

PROJET HEX

BOUIZEGARENE Hachimi

ESTRADE Yoan

BENYETTOU Imane

201

Table des matières

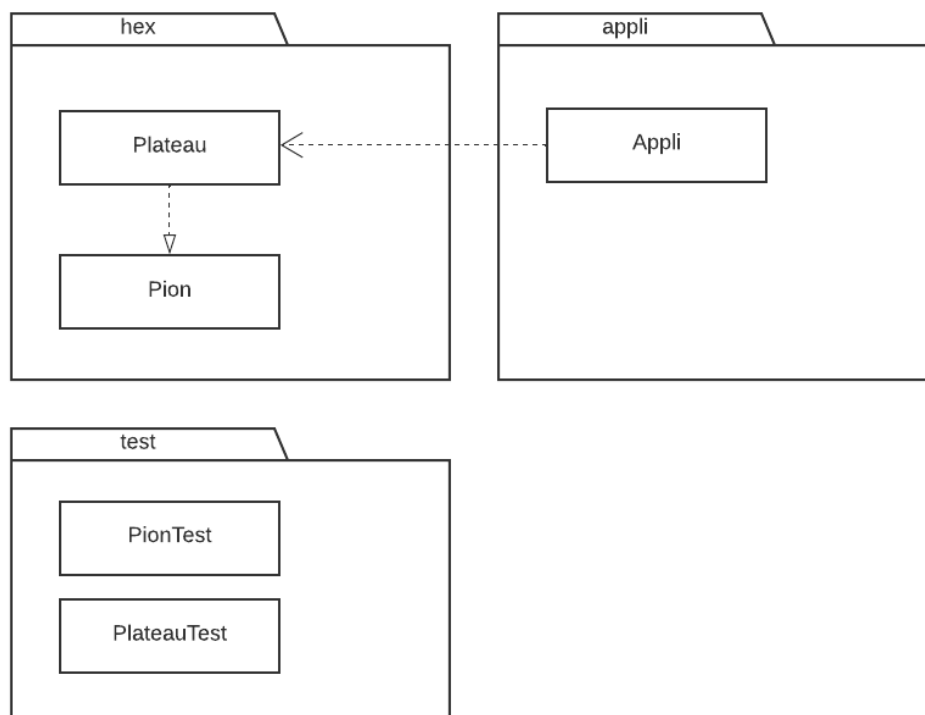
Table des matières	2
Introduction	3
Diagramme d'architecture	3
Liste des tests unitaires	4
Pour la classe Pion :	4
Pour la classe Plateau :	4
Explication des tâches à accomplir	4
Bilan du projet : Ce qui est réussi, ce qui peut être amélioré	5
Difficultés	5
Améliorations	5

Introduction

Ce projet consiste en le développement d'un jeu fonctionnel (hex), permettant à deux joueurs de s'affronter dans une partie complète, les joueurs pouvant être soit humain, soit un algorithme, avec des coups choisis aléatoirement à chaque fois. L'interface a été réalisé en mode texte plutôt qu'en mode graphique, par simplicité et gain de temps.

Le projet comporte plusieurs classes Pion et Plateau, ainsi qu'une classe principale Appli, comportant la fonction main.

Diagramme d'architecture



Liste des tests unitaires

Pour la classe Pion :

On vérifie que les symboles correspondent bien à la croix, au cercle et au vide avec la méthode `assertEquals()` et `assertTrue()`. Pour chaque symbole invalide (n'étant pas égal à 'X', 'O', ou '.', on le remplace par le caractère '**');

Pour la classe Plateau :

On vérifie que la taille du plateau est bien 4, et que son format est correct avec la méthode `assertEquals()`. On utilise cette même méthode pour vérifier que les joueurs sont bien des croix et rond, et pour vérifier que les méthodes de coordonnées (en dessous, au dessus, à gauche et à droite) marchent et correspondent bien avec les coordonnées voisines d'une case donnée.

On simule ensuite une partie où le rond gagne, en testant la méthode `jouer()`, et en vérifiant qu'il existe un chemin avec les méthodes `estCheminVersBas()` et `VerifPartie()`, puis une autre partie où la croix gagne, en testant de la même manière les mêmes méthodes.

Explication des tâches à accomplir

Pour recréer le jeu en java :

- Il faut tout d'abord une classe Pion, ayant pour attributs un symbole (croix, rond, ou vide) pour représenter les coups des joueurs
- Une classe Plateau ayant pour attributs une taille, un nombre de joueurs, et une première ligne et colonne pour définir les coordonnées des cases.
 - Le constructeur et la méthode redéfinie `toString()` pour formater l'apparence du plateau
 - La méthode `jouer()` pour gérer la partie (mouvement des pions et tour des joueurs
 - Les méthodes `coordEnDessous()`, `coordAuDessus()`, `coordADroite()` et `coordAGauche()`, qui renvoient respectivement la coordonnée sur chaque cardinalité. ainsi que `AllCoordsEnDessous()`, `AllCoordsAuDessus()`, `AllCoordsAGauche()` et `AllCoordsADroite()`, qui renvoient respectivement l'ensemble des coordonnées sur chaque cardinalité.

- Les méthodes `estCheminVersBas()` et `estCheminVersDroite()` pour tester l'existence d'un chemin vers la droite ou vers le bas, qui servira pour définir le gagnant de la partie
 - Et enfin la méthode `VerifPartie()`, qui détermine à l'aide des méthodes `estCheminVersLaDroite()` et `estCheminVersLeBas()` le gagnant de la partie.
-
- Des fichiers tests, pour effectuer les tests unitaires et vérifie que chaque méthode renvoi ce qu'elle est censée renvoyer (un fichier test pour le Plateau, et un pour le Pion)
 - Une classe `Appli` contenant le main, afin de pouvoir jouer les parties

Bilan du projet : Ce qui est réussi, ce qui peut être amélioré

Difficultés

Les principales difficultés rencontrées durant la réalisation de ce projet ont été de déterminer lorsque la partie est terminée, qui à remporté la victoire : nous avons mis du temps à trouver l'idée d'implémenter une fonction récursive pour déterminer l'existence de chemins, et donc la fin de la partie. Nous avons aussi eu du mal à appliquer à la lettre les principes SOLID, car pour chaque fonction, il fallait penser à si elle respectait ces principes, et la modifier si ce n'était pas le cas, ajoutant du travail supplémentaire.

Améliorations

Nous pensons que le code peut être optimisé, et rendu plus court : certaines fonctions pourraient être assemblées et différenciées par un paramètre, rendant le code plus lisible, ainsi que d'autres détails pouvant être améliorés.