

Architecture de JUnit

Ce sujet propose de réaliser un framework de test et, à l'instar de JUnit version 3, utiliser l'inspection pour permettre au programmeur d'écrire plus facilement ses classes de test. Ce sujet s'appuie sur le sujet du TD 8 et le complète.

Le diagramme de classe de la figure 1 présente :

- l'application à tester (Monnaie et l'exception DeviseInvalideException) et ses deux programmes de « test » initiaux (TestMonnaie1 et TestMonnaie2)
- l'architecture du framework de test composée de :
 - l'interface Test qui décrit un test,
 - la classe Assert qui définit la méthode `assertTrue` et l'exception `EchecError`,
 - la classe TestElementaire, à l'exception du code de la méthode `lancer`,
 - le squelette de la classe SuiteTest qui permet de regrouper plusieurs tests.
- Les deux programmes de tests de la classe Monnaie réécrit sous la forme de test élémentaires (TEMonnaie1 et TEMonnaie2 avec la classe TEMonnaie factorisant l'initialisation qui est commune aux deux classes de test).

Exercice 1 : Comprendre l'architecture proposée

Commençons par comprendre les choix faits.

- 1.1 Regarder rapidement le code des classes fournies, en particulier TestElementaire.
- 1.2 Expliquer pourquoi la méthode `lancer` est déclarée **final**.
- 1.3 Expliquer pourquoi `prepare`, `nettoyer` et `tester` sont **protected**.
- 1.4 Expliquer pourquoi `prepare`, `nettoyer` et `tester` déclarent l'exception `Throwable`.
- 1.5 Expliquer pourquoi `prepare` et `nettoyer` ne sont pas déclarées abstraites alors que `tester` l'est.

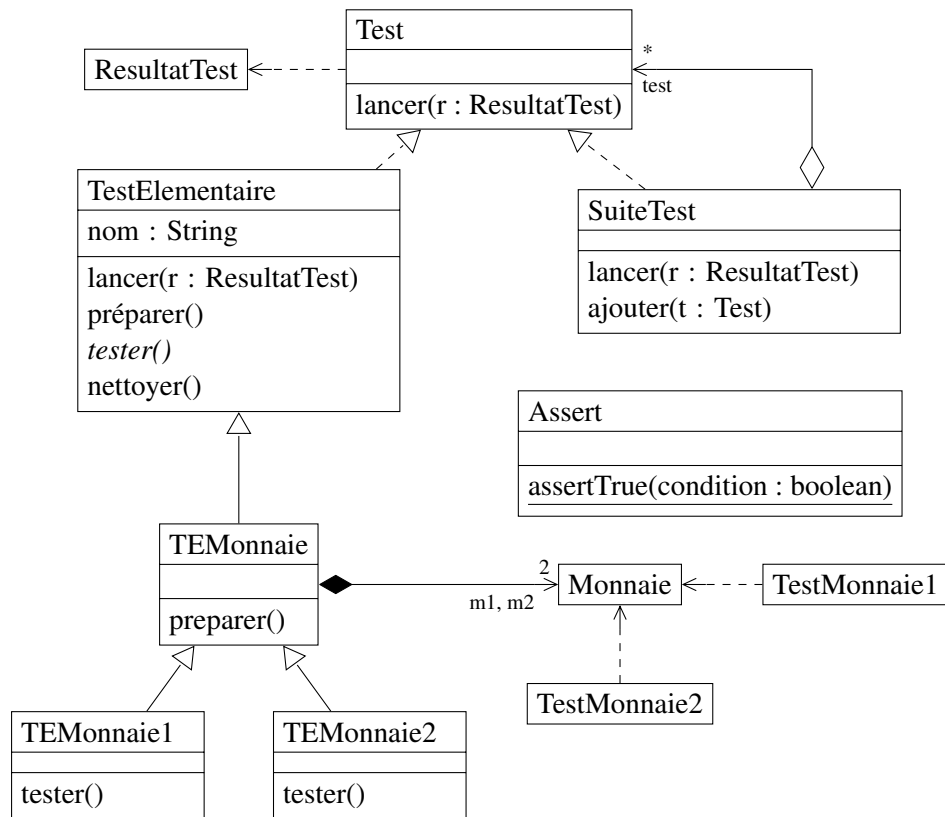
Exercice 2 : La classe TestElementaire

Complétons la classe TestElementaire.

- 2.1 Écrire le code de la méthode `lancer`.
- 2.2 Écrire les deux programmes de test TEMonnaie1 et TEMonnaie2 qui correspondent respectivement à TestMonnaie1 et TestMonnaie2. On factorisera l'initialisation des données `m1` et `m2` dans une classe TEMonnaie.
- 2.3 Ajouter deux nouveaux tests sur la classe Monnaie, le premier qui contient une erreur fonctionnelle (TEMonnaieEchec), le second qui contient une erreur de programmation (TEMonnaieErreur) et vérifier qu'ils sont correctement comptabilisés.

Exercice 3 : La classe SuiteTest

La classe SuiteTest est une réalisation de Test. Elle regroupe plusieurs tests. Lancer une suite de tests consiste à lancer successivement chacun des tests qu'elle contient.

FIGURE 1 – L'architecture du framework de test utilisée pour tester la classe `Monnaie`

3.1 Écrire la classe `SuiteTest` en utilisant, pour stocker les tests, une structure de données existant dans la bibliothèque Java. Plus précisément, on utilisera `java.util.Collection` comme type de l'attribut représentant les tests.

3.2 Écrire une classe principale `SuiteMonnaie` qui construit, puis lance une suite qui contient tous les tests élémentaires déjà écrits pour la classe `Monnaie`.