

TP 5 : Tests unitaires avec MSTest

Partie 1

Pour un contexte particulier, on a besoin de redéfinir la classe `List` pour bien gérer les occurrences d'un nombre. Pour cela, on utilise l'interface suivante :

```
namespace ExerciceTest
{
    public interface IOccurrence
    {
        int ComputeOccurrence(int i);
        bool ContainsNOccurrence(int i, int occ);
        void RemoveAllIOccurrence(int i);
        void ReplaceAllIOccurrence(int oldValue, int newValue);
        List<int> SubListWithoutOccurrence(int i);
    }
}
```

1. Dans une nouvelle solution, créez un nouveau projet Console (.NET Framework) nommé `ExerciceTest`.
2. Créez l'interface `IOccurrence`.
3. Créez une classe `OccurrenceList` qui étend `List<int>` et implémente `IOccurrence`.
4. Définissez un constructeur sans paramètre et un constructeur qui permet l'écriture suivante :
`OccurrenceList list = new OccurrenceList { 1, 2, 3, 2, 3, 3 }.`
5. Implémentez toutes les méthodes abstraites de l'interface `Occurrence` :
 - `ComputeOccurrence(int i)` retourne le nombre d'occurrence de `i` dans la liste.
 - `ContainsNOccurrence(int i, int occ)` retourne `true` si la liste contient au moins `occ` occurrences de `i`, `false` sinon.
 - `RemoveAllIOccurrence(int i)` supprime toutes les occurrences de `i` si ce dernier est dans la liste, sinon elle lève une exception de type `ValueNotFoundException` (une exception à définir).
 - `ReplaceAllIOccurrence(int oldValue, int newValue)` remplace toutes les occurrences de `oldValue` par `newValue` si `oldValue` est dans la liste, sinon elle lève aussi une exception de type `ValueNotFoundException`.
 - `SubListWithoutOccurrence(int i)` retourne une nouvelle sous-liste sans les occurrences de `i` (la liste originelle ne sera pas modifiée). Si la liste ne contient aucune occurrence de `i` elle lève une exception de type `ValueNotFoundException`.
6. En utilisant **MSTest**, créez une classe de test pour `OccurrenceList` ayant comme attribut :
`OccurrenceList list = new OccurrenceList { 1, 2, 3, 2, 3, 3 }.`
7. Utilisez les méthodes de **Fluent Assertions**, vues en cours, pour tester et valider les trois dernières méthodes de la classe `OccurrenceList`.
8. Utilisez `[DataRow]` pour tester `ContainsNOccurrence(int i, int occ)`.
9. Utilisez `[DynamicData]` pour tester `ComputeOccurrence(int i)`.

Partie 2

Considérons l'interface suivante :

```
namespace ExerciceTest
{
    public interface IAverage
    {
        float Compute(List<int> l);
    }
}
```

1. Créez l'interface `IAverage`.
2. Dans `OccurrenceList`, déclarez un attribut de type `IAverage`.
3. Définissez un constructeur qui permet l'écriture suivante :
`OccurrenceList list = new OccurrenceList(average, new List<int> { 1, 2, 3, 2, 3, 3 })`.
4. Définissez une méthode `AvgWithoutOccurrence(int i)` qui utilise la méthode `Compute` de `Average` pour retourner la moyenne de la sous liste qui ne contient aucune occurrence de `i`.
5. Dans `OccurrenceListTest`, définissez un mock sur `IAverage`.
6. Utilisez `Setup ... Returns` pour tester `AvgWithoutOccurrence(int 2)`.
7. Vérifiez que le mock a été utilisé.