

---

# **Analyse et Vérification des Logiciels**

## **Rapport projet**

---

Master 2 Génie Logiciel

Réalisé par : Les marmottes

- Ali ABIDAR
- Líticia MEDJOU DJ
- Nadia AMIRAT

Groupe : Groupe 1 GL

Année universitaire : 2021/2022

## 1. Qu'a-t-on testé ? Pourquoi ?

Toutes les méthodes qui implémentent la logique du jeu ont été testées.

Le fonctionnement du jeu dépend fortement du bon fonctionnement de celles-ci, il est alors impératif de les tester.

Pour une méthode donnée, on fait un test par branche d'exécution.

### Exemple:

Pour la méthode will invest de Cautious player on a 4 tests (un test par branche d'exécution).

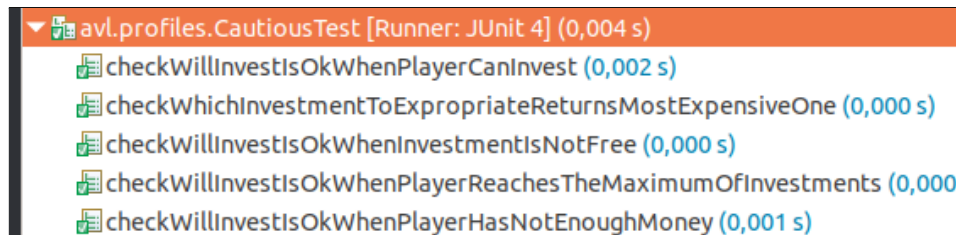


Figure 1 : Tests de la méthode willinvest d'un joueur prudent

## 2. Qu'a-t-on pas testé ? Pourquoi ?

### 2.1 Getters / Setters

On a pas testé les getters et setters car on a ajouté aucun contrôle à l'intérieur, ils ne contiennent aucune logique. On a utilisé des instructions java simples, le test n'a alors aucune valeur ajoutée.

### 2.2 Affichage (Instructions / résultats)

On n'a pas testé l'affichage car celui-ci affiche seulement le résultat de méthodes déjà testées. On a vérifié son fonctionnement lorsqu'on a testé les fonctionnalités de l'application (tests fonctionnels).

## 3. Qu'a-t-on mocké ? Pourquoi ?

On a mocké les méthodes :

- action de la classe Investment
- canPlay() de la classe Game

On a utilisé des mocks pour tester des méthodes qui dépendent d'autres composants complexes et aussi pour avoir un comportement déterministe sur lequel on se base pour nos tests. Ainsi on teste notre méthode indépendamment des autres.

## 4. Comment a t-on fait un code plus testable ?

On a mis en place une conception du jeu qui minimise les dépendances entre composants ce qui rend le code plus compréhensible et plus testable.

De plus, on a rencontré des difficultés à tester les inputs de l'utilisateur car les scanners étaient déclarés à l'intérieur des méthodes. Pour rendre le code testable on a dû le modifier pour mettre les scanners en paramètre.

## 5. Comment les tests sont-ils structurés ?

On a repris la même structure que le code, les tests sont mis dans des packages différents, ainsi on a utilisé de l'héritage entre les classes de tests quand c'est possible pour moduler ces derniers.

## 6. Autres remarques

### Choix liés à l'implémentation du jeu

Certains profils sont contradictoires, quand le joueur choisit un 2ème profil on lui propose que ceux qui **ne** sont **pas** contradictoires à celui qu'il a choisi en premier.

Exemple de profils contradictoire: Néolibéral et Socialiste

- Néolibéral : Certains joueurs ont beaucoup d'argent et d'autres en ont peu.
- Socialiste : Tous les joueurs ont autant d'argent au départ.

On a pas implémenté le profil COVID19 faute de temps.

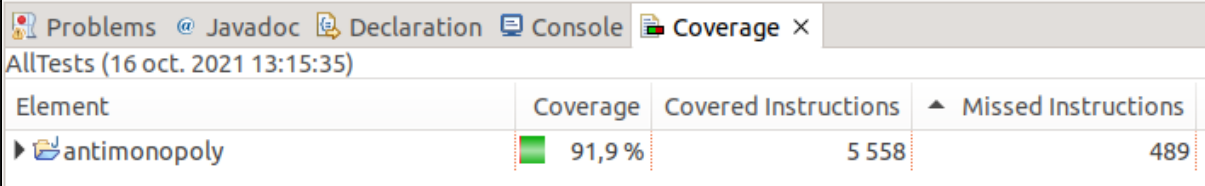
### Usage du TDD

Certains ont eu recours à cette méthodologie, d'autres non.

Ceux qui ont utilisé le TDD ont rapporté que c'est une technique qui permet de détecter les erreurs de code plus efficacement, par contre ça prend plus de temps, ce qui peut être expliqué par le fait que c'est une pratique à laquelle on est pas habitué.

### Coverage:

On a un coverage de tests de 91.9% (Figure 2), ce qui est un très bon pourcentage.



AllTests (16 oct. 2021 13:15:35)			
Element	Coverage	Covered Instructions	Missed Instructions
▶ antimonopoly	91,9 %	5 558	489

Figure 2 : Test coverage