L'art du test

Thomas Clavier

Évaluation

Le score sur le serveur n'est pas un critère d'évaluation, en revanche la propreté du code, la présence des tests et la rigueure de l'utilisation de git sont des critères d'évaluation.

Codons

L'objectif du TP est de réaliser un convertisseur de chiffre romains.
Créez un projet git public et le référencer sur l'url au tableau et sur `https://github.com/tclavier/tp-tdd`. Attention de ne pas versionner votre "workspace" mais bien votre projet de conversion en chiffre romain.
Décompresser l'archive du projet dans votre workspace eclipse puis l'importer avec le menu "import... > General > Existing projects into workspace"
Dans le répertoire du projet ajouter le remote git vers votre projet public.

```
git remote add origin url
```

avant de commiter et "pusher" votre travail.

Test Driven Development

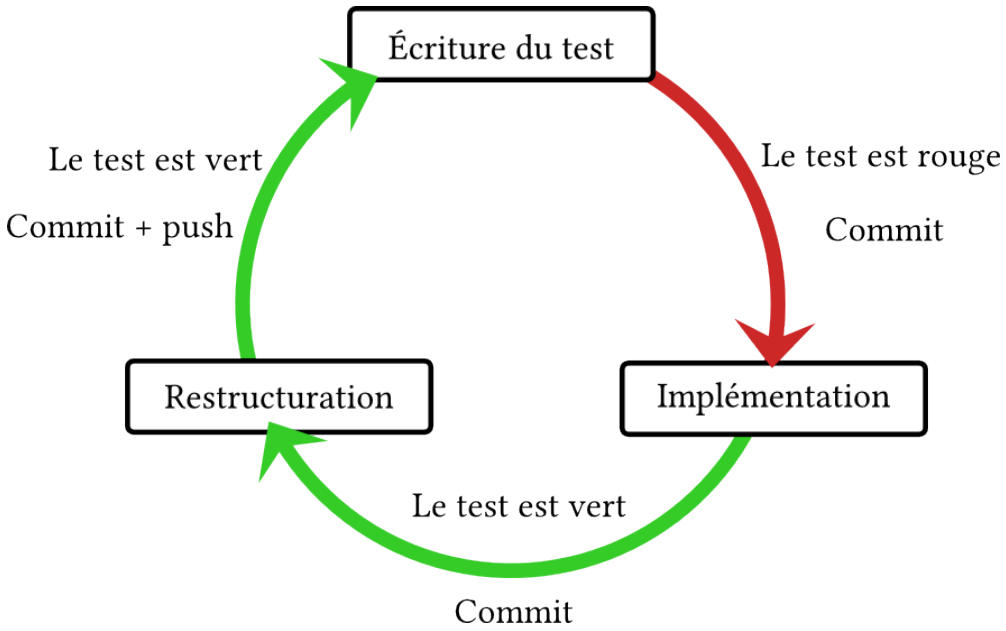
Faire du développement piloté par les tests permet d'atteindre 3 objectifs :

- Contrôler et conserver la qualité du produit tout au long du projet
- Se focaliser sur l'amélioration du code, plutôt que sur la correction des bugs
- améliorer la qualité de vie de l'équipe en automatisant un maximum de tâches redondantes et ennuyantes

Le déroulement des développements est le suivant :

- écriture du test
- vérifier que le test est rouge
- commit
- implémenter la solution la plus simple pour faire passer le test au vert
- commit

- restructurer le code
- vérifier que tous les tests sont encore verts
- commit



À propos des tests

Avec JUnit 4 il est possible d'utiliser des annotations :

- @Test pour définir une méthode "public void" comme une méthode de test.
- @Before pour lancer une méthode "public void" avant chaque test.
- @After pour lancer une méthode "public void" après chaque test.

Voir `https://github.com/junit-team/junit/wiki` pour toute la documentation.
Un test se décompose toujours en 3 parties : Given, When, Then. Dans le Then il est préférable de ne mettre qu'un seul assert.
Il est important de nommer le test avec un nom qui explique clairement l'intention fonctionnelle. Comme les phrases peuvent être longue l'utilisation d'une notation snake_case est authorisé. C'est d'ailleurs le seul cas d'usage de cette notation dans la norme Oracle.

```
import org.junit.Test;
import org.junit.Assert;

...

@Test
```

```
public void num2text_should_return_zero_when_0_is_in_inut () {
  // Given
  Convert myConverter = new Convert();
  // When
  String text = Convert.num2text("0");
  // Then
  Assert.assertEquals("zero", text);
}
```

Code propre

Le code propre c'est du code :

- qui passe ses tests,
- qui ne contient pas de redondance,
- qui est explicite,
- qui est minimal.

Nous passons plus de temps à lire du code qu'à en écrire. En rendant le code facile à lire, nous le rendons plus facile à écrire.
Quelques règles de base :

- La règle du boy scout
- La ligne vide
- pas de commentaire
- max 20 lignes par méthode
- moins de 3 niveaux d'indentation
- Utiliser des Exceptions et pas des valeurs spéciales
- Utiliser stdin, stdout et stderr
- Pas de variable globale
- Des noms de variables explicites
- Modulariser
- Couplage faible