ABDERRAHMANI Wijdan
PASCAL Arthur

Rapport de projet
Gomoku

Sommaire

Dans le cadre d'un projet en Programmation Orientée Objet (M2103), nous, ABDERRAHMANI Wijdan et PASCAL Arthur, élèves en S2A', avons codé un Gomoku en Java sur Netbeans. Les règles que nous devions respecter, étaient les suivantes :

- Les joueurs posent des pions à tour de rôle dans une case vide.
- Doivent jouer à côté d'une case déjà occupée, sauf au premier coup.
- Le gagnant est le premier qui aligne 5 pions selon une ligne, colonne, ou diagonale.
- La partie est nulle si toutes les cases sont occupées.

I Répartition des responsabilités entre les classes :

Notre projet est constitué de 11 classes et 5 exceptions, réparties dans 6 paquets différents.

1) Paquet Coordonnees :

La classe Directions renvoie des directions dans un tableau. Ses directions sont verticales, opposées, horizontales et diagonales. Cette classe nous permet aussi de voir le nombre de cases parcourues horizontalement lorsque l'on avance dans cette direction (0 pour Nord et Sud, -1 pour Ouest et 1 pour Est) et de même verticalement (0 pour Est et Ouest, -1 pour Nord ainsi que 1 pour le Sud).
La classe Position possède un constructeur de Position, chaque Position est constitué d'une ligne et d'une colonne. Elle permet aussi de vérifier si une position est jouable en vérifiant la non-présence d'un pion sur cette case. Elle renvoie aussi une liste de positions voisines à une position donnée et dans un tableau de direction donné ainsi qu'une distance à parcourir. Elle renvoie aussi la liste des positions voisines, et nous indique qu'une case possède une voisine dans une liste de positions. Elle permet aussi de vérifier si une position p est vérifiable et si elle est bien comprise entre 0 et la taille maximale du plateau. Elle permet aussi de comparer deux positions.

2) Paquet Grille :

La classe Match possède un constructeur de match. Chaque match possède X nombres de colonnes et Y nombres de lignes. Cette classe permet aussi de signaler la possibilité de pose (si le coup est dans les dimensions du plateau et si la pose est bien à côté d'une case déjà jouée).
La classe Partie possède un constructeur de partie. Chaque partie est constituée de la couleur du prochain joueur et d'un plateau. Cette classe permet aussi l'actualisation du plateau (à partir de la dernière position jouée, de la couleur du dernier joueur ayant joué et d'un match). Cette classe permet aussi d'indiquer si 5 cases alignées sont de la même couleur, d'effectuer un tour particulier lors de la première pose ainsi que d'alterner le tour de chaque joueur en fonction de leurs couleurs lors d'un match
La classe Plateau possède une méthode permettant d'associer une couleur a une position ainsi qu'une méthode permettant de récupérer la couleur d'une position ainsi que d'afficher le plateau visuellement dans le tableau de jeu.
3) Paquet Utilitaire:

La classe UtilisGomo permet de transformer une lettre en chiffre (0 pour a,1 pour b etc …), de lire la prochaine ligne saisie par l'utilisateur, de renvoyer un string associé à une couleur, d'associer une couleur lors d'une saisie de ligne lors d'un match, l'entier associé aux X du match. L'entier associé aux Y du match.
4) Paquet Gomoku :

La classe Gomoku, est la classe principale permettant l'affichage et la fluidité du jeu.
5) Paquet utilisateur :

La classe Couleur, énumère les couleurs et renvoie la couleur des pions.
La classe Joueur est une classe abstraite.
La classe JoueurHumain possède un constructeur pour un joueur. Chaque joueur possède un nom et une couleur (noir ou blanc). Cette classe permet aussi de récupérer à partir d'une chaîne de caractère, la position choisie par un joueur.
La classe JoueurRobot possède un constructeur d'une IA. Cette classe permet aussi de récupérer la position jouée par l'IA.

6) Paquet Exception :

La classe ExceptionHorsDuPlateau qui possède un constructeur de l'exception qui gère le cas où la position est hors du plateau.
La classe ExceptionMauvaiseEntree qui possède un constructeur qui gère le cas où l'entrée n'est pas une position.
La classe ExceptionPasVoisin qui possède un constructeur de l'exception qui gère le cas où il n'y a pas de voisins.
La classe ExceptionPositionDejaPose qui possède un constructeur de l'exception qui gère le cas où la position est déjà prise.
La classe ExceptionQuitter qui possède un constructeur de l'exception qui gère le cas où le joueur rentre "/quit".
La classe ExceptionMauvaiseEntree qui possède un constructeur de l'exception qui gère le cas où le joueur rentre une mauvaise entrée

II Choix effectués pour la programmation :
Couleur :
Nous avons décidé de créer une "énumération" pour permettre d'avoir accès plus facilement aux attributs et car nous n'avions pas besoin de méthode particulière pour cette classe

Directions :
De même pour directions nous avons choisi une "énumération" car il y avait une longue liste de directions bien qu'il y a quelques méthodes.

Gomoku :
Étant la classe principale, nous avons décidé de la simplifier le plus possible pour rendre sa lecture la plus fluide possible . L'utilisation du switch est là pour éviter un if, else if … trop long et simplifier le code

Joueur :
Cette classe est une interface abstraite. En effet, celle-ci va implémenter les deux classes suivantes : JoueurHumain et JoueurRobot. Nous avons décidé de la rendre abstraite car elle ne sera jamais instancié seule et c'est une interface car les deux classes qui l'implémente auront la même méthode "choix" qui leur sera commune tant bien sur les paramètres que sur les exception levées.

JoueurHumain :
Implémente la l'interface abstraite vue au-dessus car à la méthode "choix".

JoueurRobot :
De même pour JoueurRobot.

Match :
Est une classe qui sera utilisée dans la classe plateau car elle contient des données essentielles aux dimensions de ce dernier ainsi qu'une méthode jouable qui sera utilisée.

Plateau :
Comme dit précédemment ,nous avons choisi de mettre en attribut un match à cette classe plateau car les deux sont reliés. Nous avons aussi décidé de faire un tableau de Positions à double entrées (de taille du plateau) pour la liste des positions car nous avons trouvé qu'il était plus facile

d’associer une ligne et une colonne à une position par ce biais. De plus, cela nous a permis d'accéder à toutes les positions de ce plateau.

Partie :
De même pour cette classe nous avons décidé d’associer un plateau à la partie car les deux sont liés. Nous avons également choisi une ArrayList pour stocker la liste des coups car ceux-ci ne peuvent être présents qu’une seule fois.

Position :
Cette classe nous a permis de gérer tout ce qui était en rapport avec les positions donc à savoir les voisines etc … Nous avons décidé de stocker dans une liste et non pas dans un tableau la liste des voisines car cela est plus facile à manipuler tant bien pour ajouter que pour accéder aux différentes données.

UtilisGomo :
Cette classe nous a permis de gérer tout ce qui était de l’ordre du “général” et nous avons décidé de vérifier ici les exceptions de saisies plutôt que dans les autres classes car cela permet une meilleure lisibilité dans ces mêmes classes.

En conclusion, nous sommes satisfaits de notre code. Notre code respecte les consignes posées par nos enseignants. Effectivement, les joueurs jouent à tour de rôle en posant seulement dans une case vide et près d’une case déjà occupée (sauf lors du premier coup). Le gagnant est notamment le premier alignant 5 pions de la même couleur sur une ligne, colonne ou encore diagonale. La partie est aussi nulle si toutes les cases sont occupées.
Nous avons ainsi pu approfondir nos connaissances en Programmation Orientée Objet, notamment grâce à ce projet.