Mini-projet – Tests et Mocks

Contextes et consignes

Dans cette évaluation vous devrez principalement réaliser des tests unitaires.

Cette évaluation sera faite en **binôme** uniquement (**pas de trinôme**, contactez votre enseignant si vous êtes seul). Vous devrez remettre **une archive au format zip contenant votre projet éclipse complet ainsi qu'un rapport au format PDF**. L'archive doit être nommée avec les noms des deux membres du binôme (ex : MARTIN-DUPONT.zip).

Votre code sera évalué sous Eclipse, à charge pour vous d'avoir quelque chose d'importable directement. Le rapport au format PDF décrira l'évolution de vos tests et de votre code pour permettre d'évaluer la méthode que vous avez suivie pour arriver au code final. Vous justifierez vos choix dans le rapport également.

Exercice 1 : Présentation générale

Dans cet exercice nous réaliserons un dictionnaire bilingue dont l'objectif est de permettre des traductions dans un sens ou dans l'autre (ex : Français vers Anglais et Anglais vers Français).

Consignes

- ✓ Ecrivez une classe **DictionaryTest** permettant de tester une classe **Dictionary**
- ✓ Ecrivez une classe **Dictionary** permettant de construire un objet à partir d'un nom fourni en paramètre du constructeur
- ✓ Ecrivez une méthode **String getName()** dans la classe **Dictionary** permettant de renvoyer le nom du dictionnaire.
- ✓ Modifiez votre test en conséquence
- ✓ Testez que votre dictionnaire est bien vide

Exercice 2 : Ajout des traductions

Consignes

- ✓ Ecrivez un test permettant de vérifier que l'ajout d'une traduction (méthode **void addTranslation(String fr, String en)**) et sa vérification (**String getTranslation(String word)**) se passent correctement.
- ✓ Modifiez votre classe **Dictionary** pour passer votre test. Vous aurez besoin d'avoir une structure de données vous permettant de stocker vos traductions (**HashMap**)

Exercice 3 : Traductions multiples

L'un des intérêts des dictionnaires bilingues est de permettre d'avoir des traductions multiples pour un même mot.

Consignes

- ✓ Ecrivez le test permettant de prendre en compte cette spécificité.
- ✓ Modifiez votre classe **Dictionary** pour pouvoir gérer plusieurs traductions pour un même mot (dans un seul sens pour le moment). Vous aurez besoin d'une nouvelle méthode **getMultipleTranslations** qui renverra une liste de traductions possibles.
- ✓ Refactorisez votre code dans la classe **Dictionary** et montrez cette phase dans votre rapport.

Exercice 4 : Traductions multiples bidirectionnelle

Pour le moment, vos traductions ne vont que d'une langue vers l'autre. Proposez une version permettant de faire des traductions dans les deux sens.

Consignes

- ✓ Ecrivez le test permettant de vérifier les traductions bidirectionnelles.
- ✓ Modifiez votre classe **Dictionary** pour pouvoir gérer cette bi-directionnalitée. Faites en sorte de ne la modifier QUE pour passer vos tests
- ✓ Proposez ensuite une nouvelle version de **Dictionary** permettant de passer tous les tests mais sans tricher. Pensez à refactoriser au maximum.

Exercice 5 : Chargement de fichiers

Dans cet exercice nous allons nous intéresser à l'alimentation de notre dictionnaire à travers l'utilisation de fichiers externes. Malheureusement, votre enseignant étant officiellement en vacances, il ne vous a pas fourni les fichiers en question…

Spécifications

Le format d'entrée des dictionnaires consiste en un fichier au format texte, dans lequel la première ligne représente l'identifiant du dictionnaire, et les lignes suivantes contiennent un caractère « ; » séparant les traductions (exemple : mot;word). Pour la lecture de ce fichier vous allez devoir écrire un parseur de fichier (**FileParser**) qui pourra s'appuyer sur la classe **Scanner** ou la classe **BufferedReader**.

Consignes

- ✓ Comment allez-vous intégrer ce parseur dans votre architecture logicielle existante ? Faites un petit diagramme de classe montrant les relations entre vos classes
- ✓ Ecrivez un test permettant d'évaluer votre parseur de fichiers. Comme vous ne disposez pas des fichiers, il est recommandé de passer par des **Mocks** avant de réaliser votre propre fichier d'exemple. Ces **mocks** pourront notamment reproduire le comportement d'un **Scanner** ou d'un **BufferedReader**.

✓ Pensez à bien tester tous les cas limites.