). Plus précisément vous définirez 4 tableaux constants :

- un tableau constant de caractères ayant pour valeur les codes spectacles du jeu de test :

```
/** Valeur du code spectacle, pour le jeu de tests contenant des valeurs correctes */
private static char[] SPECTACLE = {
        'T', 'T', 'T', 'T', 'O', 'O', 'O', 'O',
        'C', 'C', 'C', 'C', 'D', 'D', 'D', 'D' };
```

- sur le même principe un tableau de caractères ayant pour valeur les codes adhérent du jeu de test :

```
/** Valeur du code catégorie, pour le jeu de tests contenant des valeurs correctes */
private static char[] CATEGORIE = {'S', . . . };
```

- puis un tableau de booléens avec les valeurs du jeu de test concernant la carte d'adhérent :

```
/** Valeur du booléen adhérent, pour le jeu de tests contenant des valeurs correctes */
private static boolean[] ADHERENT = {true, . . . };
```

- un tableau d'entiers avec les résultats attendus pour chaque calcul :

```
/** résultats attendus pour le jeu de tests contenant des valeurs correctes */
```

```
private static int[] RESULTAT = {14, . . . };
```

Bien sûr, il y a un lien entre les cases des différents tableaux situées aux mêmes indices. Ainsi la valeur d'indice 0 du tableau des résultats (donc 14) est le prix à payer pour les valeurs situées à l'indice 0 des 3 autres tableaux (donc 'T', 'S' et *true*, pour Théâtre, Scolaire avec carte d'adhérent).

ETAPE 2 – Coder la méthode de test

Il faut maintenant coder la méthode de test.

1) Vous pouvez d'abord ajouter à la classe la méthode qui affichera le résultat du test (nombre de tests effectués, nombre de tests corrects ou incorrects) : *afficherResultatTest* (voir code sur Moodle).

2) Coder ensuite la méthode de test. Si la méthode testée, donc ***prixAPayer***, renvoie des résultats corrects pour le jeu d'essai, ou dit autrement est correctement programmée, il faut que la méthode de test affiche :

```
Test de la méthode prixAPayer :
-------------------------------


16 test(s) ont réussi sur un total de 16 tests réalisés.
   ==>  Tous les tests sont OK
```

Supposons que par contre la méthode ***prixAPayer*** comporte des erreurs. L'affichage souhaité sera de la forme suivante :

```
Test de la méthode prixAPayer :
-------------------------------

Echec du test avec les valeurs : O / E / true     ==>  résultat calculé = 24     resultat attendu = 21
Echec du test avec les valeurs : C / S / true     ==>  résultat calculé = 18     resultat attendu = 16
Echec du test avec les valeurs : C / E / false    ==>  résultat calculé = 21     resultat attendu = 19
Echec du test avec les valeurs : C / M / true     ==>  résultat calculé = 25     resultat attendu = 23
Echec du test avec les valeurs : C / P / false    ==>  résultat calculé = 29     resultat attendu = 27
Echec du test avec les valeurs : D / S / false    ==>  résultat calculé = 19     resultat attendu = 21
Echec du test avec les valeurs : D / E / true     ==>  résultat calculé = 19     resultat attendu = 18
Echec du test avec les valeurs : D / M / false    ==>  résultat calculé = 23     resultat attendu = 25
Echec du test avec les valeurs : D / P / true     ==>  résultat calculé = 24     resultat attendu = 26

7 test(s) ont réussi sur un total de 16 tests réalisés.
   ==>  9 test(s) ont échoué.
```

ETAPE 3 – Tester

Exécuter la méthode de test automatique :

1) D'abord pour tester la méthode ***prixAPayer*** de la classe ***TarifSpectacle***

2) Puis celle de la classe ***TarifSpectacleBis***

ETAPE 4 – Jeu de tests avec valeurs incorrectes

On souhaite maintenant tester le comportement de la méthode ***prixAPayer*** lorsque l'une des données d'entrée, le code spectacle ou le code catégorie, est incorrecte. Le comportement attendu est la levée de l'exception *IllegalArgumentException*.

1) Si vous n'avez pas traité l'étape 6 du TP 1, compléter le jeu de tests élaboré avec la stratégie ***All pairs***, de manière à prendre en compte les cas dans lesquels un argument de la méthode ***prixAPayer*** serait invalide.

2) Définir ce jeu de tests comportant des valeurs incorrectes en définissant 3 autres tableaux constants (il est inutile de définir le 4ème tableau contenant les résultats)

```
/** Valeur du code spectacle, pour le jeu de tests avec valeurs incorrectes */
private static char[] SPECTACLE_ERR = {'T', 'O', 'C', 'D', 'I', 't', 'c', '5' };

/** Valeur du code catégorie, pour le jeu de tests avec valeurs incorrectes */
private static char[] CATEGORIE_ERR = {'i', . . .};

/** Valeur du booléen adhérent, pour le jeu de tests avec valeurs correctes */
private static boolean[] ADHERENT_ERR = {
        true, false, true, false, true, false, true, false, true };
```

ETAPE 5 – Coder la méthode de test

La méthode de test devra maintenant effectuer 2 séries de tests : une série avec un jeu de tests contenant des valeurs correctes et une autre série avec des valeurs incorrectes. Pour que cette méthode ne comporte pas trop de lignes de code, il faut faire en sorte qu'elle invoque 2 autres méthodes :

```
/**
 * Test de la méthode prixAPayer avec des données d'entrée correctes
 * et incorrectes. Le test est réalisé en 2 parties.
 */
public static void testPrixAPayer() {
    System.out.print("\nTest de la méthode prixAPayer :\n"
            + "-------------------------------\n\n");
    testPrixAPayerValeurCorrecte();
    testPrixAPayerValeurIncorrecte();
}
```

Exemple d'exécution si la méthode ***prixAPayer*** est correctement codée :

```
Test de la méthode prixAPayer :
-------------------------------

  ===> Test 1/2 - Test avec des valeurs correctes :
  ------------------------------------------------

16 test(s) ont réussi sur un total de 16 tests réalisés.
   ==>  Tous les tests sont OK


  ===> Test 2/2 - Test avec des valeurs incorrectes :
  --------------------------------------------------

8 test(s) ont réussi sur un total de 8 tests réalisés.
   ==>  Tous les tests sont OK
```

Exemple d'exécution si la méthode ***prixAPayer*** comporte des erreurs :

```
Test de la méthode prixAPayer :
-------------------------------

  ===> Test 1/2 - Test avec des valeurs correctes :
  ------------------------------------------------

Echec du test avec les valeurs : C / S / true     ==>  résultat calculé = 18     resultat attendu = 16
Echec du test avec les valeurs : C / E / false    ==>  résultat calculé = 21     resultat attendu = 19
Echec du test avec les valeurs : C / M / true     ==>  résultat calculé = 25     resultat attendu = 23
Echec du test avec les valeurs : C / P / false    ==>  résultat calculé = 29     resultat attendu = 27

12 test(s) ont réussi sur un total de 16 tests réalisés.
   ==>  4 test(s) ont échoué.


  ===> Test 2/2 - Test avec des valeurs incorrectes :
  --------------------------------------------------

Echec du test avec les valeurs : t / E / false   ==>  le résultat a pu être calculé et est égal à 17
Echec du test avec les valeurs : c / M / true   ==>  le résultat a pu être calculé et est égal à 25

6 test(s) ont réussi sur un total de 8 tests réalisés.
   ==>  2 test(s) ont échoué.
```

1) Transformer la méthode de l'étape 2 pour quelle devienne ***testPrixAPayerValeurCorrecte***

2) Coder la méthode ***testPrixAPayerValeurIncorrecte***.

ETAPE 6 – Tester

Exécuter la méthode de test automatique :

1) D'abord pour tester la méthode ***prixAPayer*** de la classe ***TarifSpectacle***

2) Puis celle de la classe ***TarifSpectacleBis***

3) Puis celle de la classe ***TarifSpectacleTer*** (à récupérer sur Moodle)