Rapport V&V TP2

LELIEVRE Thomas, LELOUP Florian

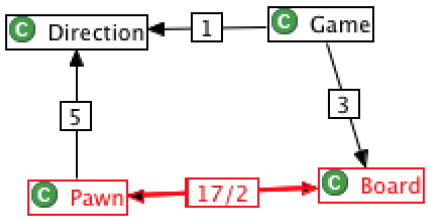
22 octobre 2013

Université Rennes 1 - ISTIC

Question 1

Use CodePro Analytix dependency analysis tool to visualize the dependency graph of the project. To be able to see all interesting classes in the same view, select both simpleGame.core and simpleGame.cli packages, do a right click, and do "Explore ! Contents". What do you notice?

On constate la présence d'un cycle de dépendances entre les classes *Pawn* et *Board* du package "*simpleGame.core"*.
On est donc dans l'obligation pour tester le projet efficacement, de faire un stub sur l'une de ces deux classes. En "simulant" le comportement d'un des deux objets on va alors pouvoir tester l'autre objet dépendant.



Question 2

Write a testing plan: in which order you would test the classes, which ones you plan to stub, etc.

Nous avons décidés d'appliquer le plan de test suivant :

- Etape 1
  Tester la classe *Pawn* avec le stub de *Board*
- Etape 2
  Tester la classe *Board* avec *Pawn*
- Etape 3
  Tester la classe *Pawn* avec la vraie classe Board
- Etape 4
  Tester la classe *Game* avec *Board*
- Etape 5
  Tester *CLIMain*

De la même manière on aurait pu choisir d'effectuer le stub sur la classe *Pawn*.

Question 3

Which classes should be mocked? Why?

D'après le diagramme de séquence, on observe deux interactions à vérifier :

- l'interaction entre la classe *Game* et *Board* via l'appel à la méthode isGameOver()
- l'interaction entre la classe *Board* et *Pawn* via l'appel à la méthode maxGold()

Nous avons donc décidés de mocker la classe *Board* pour l'appel à la méthode isGameOver(). Dans le deuxième cas on effectue un mock sur les deux instances de la classe *Pawn (pawns[0], pawns[1]).*

Question 4

For which reasons might you want to use Mockito methods *when* or *thenReturns*?

Dans le but d'effectuer le scénario décrit dans le diagramme de séquence, nous sommes dans l'obligation d'utiliser les méthodes de Mockito pour fournir les mocks aux différentes classes. Par exemple, on utilise la méthode *when (*"when(bo.maxGold()).thenReturn(3);") dans la méthode "testNumberOfPawns()" afin de vérifier le bon fonctionnement de l'interaction entre la classe Board et Game.On ne teste pas ici les fonctionnalités mais bien le bon déroulement du scénario, et ceci est uniquement possible grâce à l'appel de la méthode "when" de Mockito.

Notes sur les modifications apportés aux sources

Afin de tester plus efficacement la classe *Board*, nous avons créer des getters pour la liste des pawns, le pawn courant et les coordonnées de la case bonus.

De plus nous avons créer un nouveau constructeur pour *Board*, afin d'une part d'isoler la partie aléatoire du constructeur de base et d'autre part de permettre l'injection du mock de pawn dans le constructeur de *Board*.

Comme nous avons ajouté la possibilité de créer un *Board* avec une liste de pawns vide (le current pawn est donc à null), nous avons modifié la méthode addPawn afin que dans le cas où le currentPawn est null, il soit mis à celui qu'on vient d'ajouter.