Techniques de test : TP1

**Objectif**

L’objectif de ce TP est de vous permettre d’appréhender les tests unitaires et la qualité logicielle. Vous verrez qu’il est difficile de définir une seule ligne de conduite dans les tests et que les spécifications peuvent être interprétées différemment…

**Tests Unitaires**

Préparez un répertoire à votre nom dans lequel vous stockerez sous-répertoires de sources et comptes-rendus sous forme de fichiers textes ou autre de votre choix.

**Création de la classe à tester**

Ecrire la classe « Justifier » permettant de justifier du texte à l’affichage. Cette classe doit implémenter l’interface IBaseJustifier (disponible en fin d’énoncé ou par mail) et avoir les propriétés suivantes :

* Un constructeur prenant en paramètre un entier définissant le nombre de caractères de justification.
* Le constructeur par défaut justifie à 30 caractères
* Une méthode appendText prenant en paramètre une chaine de caractères et l’ajoutant au texte de la classe (par défaut vide) en ajoutant un espace si du texte était déjà présent.
* Une méthode displayText qui affiche sur la sortie standard le texte retourné par la méthode justifyText.
* La méthode principale du TP : justifyText qui retourne le texte actuellement entré, justifié : c'est-à-dire qu’on passe à la ligne lorsqu’on a atteint le nombre de caractères de justification ou avant de manière à ne pas couper les mots.

Ecrire également une application exécutable (Main) appelant la classe Justifier avec quelques cas de vérification.

*Enregistrer ce projet dans un sous-répertoire Test\_TP1\_01*

**Création des tests unitaires**

Ecrivez une série de tests unitaires pertinents permettant de valider le comportement de votre classe. Justifiez vos choix dans un fichier texte. Notamment, par rapport à l’énoncé de la classe : Quels choix ou hypothèses avez-vous fait ? Que peut-on ajouter à la définition de la classe ?

*Note : Pour faire un test JUnit, dans Eclipse : se référer à la dernière partie de cet énoncé.*

*Enregistrer ce projet (classe + Tests) et le commentaire dans un sous-répertoire Test\_TP1\_02*

**Tests croisés**

Echangez votre classe « Justifier » avec un autre élève. Utilisez vos tests unitaires sur cette classe et analysez le temps mis pour les exécuter (par rapport au temps précédemment mis pour les écrire et au temps de réalisation de tests manuels avec contrôle « humain »). Détaillez cette analyse dans un fichier texte.

Analysez ensuite le résultat des tests : Comparez avec les résultats de votre classe, quelles sont les divergences et pourquoi ? Détaillez cette analyse dans le fichier texte.

Répéter cette séquence avec d’autres classes obtenues dans le groupe.

Concluez : Quels sont les tests « manquants » à votre classe, quelles hypothèses différencient votre classe des autres ?

*Enregistrer ce projet (les classes testées + Tests) et le fichier des commentaires (en précisant pour chaque commentaire sur quelle classe il porte) dans un sous-répertoire Test\_TP1\_03*

**Question subsidiaire :** Comment aurait-on pu faire (hormis en copiant les uns sur les autres ) pour que les classes répondent toutes correctement à la série de tests unitaires ? (Test\_TP1\_04.txt)

**Modes opératoires**

**Interface IBaseJustifier**

/\*\*

\* Package principal du TP1 du module "Techniques de tests" pour le CERI.

\* @author Thomas MERLY

\*/

package ceri.tests.tp1.merly;

/\*\*

\* Interface de base des classes de justification.

\* @author Thomas MERLY

\*

\*/

public interface IBaseJustifier {

/\*\*

\* Justifie et affiche dans la console le texte contenu en buffer.

\*/

void displayText();

/\*\*

\* Ajoute le texte fourni au buffer.

\* @param textToAppend Le texte à ajouter.

\*/

void appendText(String textToAppend);

/\*\*

\* Justifie le texte du buffer et retourne le résultat.

\* @return Le texte du buffer "justifié" (Ajout de passages à la ligne afin de respecter une largeur maximum donnée).

\*/

String justifyText();

}

**Ajouter des tests unitaires (Eclipse)**

Pour ajouter des tests unitaires à votre classe, sélectionnez « Fichier », « Nouveau » puis « Cas de test JUnit ».



**Figure 1 : Création d'une classe de tests.**

Dans les boîtes de dialogues, vérifier que vous êtes sur la version 1.4 ou supérieure de JUnit.



**Figure 2 : Paramétrage des tests**

Puis sélectionner les méthodes faisant l’objet de tests (dans notre cas, seules les méthodes appendText et justifyText seront testées).



**Figure 3 : Sélection des méthodes à tester (construction des squelettes de test)**

Une boîte de dialogue demandant la confirmation d’ajout des librairies NUnit au projet peut apparaître. Il faut la valider.



**Figure 4 : Confirmation d'ajout des JAR NUnit à la compilation.**

**Assertions**

Rappel : Les tests se font par assertion (condition à vérifier pour que le test continue…)