

Exercice : Les Design Patterns

# Enoncé

L’objectif de cet exercice est d’implémenter 4 scénarios différents reposant sur trois Design Patterns. La vérification se fait par le biais de tests unitaires. Le but est de faire passer tous les tests en vert.

Vous disposez pour cela d’une solution comprenant 2 projets :

* ITI.DesignPattern, contenant 4 dossiers correspondant aux différents scénarios (le code écrit ne permet rien de plus que de pouvoir compiler sans erreur).
* ITI.DesignPattern.Tests, contient les tests unitaires.

Pour faire tourner les tests unitaires, exécuter les tests via l’onglet *Test* de Visual Studio.

Pour les autres tests, à vous de faire en sorte qu’ils passent en vert. Pour cela, vous avez le droit de faire ce que bon vous semble dans le projet ITI.DesignPattern (modifier l’implémentation des méthodes, des propriétés ou des constructeurs, ajouter des membres *internal* ou *private*, créer des nouveaux types, etc.).

En revanche, vous ne devez pas modifier le projet ITI.DesignPattern.Tests.

Les tests unitaires sont là pour spécifier la façon détaillée les fonctionnalités attendues.

Cependant voici une courte description de ce qui est attendu :

* AbstractFactory (Design Pattern : Abstract Factory)
  + Permet de générer une multitude d’objets différents à partir de la même classe.
* Observer (Design Pattern : Observer)
  + Permet à 0, un ou n observateurs de s’abonner à un fournisseur et d’être notifié lors d’événements ou de mise à jour de ce fournisseur.
* Decorator (Design Pattern : Decorator)
  + Permet de modifier ou d’étendre le comportement de plusieurs objets sans avoir recours à de l’héritage multiple.
* AbstractFactoryWithObserverWithDecorator (Design Patterns : Abstract Factory / Observer / Decorator)

Il y a 22 tests à passer au vert :

Une image contenant texte

Description générée automatiquement

Au boulot !

Fait par : 🐱‍👤👾👹