Conduite de tests et qualite

January 11, 2019

Thomas COUCHOUD
thomas.couchoud@etu.univ-tours.fr
Victor COLEAU
victor.coleau@etu.univ-tours.fr

Chapter 1

Gitlab

1.1 Scénario 1 avec bouchons

Commit 5636df0ebce42614784c82e3efece3b0b1619001

Etant donné que nous étions parti dans la mauvaise direction avec des tests unitaires, nous avions une grosse partie de déjà codé, notamment la pile.

Le commit cité plus haut implémente le premier test fonctionnel « FunctionnalKeyboardConsole ». Nous avons pu ajouter quelques bouchons:

1. ViewBottomPile
2. ViewInputPile
3. ViewTopPile
4. Pile (le peu qu'il reste):
   4.1. notifyObservers
   4.2. peek

Le code ne compile pas car certains constructeurs ne sont pas bons, mais nous avons fixé ça par la suite. Cependant avec les constructeurs intégrés, les tests ne passent pas, mais le corps du code est présent.

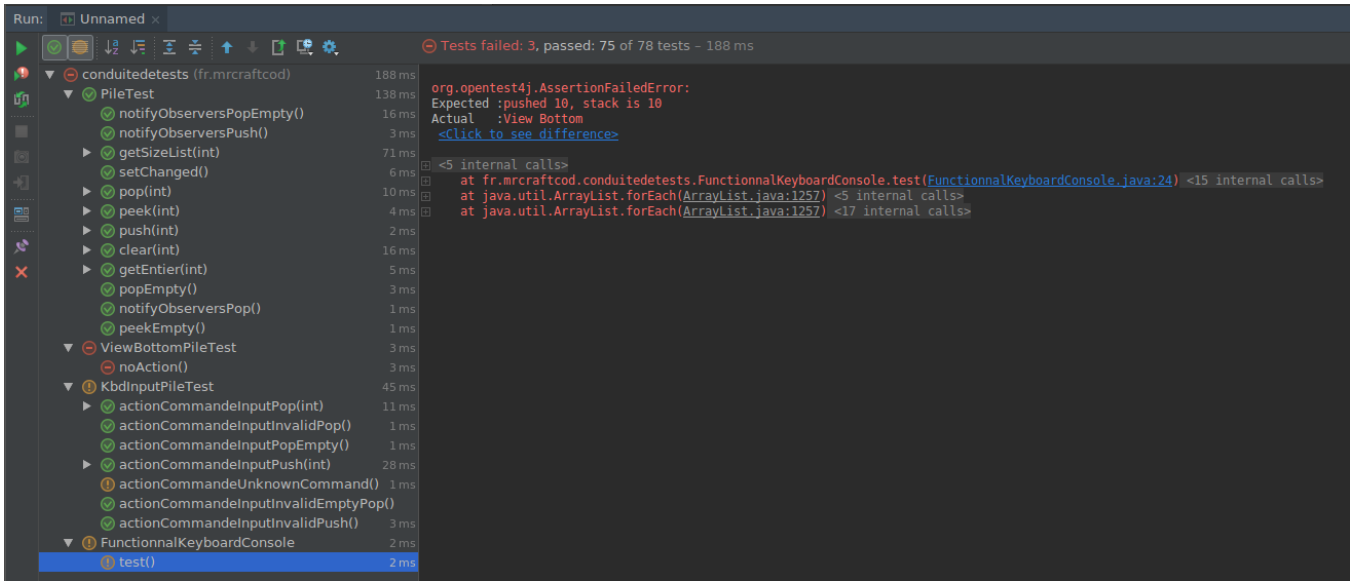

Figure 1.1 – Tests une fois le code fixé

Comme on peut le voir sur l'image le code peut être lancé, cependant le test bloque sur la première instruction car après un push, rien n'est affiché dans la console.

1.2 Etape 2

Une fois le premier scénario complété, nous avons rajouté un test fonctionnel très proche du premier mais qui cette fois-ci utilise un autre observer (top pile). Le commit est disponible ici.

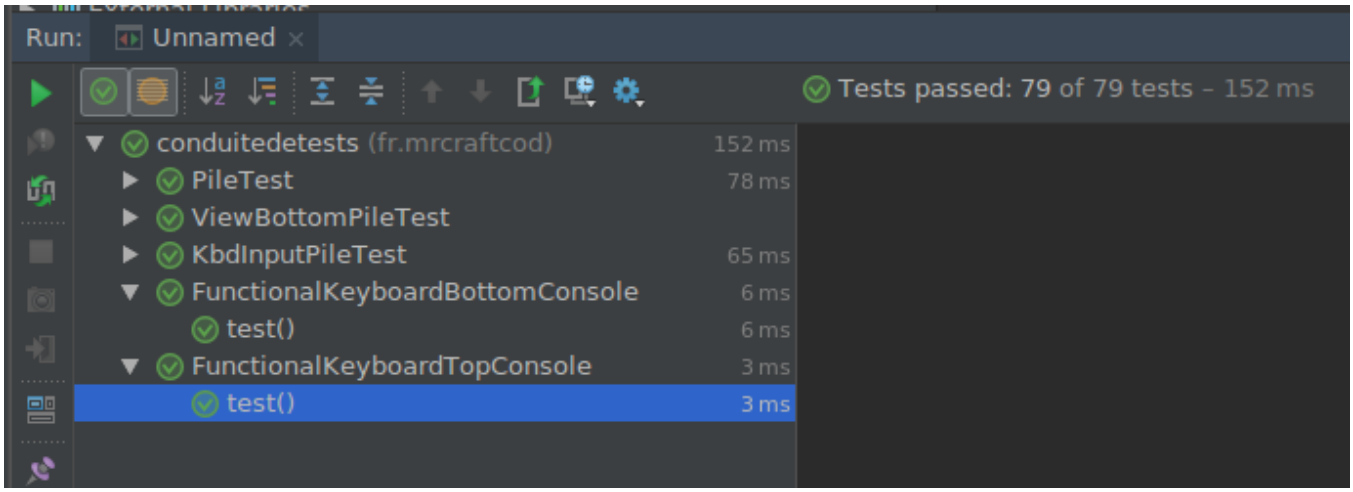

Figure 1.2 – Tests à cette étape

1.3 UI

Commit

Nous nous sommes par la suite intéressé à la partie UI. Cette fois-ci, nous avons réalisé les tests fonctionnels en premier. Nous avons donc mis des bouchons dans les nouvelles classes correspondantes:

1. ViewInputPile
2. ViewBottomPileUI

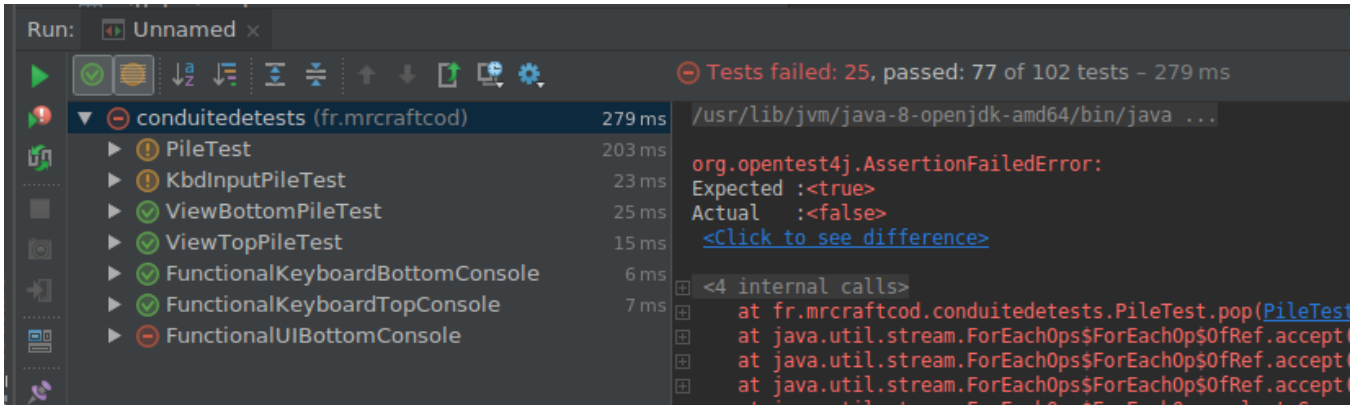

Figure 1.3 – Tests

Encore une fois les tests ne passent pas, mais cela est normal. (C'est à ce moment là que gitlab ci fail tout le temps à cause des nouvelles classes de javafx).

Nous avons donc par suite implémenté les différentes classes afin de valider les tests. En même temps nous avons enrichi les tests fonctionnels en testant avec un observer différent bouchonné (copié collé de l'autre observer): commit.

Enfin nous avons terminé de faire passer les tests au vert (commit).

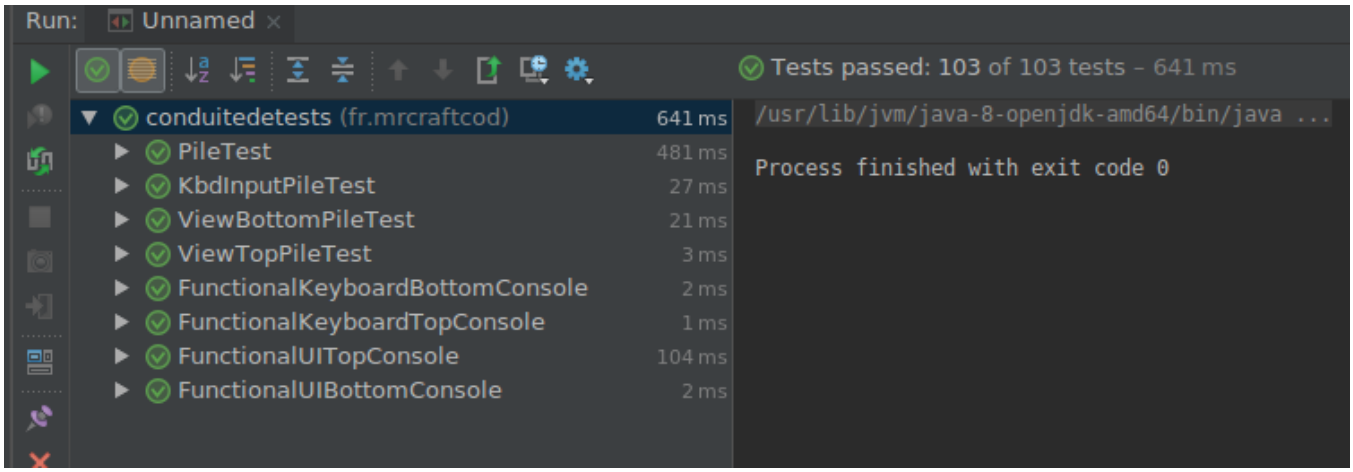

Figure 1.4 – Tests

1.4 Gitlab CI

Nous avons mis en place le CI dès le départ afin de build le code puis lancer les tests. Pour cela nous avons fait un ficher .gitlab-ci.yml. Il se base sur l'image docker comprenant Java JDK 8 et maven. Ce dernier comprends deux jobs:

1. build: fait parti du group build et permet de compiler le code grâce à maven

2. test: fait parti du groupe test (et donc dépend de build) et permet de lancer les tests grâce à maven.

Cela nous permet ainsi d'avoir un retour rapide sur l'état du code après chaque commit.

Cependant vers la fin du projet de nombreux build ont fail car l'image docker ne contenait pas les packages pour javafx.