BROUET Lucas, SIMON Gregory

Jeu du Gomoku

M2103

10 / 04 / 2021

Sommaire

1. Présentation du jeu, des classes, interfaces et énumérations

1.1 Le jeu

Le jeu du gomoku est un jeu de plateau stratégie qui oppose deux joueurs l'un contre l'autre. Chaque joueur à une couleur définie (soit blanche soit noire). Le but du jeu est d'aligner 5 cinq pions de la même couleur. Les pions peuvent être alignés soit en ligne soit en colonne soit en diagonale.

1.2 Les classes

Voici la liste des classes utilisés :

Board

Cette classe représente le plateau de jeu. Elle comporte un nombre de ligne et de colonne défini, ainsi qu'une liste de positions qui sont chacune associées à une couleur (noire, blanche ou rien). L'affichage du plateau (avec l'aide d'un tableau de caractère), ainsi que des test concernant le plateau (plein, vide etc...) sont réalisés par cette classe

Position

La classe *Position* est constituée d'une ligne et d'une colonne. Il est possible de construire une position à partir de caractères (comme "A5" par exemple), ou alors à partir de chiffres (ligne et colonne). De plus, cette classe se charge elle-même de se convertir en chaîne de caractère (dans le but d'être affichée).

Game

Cette classe concerne la partie en elle-même. Elle va principalement effectuer un coup qui lui est indiqué (si celui-ci est possible), mais vérifie également la victoire ou la partie nulle. La classe contient la couleur du joueur, le plateau de la partie, ainsi que la liste des coups joués.

Match

Cette classe étend la classe *Game*. Elle applique un match entre deux joueurs pour une taille de plateau défini grâce à la méthode *run*.

IAChoice

Définit le choix d'un joueur machine. Dans notre cas, il s'agit du hasard, mais cela peut être amélioré.

HumanPlayer

Implémente l'interface Player. Demande au joueur de choisir la position (sous forme de chaîne de caractère).

RobotPlayer

Implémente l'interface Player. Demande à l'IA de choisir une position (celui choisi grâce à la classe IAChoice).

GamePlayerLeaves / InvalidCoordonates

Classes qui héritent de la classe "Exception". GamePlayerLeaves est jetée lorsqu' un joueur écrit "/quit" et invalidCoordonate lorsque le joueur inscrit des coordonnées invalides (comme "AA" ou "C2D" par exemple).

1.3 L’interface

Player

Cette interface représente un joueur de manière générale, et s’occupe du choix de celui-ci concernant la position.

1.4 Les énumérations

Color

Représente les couleurs du jeu : noir, blanc, ou rien (concernant un pion ou un joueur).

Direction

Représente les 8 directions cardinales. Elle permet la sélection de deux directions opposées (utile pour tester la victoire).

2. Choix effectués pour le code

2.1 Match et Game

Nous avons décidé que la classe *Match* étendait la classe *Game*. En effet, un match correspond à une partie entre deux joueurs, et utilise donc les attributs et méthodes de *Game* car c’est la classe qui contient les informations sur l’état d’une partie. Nous avons donc lié ces deux classes afin de ne pas recopier du code et de ne pas avoir toutes les méthodes de la classe game public alors que les autres classes n’en ont pas besoin.

2.2 Le plateau

Tout d'abord, le plateau comporte un nombre fixe de lignes et de colonnes (on les met en final), puis un attribut static correspondant à la première lettre (pour l'affichage de lettres représentant les colonnes).
Nous avons ensuite décidé que c'était le plateau qui possédait les pions des joueurs, et non une position. Ainsi, la classe Plateau possède premièrement un tableau associatif de position et de couleur (Map<Position, Color> colors = new HashMap<>()), puis un tableau de charactère afin d'afficher un pion.

2.3 Taille minimale du plateau

Nous avons décidé de mettre une taille minimale de plateau afin de pouvoir jouer. Normalement la taille minimale est soit 5x4 soit 4x5, mais afin d'avoir des parties plus intéressantes nous avons mis une taille minimale de 10x10.

2.4 Taille maximale du plateau

A l'instar de la taille minimale, nous avons limité la taille maximale; limite qui est de 26x26. Nous avons décidé de placer cette limite tout simplement pour éviter de lever des exceptions car notre code nous permet seulement de choisir une colonne comprise entre la lettre A et Z, on ne peut pas choisir de caractères spéciaux.

2.5 Condition de victoire

Celle-ci se calcule à partir de la dernière position jouée. Pour chaque direction et son opposée (direction complémentaire) (Nord / Sud, Est / Ouest etc..), on additionne chaque pion de la même couleur que le joueur

courant (tant que l'on ne sort pas du plateau). Si la somme est supérieure ou égale à 5, alors il y a victoire.

2.6 Condition de match nul

Comme tout jeu de plateau sans restriction de tour, il se peut que le plateau soit rempli et qu'aucun des joueurs n'ait rempli la condition de victoire. Nous avons donc ajouté une condition de partie nulle qui met fin au match.

3. Remarque

- Le projet fournit correspond à la demande dans son intégralité
- Tests unitaires intéressants testant la majeur partie des méthodes utilisées
- Absence d'un menu au démarrage du jeu (il faut ajouter/changer du code dans la méthode *main*)