Gomoku

Rapport de projet

Candau Sébastien
S2A”

Sommaire:

1. Présentation

1.1 Le jeu

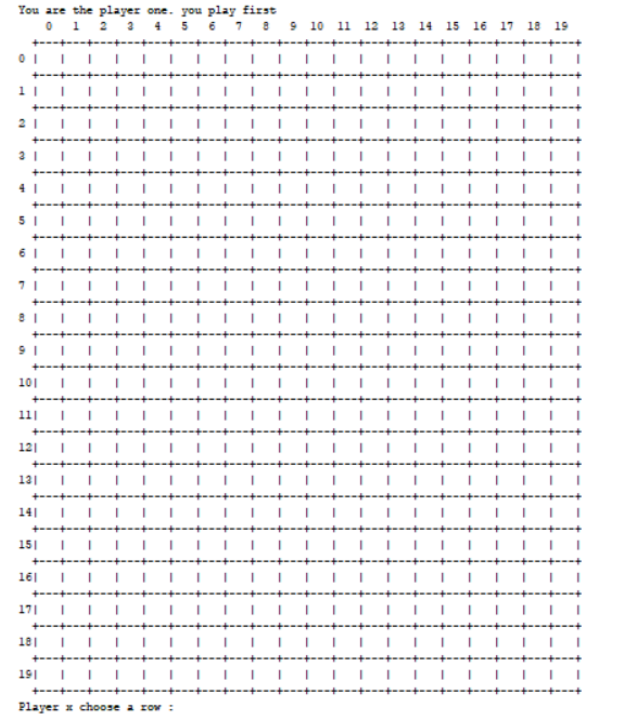
Le Gomoku se joue à deux joueurs. Ils doivent placer à tour de rôle un pion d'une couleur assignée dans une grille. Le joueur plaçant 5 de ses pions cote à cote, que ce soit en diagonale ou alignés, gagne la partie.

1.2 Les modes de jeu

Le jeu comporte deux modes. Ils sont appelés directement lors du lancement d'une nouvelle partie.

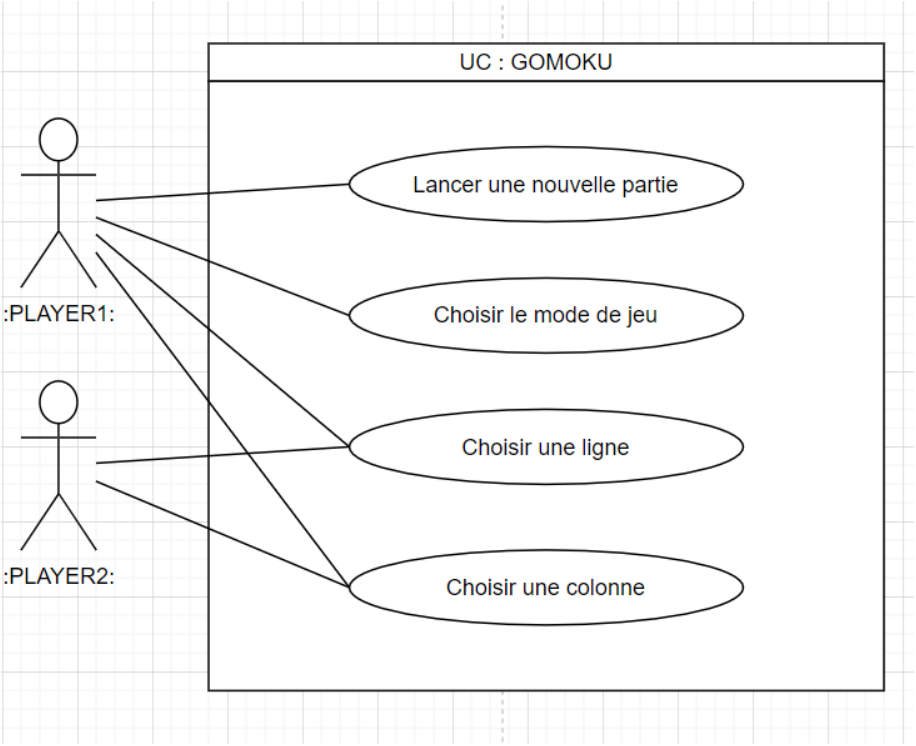
```
Hello, welcome to Super Gomoku !
Please choose a game mode :
1. -> two player modes
2. -> One player versus AI
```

Le joueur doit alors choisir le mode de jeu sur lequel il veut jouer. Le premier mode est un mode à deux joueurs dans lequel deux personnes réelles s'affrontent, le second mode est un mode se jouant seul, le joueur affronte une IA.

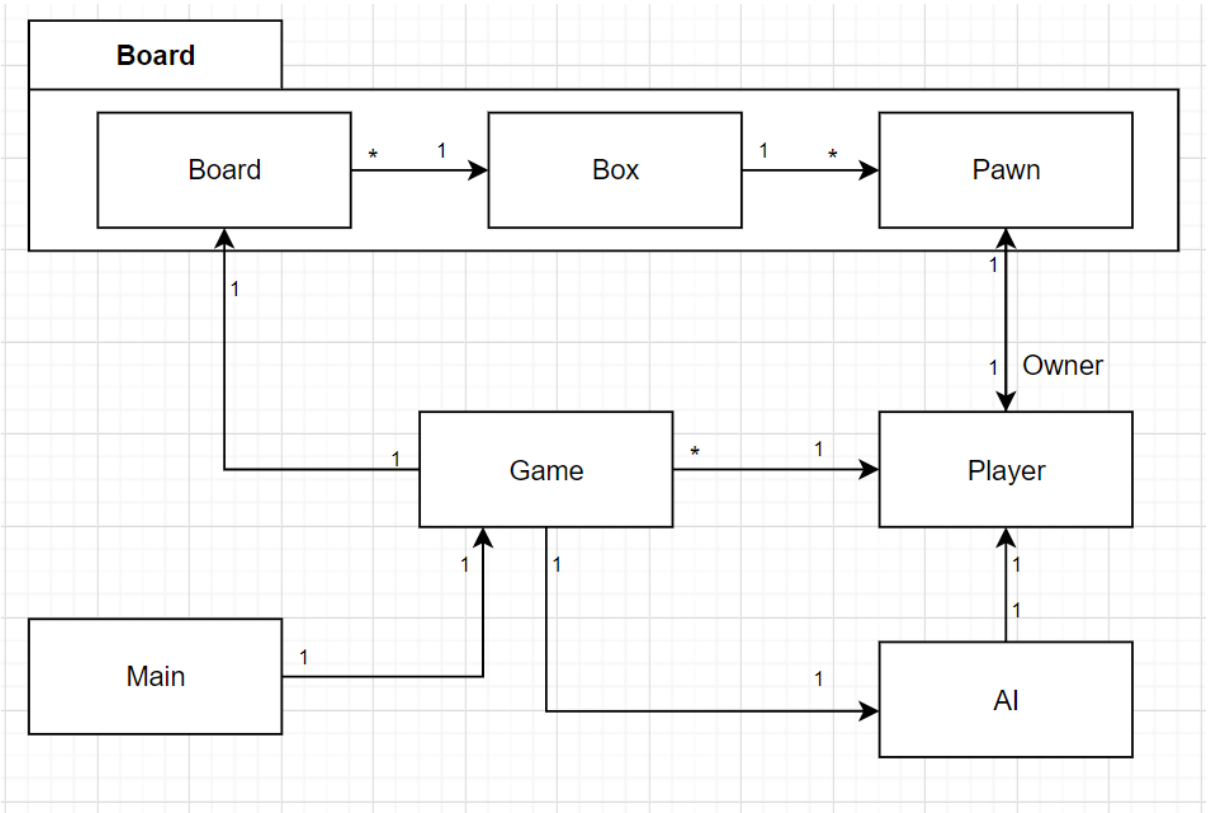


2. Conception du projet

2.1 Les cas d'utilisation



2.2 Les différentes classes et liens entre elles



La classe main permet de créer une nouvelle partie : Game. Les différents choix de jeu sont alors lancés par la méthode menu();

Les méthodes start() permettent la création d'un nouveau tableau de jeu : Board qui sera rappelé et mis à jour à chaque tour de jeu.
La classe board représente donc un tableau à deux dimensions composé de cases: Box. Sa méthode showBoard permet l'affichage de celui-ci dans la console.
Chaque case a pour contenu un pion. Il est représenté par un caractère propre à un joueur. Si la case n'est pas jouée et est dite "vide", le pion la composant est représenté par un caractère de type espace.
Chaque pion a un propriétaire de type Player. Tous les joueurs ont un pion Pawn assigné.
Si le mode IA est choisi dans le menu, c'est l'IA qui prendra le rôle du joueur numéro 2.

3. La programmation du jeu

3.1 Déroulé d'une partie

Le jeu fonctionne à partir d'une suite événementielle composée des différents objets.
Premièrement le tableau de notre jeu est affiché par la fonction printBoard()

```
private void printBoard() {
    board.showBoard();
}
```

Ensuite il est demandé à l'utilisateur d'entrer la ligne puis la colonne dans laquelle il veut jouer. La classe InputOutput permet de convertir la chaîne de caractères entrée et de gérer les exceptions.

```
public class Input {

    /**
     * allows you to transform the string entered by the player into an integer.
     *
     * @param scan
     * @return
     */
    public static int readIn(Scanner scan) {
        String input = scan.nextLine();
        int result = -1;
        try {
            result = Integer.parseInt(input);
        } catch (NumberFormatException e) {
            return result;
        }
        return result;
    }

}
```

Certaines autres exceptions sont gérées par la fonction vérification.

```
private void verification(int input) {
    if (input < 0 || input >= 20) {
        System.err.println("Error -> Please choose a valid number");
        startTwoPlayers();
    } else if (Integer.toString(input).isEmpty()) {
        System.err.println("Error -> Please choose a valid number");
    }

}
```

On met ensuite à jour notre tableau en lui ajoutant le nouveau pion aux coordonnées indiquées.

On vérifie si il y a un gagnant avec haveWinner, si il y en a un, la boucle se coupe et le gagnant est affiché.
Sinon, on appelle la méthode swapPlayer permettant de changer de joueur et on revient au début de la boucle.

```
private void swapPlayer() {
    if (this.currentPlayer.getId() == this.player1.getId()) {
        this.currentPlayer = this.player2;
    } else {
        this.currentPlayer = this.player1;
    }
}
```

3.2 L'affichage

L'affichage de notre tableau doit se faire sous forme de chaîne de caractères. Le but ici est d'afficher l'ensemble du contenu de notre tableau, c'est-à-dire des cases sous forme de caractères. C'est le rôle de la méthode showBoard()

```
    /**
     * @return the representation in String of an empty box and a filled box.
     */
    @Override
    public String toString() {
            return "| " + this.content.toString() + " ";
        }
}
```

```
    /**
     * Method allowing the construction of a game board
     */
    public void showBoard() {
        System.out.print("    0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19\n");
        System.out.print("  ");
        for (int i = 0; i < 20; i++) {
            System.out.print("+---");
        }
        System.out.print("+");
        for (int j = 0; j < 20; j++) {
            System.out.println("");
            if (j < 10) {
                System.out.print(j + " ");
            } else {
                System.out.print(j);
            }
            for (int i = 0; i < 20; i++) {
                System.out.print(grid[j][i].toString());
            }
            System.out.print("|");
            System.out.println("");
            System.out.print("  ");
            for (int i = 0; i < 20; i++) {
                System.out.print("+---");
            }
            System.out.print("+");
        }
    }
}
```

3.3 L'IA

L'IA codé est extrêmement simple. Tellement simple qu'il ne serait pas intéressant pour un joueur moyen de l'affronter. Elle peut être utile pour apprendre aux enfants à jouer par exemple.

l'IA ici codée se base sur du simple hasard. Elle prend le dernier pion joué et se contente de choisir au hasard une case voisine non vide pour poser son pion.

```
public void AIPlay(int i, int j, Board board, int min) {
    Random random = new Random();
    int max = 1 + random.nextInt(8);
    Random random2 = new Random();
    int nb;
    nb = min + random2.nextInt(max);
    if (nb == 1) {
        System.out.println("Alo");
        bottomBox(i, j, board);
    }
    ...
```

Le but dans le futur sera d'améliorer cette IA, de façon à ce qu'elle puisse imiter le joueur afin de le bloquer.

4. Le code

4.1 La taille du code

Nombre de classes : **8**

Main:
-main **1** fonction, **0** constructeur, **26** lignes

Game:
-Game : **7** fonctions, **1** constructeur, **153** lignes
-AI : **10** fonctions **1** constructeur **177** lignes

Board:
-Board: **4** fonctions **1** constructeur **141** lignes
-Box: **9** fonctions **1** constructeur **112** lignes
-Pawn: **3** fonctions **1** constructeur **53** lignes

Player:
-Player: **2** fonctions **1** constructeur **34** lignes

-InputOutput:
-Input : **1** fonction **0** constructeur **34** lignes

Total de lignes: **730**
Total de fonctions : **37**