R4.02 - Qualité de développement
TP2

Ce TP est à réaliser en équipes de 2 à 3 étudiants. Il est important qu'il y ait un nombre pair d'équipes.

Le code pour ce TP est disponible sur Gitlab :

`https://gitlabinfo.iutmontp.univ-montp2.fr/r402/tp2/`

1 Spécifier une structure de données

Le but de cette première partie est d'écrire une spécification pour une classe qui modélise une structure de données. Vous ne *devez pas* implémenter la structure de donnée en question, mais uniquement définir les éléments nécessaires aux utilisateurs de cette classe (signature des méthodes + documentation). Le fichier `Ensemble.java` fournit un exemple d'une telle spécification.

La structure de donnée à spécifier dépend du numéro de groupe :

— pour les groupes avec un numéro pair : une pile [1]

— pour les groupes avec un numéro impair : un tableau associatif, aussi appelé dictionnaire [2]

Rédigez la spécification pour votre classe :

1. Définissez l'ensemble des méthodes publiques de la classe (y compris le ou les constructeurs), en spécifiant le type de retour, celui des paramètres, et éventuellement les exceptions qu'une méthode peut retourner (avec le mot-clé `throws`).
2. Spécifiez chaque méthode : décrivez ses paramètres et le résultat qu'elle produit. Si le comportement de la méthode dépend d'éléments autre que ses paramètres (par exemple l'état de la structure de donnée, ou bien une variable globale), décrivez cette dépendance. De même, si la méthode modifie l'état du système, décrivez ce comportement.
3. Documentez aussi la classe avec une brève description de son rôle, et de ses invariants éventuels.

1. `https://fr.wikipedia.org/wiki/Pile_(informatique)`
2. `https://fr.wikipedia.org/wiki/Tableau_associatif`

Après 1 heure, échangez votre fichier de spécification avec un autre groupe (les groupes 1 et 2 échangent leurs fichiers, ainsi que les groupes 3 et 4, etc.) et passez à la partie suivante.

2 Lire une spécification

Pour la deuxième partie de ce TP, chaque groupe va lire une spécification écrite par un autre groupe et l'évaluer selon les critères suivants :

- langage : les descriptions sont-elles claires ? le vocabulaire utilisé est-il cohérent ?
- correction : le comportement décrit par la spécification vous semble-t-il conforme à ce qui est généralement attendu de ce type de structure de données ?
- complétude : la spécification décrit-elle le comportement des méthodes sans ambiguïté ? Décrit-elle tous les différents cas qu'il est possible de rencontrer ?

Après 30 minutes, rejoignez le groupe avec lequel vous avez échangé votre spécification. Ensemble, vous discuterez des points positifs et négatifs de chaque spécification, et vous les corrigerez si besoin.

3 Écrire des tests d'après une spécification

En utilisant JUnit, écrivez une suite de test correspondant à la spécification écrite par l'autre groupe. Chaque méthode devra avoir au moins un test. Si la spécification décrit plusieurs cas possibles pour une méthode, les tests devront couvrir ces différents cas. Le fichier `EnsembleTests.java` fournit un exemple d'une suite de test correspondant à la spécification de la classe `Ensemble`.

Lancez les tests écrits pour vous assurer qu'ils échouent (cette étape sert à valider le fait que les tests contrôlent effectivement quelque chose : un test qui passe sans implémentation ne contrôle rien).

4 Développer une implémentation minimale

Implémenter la classe pour laquelle vous venez d'écrire des tests. Le but est d'obtenir le plus rapidement possible une structure qui satisfasse tous les tests. Il est donc inutile à ce stade d'essayer d'optimiser la structure de donnée. Cette étape permet de produire rapidement un logiciel fonctionnel, mais elle sert aussi à contrôler le code de test, car celui-ci peut aussi contenir des erreurs.