Introduction au développement logiciel

Partie 4 : TDD

1 Préambule

Vous allez développer un petit programme à l'aide de la méthode *TDD*. Pour rappel à chaque étape, vous devez d'abord produire un test qui devra échouer, puis coder la fonctionnalité qui fait passer le test.

Plus concrètement, vous devez systématiquement passer par trois phases successives :

1. Écrire un test rouge
2. Écrire le code le plus simple pour que le test soit vert
3. Supprimer la duplication (code/test) et améliorer la lisibilité. C'est la phase du refactoring.

Pour la phase 2, tous les coups sont permis. Le plus important est de faire passer le test au vert.

Plus généralement, le TDD est une approche avec laquelle on cherche à cadrer la production de code par petits incréments qui marchent (c'est-à-dire qui sont testés).

2 Une calculatrice

Procédez à la conception d'une calculatrice prenant et analysant une chaine de caractère en paramètre. Veuillez à ce que chaque étape suive bien les règles du *TDD*.

ÉTAPE 1. Créez une simple calculatrice avec une fonction `add(numbers)`. Cette méthode prend en paramètre une chaîne de caractères qui représente 0, 1 ou 2 nombres séparés par des virgules. Elle retournera leur somme (pour une chaîne de caractères vide, elle retournera 0). Testez avec `""`, `"1"` ou `"1,2"`

ÉTAPE 2. Autoriser la fonction `add` à prendre en compte un nombre quelconque de nombres.

ÉTAPE 3. Autoriser la fonction `add` à prendre en compte les retours à la ligne en plus des virgules comme séparateur. Par exemple : `"1\n2,3"` donne `6`. Par contre `"1,\n2"` n'est pas ok (pas deux séparateurs à la suite).

ÉTAPE 4. Autoriser la fonction `add` à prendre en compte des séparateurs de un caractère. Pour changer de séparateur, la chaînes de caractères peut commencer par la ligne : `"//[séparateur]\n"`. La première ligne est optionnelle et tous les scénarios précédents restent valides. Par exemple : `"//;\n1;2"` vaut `3`.

ÉTAPE 5. Les nombres supérieurs à 1000 sont ignorés. Par exemple `"2,1001"` vaut `2`.

ÉTAPE 6. Appeler la fonction `add` avec un nombre négatif dans la chaîne entraîne une exception avec le message `"negatives not allowed"` suivi du nombre négatif passé dans la chaîne de caractères. S'il y a plusieurs nombre négatifs, il faut tous les ajouter dans le message de l'exception.

ÉTAPE 7. Le délimiteur peut avoir n'importe quelle longueur. Par exemple : `"//::$\n1::$2::$3"` vaut `6`.

ÉTAPE 8. Autoriser plusieurs délimiteurs, séparés par le caractère `/` : `"//*=/%\n1*=2%3"` vaut `6`.