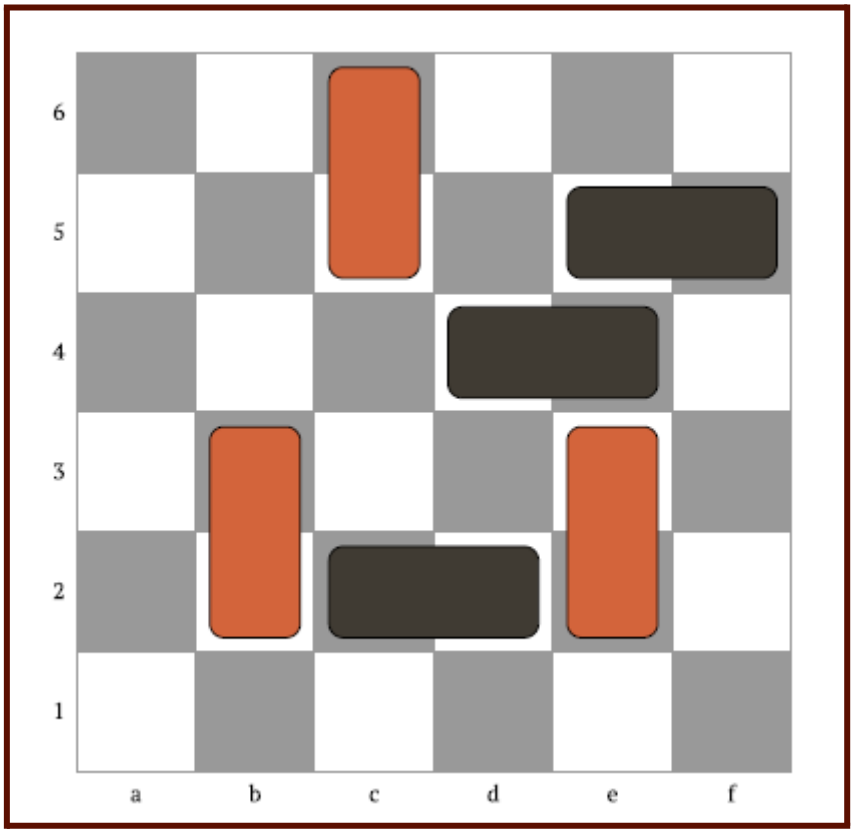
Université Abdelmalek Essaadi
Faculté des sciences et techniques de Tanger
Département génie informatique
Cycle d'ingénieur Logiciels et systèmes intelligents « LSI »
Année universitaire 2023/2024

Méthodologies de l'intelligence artificielle

"Domineering Game"



Réalisé par :

DAAOUAN MOHAMMED
FRIKH SAID

Encadré par :

Pr.M'HAMED AIT KBIR

Année universitaire: 2023-2024

Table des matières

Introduction

1. Options de Jeu
2. Exécution dans le console
   - 2.1. Joueur Vs Joueur
   - 2.2. Sauvegarder la partie
   - 2.3. Rejouer la partie sauvegardée
3. Interface Utilisateur
4. La hiérarchie du projet
5. Algorithmes et heuristiques
6. Explication du code
   - 6.1. La classe "Domineering"
   - 6.2. La classe abstraite "GameSearch"
   - 6.3. La classe abstraite "DomineeringMove"
   - 6.4. La classe abstraite "DomineeringPosition"
   - 6.5. Méthodes et Fonctionnalités
     - 6.5.1. Méthode playGame
     - 6.5.2. Méthode playGameP2P
     - 6.5.3. Méthode possibleMoves
     - 6.5.4. Méthode saveGame
     - 6.5.6. Méthode loadGame
     - 6.5.7. Méthode playLoadedGame

Conclusion

Introduction:

Dans le cadre de ce projet, l'objectif principal était de développer une application Java mettant en œuvre le jeu du Domineering, également connu sous le nom de Stop-Gate ou Crosscram. Ce jeu oppose deux joueurs (HUMAIN contre PROGRAMME ou HUMAIN contre HUMAIN), qui, tour à tour, placent des dominos sur un plateau rectangulaire. L'application intègre des algorithmes de recherche adversariale, tels que Minimax et Alpha-Bêta, ainsi que des heuristiques innovantes et intelligentes. Ces éléments ont été soigneusement sélectionnés pour offrir une expérience de jeu interactive et stimulante.

1. Options de Jeu:

Les joueurs ont la possibilité de demander de l'aide à la machine un nombre fini de fois (deux fois) pendant la partie.

La complexité du jeu peut être ajustée pour répondre aux préférences de chaque joueur.

La stratégie de la machine peut également être définie par l'utilisateur, offrant un contrôle sur le niveau de défi souhaité.

Une fonctionnalité essentielle de l'application est la possibilité de sauvegarder une partie en cours et de la reprendre ultérieurement. Cette fonctionnalité offre aux joueurs une flexibilité précieuse, leur permettant de prendre des pauses sans perdre leur progression. De plus, la possibilité de rejouer des parties sauvegardées offre une opportunité d'analyser les stratégies passées et d'améliorer les performances futures.

2. Exécution dans le console:

```
1: Player To Player.
2: Player To AI.
3: Load Saved Game.
1
Board position:
0000
0000
0000
0000
Player 1's move:
Do you want help from the program? (yes/no)
no
Entrer la position de la tête du domino comme (e.g., 0 1):
1 1
Board position:
0000
0H00
0H00
0000
Player 2's move:
Do you want to save the game? (save) or Do you want help from the program? (yes/no)
no
Entrer la position de la tête du domino comme (e.g., 0 1):
0 0
Board position:
PP00
0H00
0H00
0000
Player 1's move:
Do you want help from the program? (yes/no)
yes
Board position:
PP00
0H00
0H0H
000H
Player 2's move:
Do you want to save the game? (save) or Do you want help from the program? (yes/no)
yes
Board position:
PP00
0HPP
0H0H
000H
Player 1's move:
Do you want help from the program? (yes/no)
no
Entrer la position de la tête du domino comme (e.g., 0 1):
1 0
Board position:
PP00
HHPP
HH0H
000H
Player 2's move:
Do you want to save the game? (save) or Do you want help from the program? (yes/no)
save
Game saved successfully.
Game saved. Exiting.

Process finished with exit code 0
```

❖ Rejouer la partie sauvegardée:

```
1: Player To Player.
2: Player To AI.
3: Load Saved Game.
3
Board position:
PP00
HHPP
HH0H
000H
Player's move:
Do you want help from the program? (yes/no)
```

❖ Joueur Vs Programme(AI):

```
1: Player To Player.
2: Player To AI.
3: Load Saved Game.
2
Board position:
0000
0000
0000
0000
Your move:
Do you want help from the program? (yes/no)
no
Entrer la position de la tête du domino comme (e.g., 0 1):
1 1
 next element: 1.0
 next element: [
  0, 0, 0, 0
  0, 1, 0, 0
  0, 1, -1, -1
  0, 0, 0, 0
]
Board position:
0000
0H00
0HPP
0000
Your move:
Do you want help from the program? (yes/no)
```

```
Board position:
0000
0HPP
HHPP
H000
Your move:
Do you want help from the program? (yes/no)
no
Entrer la position de la tête du domino comme (e.g., 0 1):
0 0
 next element: 4.0
 next element: [
  1, 0, 0, 0
  1, 1, -1, -1
  1, 1, -1, -1
  1, 0, 0, 0
]
Board position:
H000
HHPP
HHPP
H000
Program won

Process finished with exit code 0
```

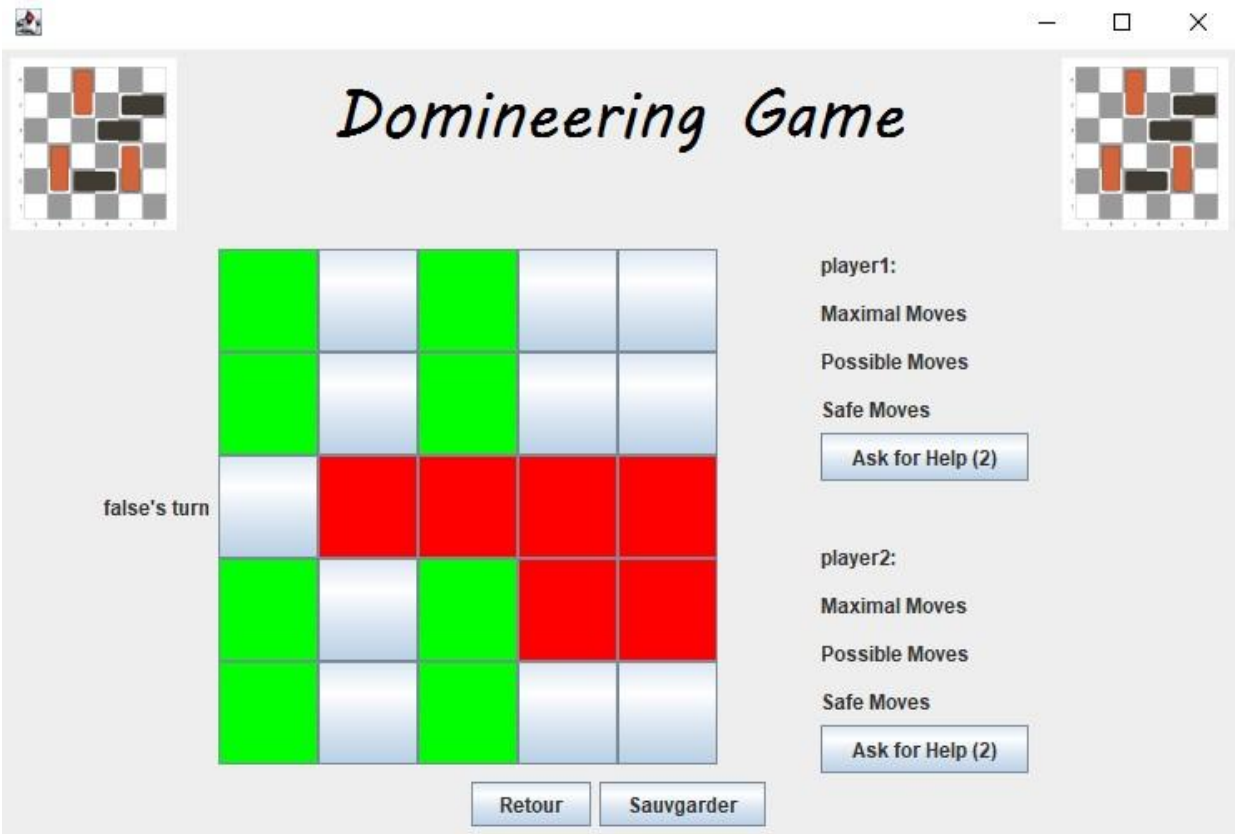
3. Interface Utilisateur:

- ❖ L'interface utilisateur de notre application offre une expérience conviviale et intuitive.
- ❖ Les boutons permettant de fixer la complexité, la taille du plateau, et de choisir l'adversaire (HUMAIN ou PROGRAMME) sont des éléments clés de l'interaction.
- ❖ Le bouton de fixation de la complexité permet aux utilisateurs de définir le niveau de difficulté du jeu, tandis que le paramètre de taille du plateau offre une personnalisation flexible.
- ❖ Le choix de l'adversaire permet aux utilisateurs de décider s'ils souhaitent jouer contre un adversaire humain ou contre la machine.

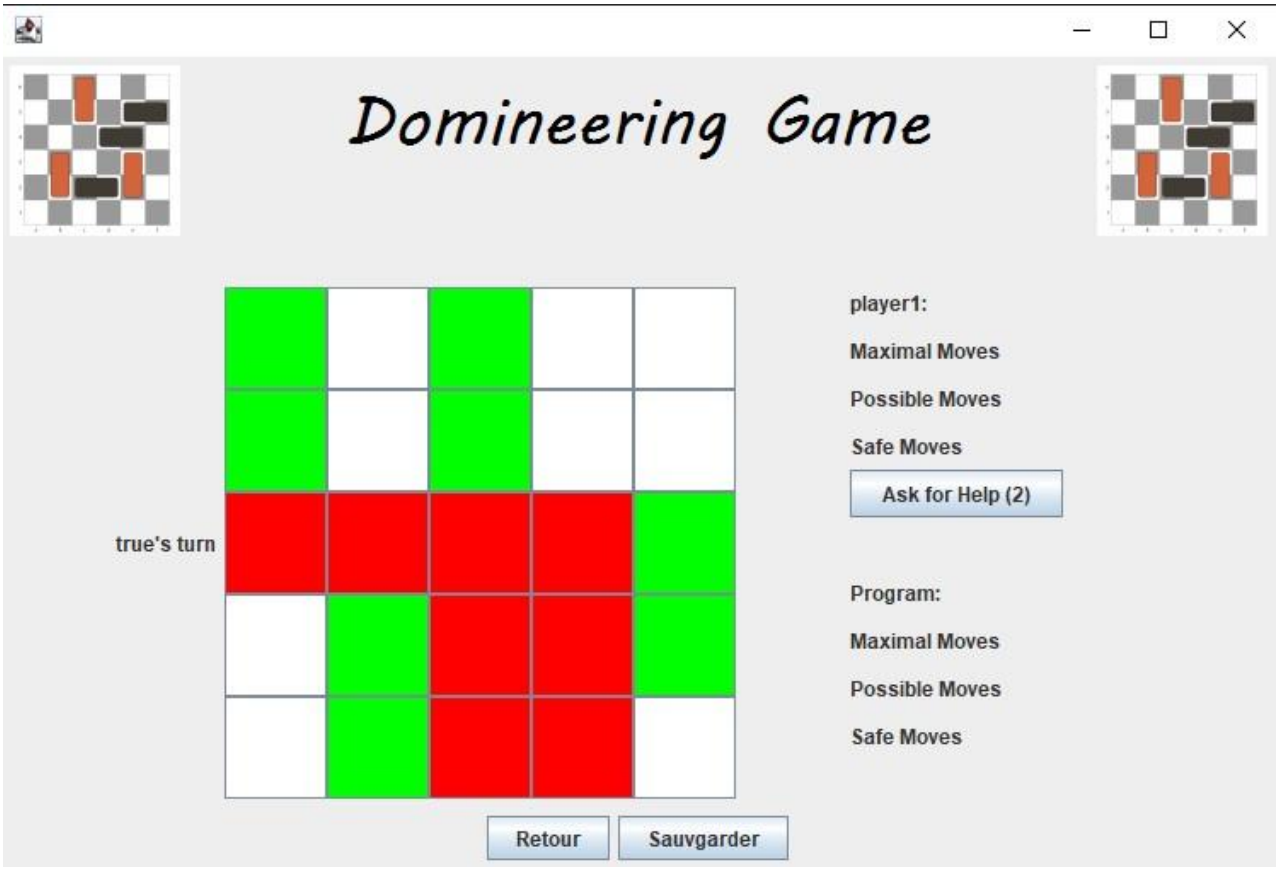

Domineering Game
Size:
5
Player 1:
Human
Player 2:
Human
Play

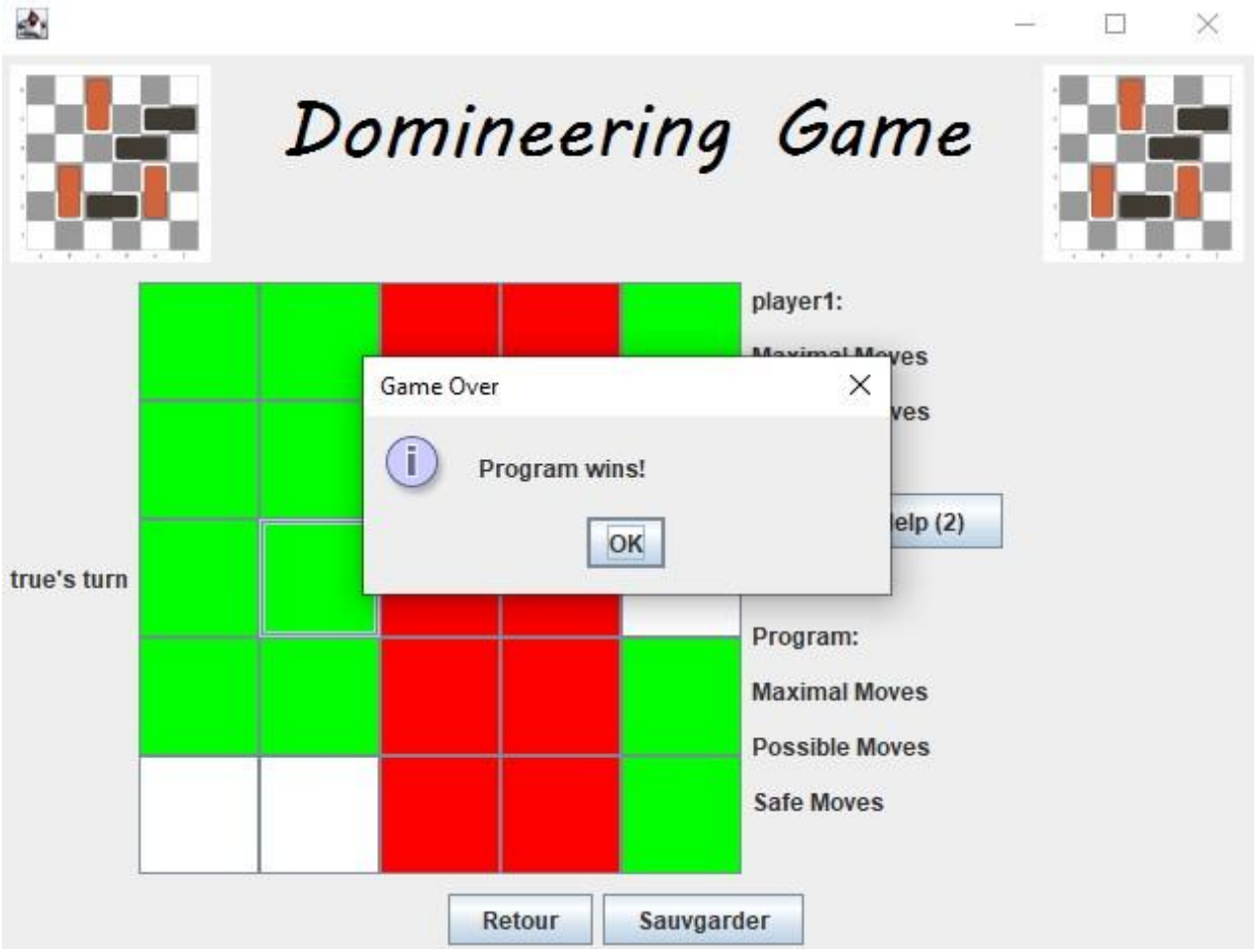

Domineering Game
Size:
5
Player 1:
Human
Player 2:
Human
Human
Program: Level1
Program: Level2
Program: Level3
Play

Joueur 1 Vs Joueur 2:



Joueur Vs Programme(AI):





4. La hiérarchie du projet:

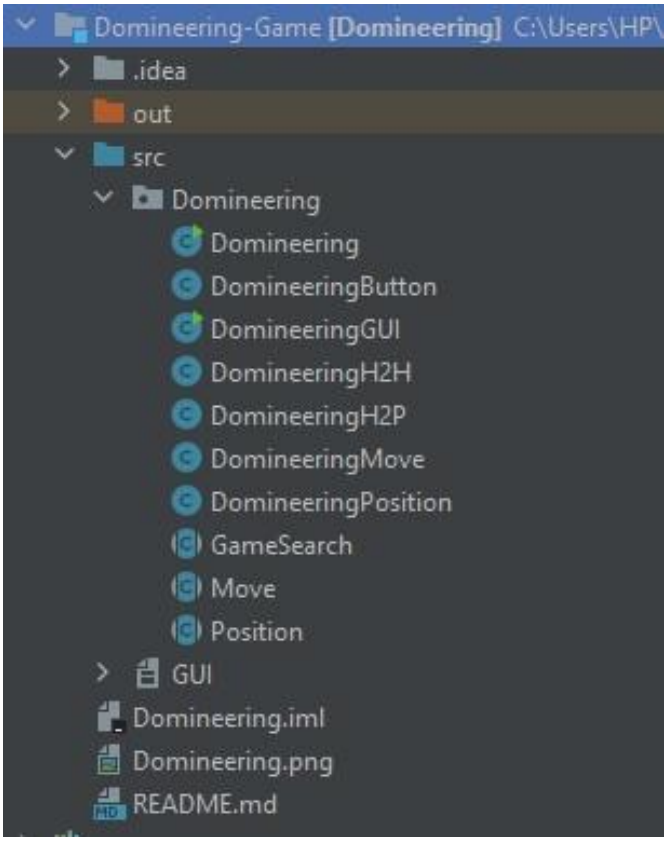
La structure du code est soigneusement organisée afin de le rendre organisé, lisible et adaptable aux évolutions futures.

La classe principale, nommée "Domineering", assume la responsabilité de la gestion globale du jeu, orchestrant le flux des tours et des interactions entre les joueurs. En parallèle, des classes additionnelles sont dédiées à des aspects spécifiques du jeu, tels que les mouvements (`DomineeringMove`), la logique des positions "DomineeringPosition", et les algorithmes de recherche adversariale "GameSearch".

Chaque classe est soigneusement conçue pour être modulaire, contribuant ainsi à une meilleure compréhension du code et facilitant sa maintenance.

La classe principale interagit avec les classes spécialisées en utilisant des méthodes définies dans ces dernières, contribuant ainsi à une

séparation claire des responsabilités. Cette approche modulaire favorise la réutilisabilité du code et offre la flexibilité nécessaire pour apporter des améliorations ou étendre les fonctionnalités du jeu à l'avenir. La clarté de la hiérarchie des classes et la modélisation réfléchie des entités du jeu contribuent à rendre le code organisé, lisible et adaptable aux évolutions futures.



5. Algorithmes et heuristiques:

L'application intègre l'algorithme Minimax avec Alpha-Beta Pruning, offrant ainsi une méthode de prise de décision éclairée lors des parties de Domineering. Cet algorithme de recherche adversariale permet à la machine de simuler différentes séquences de mouvements pour anticiper les conséquences à court et long terme. L'utilisation du mécanisme de coupure Alpha-Beta contribue à optimiser le processus de recherche en éliminant les branches d'exploration non prometteuses, améliorant ainsi l'efficacité globale de l'algorithme.

Parallèlement, l'application intègre des heuristiques qui influent sur la stratégie adoptée par la machine. Ces heuristiques guident la prise de décision en évaluant la valeur relative des positions de jeu et en orientant la machine vers des choix plus avantageux. La stratégie spécifique de la machine peut être configurée lors du lancement du jeu, permettant ainsi une

personnalisation selon les préférences de l'utilisateur.

En particulier, la méthode "alphaBetaHelper" a été adaptée pour refléter la stratégie Alpha-Beta décrite dans le cours, contribuant ainsi à une performance optimale même pour des plateaux de jeu de grande taille.

6. Explication du code:

5.1. La classe "Domineering":

La classe **"Domineering"** constitue le cœur de l'application. Elle gère la logique du jeu, les interactions avec les joueurs, et l'utilisation des algorithmes de recherche adversariale.
Cette classe utilise les méthodes définies dans la classe abstraite **"GameSearch"** pour implémenter les algorithmes Minimax avec Alpha-Beta Pruning. Elle coordonne également les différentes phases du jeu, de l'initialisation à la fin de la partie.

La méthode **"main"** est le point d'entrée du programme. Elle crée une instance de **"DomineeringPosition"** pour représenter l'état initial du plateau, puis instancie Domineering pour lancer le jeu. L'utilisateur peut choisir entre un mode joueur contre joueur gèré par la methode **"playGameP2P"** ou joueur contre machine gèré par la methode **"playGame"**.
Ces méthodes font appel à des fonctions telles que **"minValue"** et **"maxValue"** définies dans la classe GameSearch pour prendre des décisions éclairées.

5.2. La classe abstraite "GameSearch":

La classe abstraite "GameSearch" définit les méthodes abstraites nécessaires pour la recherche adversariale. Les méthodes **"minValue"** et **"maxValue"** implémentent l'algorithme Minimax avec Alpha-Beta Pruning, tandis que d'autres méthodes abstraites (**positionEvaluation**, **drawnPosition** et **makeMove**) définissent les opérations fondamentales comme évaluer une position, générer des mouvements possibles, effectuer un coup, etc.
Cette classe sert de base pour les jeux implémentant des algorithmes de recherche adversariale.
Les constantes PROGRAM et HUMAN déterminent les rôles des joueurs, facilitant l'adaptation à d'autres jeux.

5.3. La classe abstraite "DomineeringMove":

La classe DomineeringMove représente un mouvement dans le jeu du Domineering. Elle contient les coordonnées d'un domino placé sur le plateau, avec les informations sur la rangée (row) et la colonne (col).

5.4. La classe abstraite "DomineeringPosition":

La classe DomineeringPosition représente l'état actuel du jeu. Elle utilise une matrice pour modéliser le plateau de jeu. La méthode toString permet d'afficher la position de manière lisible.

5.5. Méthodes et Fonctionnalités:

Les méthodes **"playGameP2P"** et **"playGame"** sont des composants essentiels de l'application, orchestrant le déroulement des parties joueur contre joueur et joueur contre machine, respectivement. Ces méthodes intègrent des fonctionnalités spécifiques pour garantir une expérience de jeu interactive et stimulante.

- **Méthode playGameP2P:**

La méthode playGameP2P gère les parties entre deux joueurs humains. Elle assure la séquence des tours entre le joueur 1 et le joueur 2, avec la possibilité pour chaque joueur de demander de l'aide à la machine. Cette aide est limitée à deux demandes par joueur, apportant un équilibre entre l'autonomie du joueur et la possibilité de bénéficier de conseils stratégiques. Les boucles while(true) garantissent que le jeu continue jusqu'à ce qu'une condition de fin soit rencontrée, telle que la victoire d'un joueur. Ces boucles assurent une interaction continue avec les joueurs.

- **Méthode playGame:**

La méthode playGame prend en charge les parties où un joueur humain affronte la machine. Elle offre au joueur la possibilité de demander de l'aide à la machine, similaire à la méthode précédente.

En somme, ces méthodes offrent une structure robuste pour la gestion des parties, assurant un équilibre entre l'autonomie des joueurs et la possibilité de recevoir des conseils stratégiques limités de la machine. La boucle continue garantit une expérience de jeu immersive et interactive.

❖ **Méthode PossibleMoves :**

Le code de la fonction PossibleMoves dans la classe Domineering vise à déterminer le nombre de mouvements possibles pour un joueur donné dans une position spécifique du jeu Domineering.
La fonction prend en paramètres un objet de type Position et un booléen player, représentant le joueur pour lequel nous souhaitons calculer les mouvements possibles. La fonction explore chaque cellule du plateau de jeu, vérifiant la possibilité de placer un domino soit horizontalement (pour le joueur H) soit verticalement (pour le joueur P).

Si un placement est possible à une position donnée, le compteur PossibleMoves est incrémenté, et la fonction simule le placement du domino en mettant à jour temporairement le tableau du plateau.

Cette approche permet d'évaluer les différentes configurations possibles sans altérer la position réelle du jeu. La fonction renvoie ensuite le nombre total de mouvements possibles pour le joueur spécifié dans la position actuelle du jeu.

❖ **Méthode saveGame:**

La fonction saveGame de la classe Domineering a pour objectif de sauvegarder l'état actuel du jeu dans un fichier sérialisé. Les paramètres de la fonction incluent la position initiale du jeu (startingPosition), des informations sur les joueurs tels que le tour du joueur 1 (player1Turn) et le nombre d'aides utilisées par chaque joueur (player1HelpCount et player2HelpCount).

À l'intérieur du bloc try avec ressources, la fonction crée un objet ObjectOutputStream pour sérialiser les données dans le fichier "saved_game.ser". Elle écrit ensuite la position initiale, les informations sur le tour et le nombre d'aides pour chaque joueur.

❖ **Méthode loadGame:**

La fonction loadGame a pour objectif de charger une partie sauvegardée à partir du fichier "saved_game.ser". Elle utilise un objet ObjectInputStream pour désérialiser les données du fichier. Les informations sur la position, le tour du joueur, et le nombre d'aides pour chaque joueur sont lues à partir du flux.

En cas de succès, la fonction appelle ensuite la méthode playLoadedGame

pour reprendre la partie en utilisant les données chargées. En cas d'erreur pendant le processus de lecture ou de désérialisation, les exceptions IOException et ClassNotFoundException sont capturées et leur traçage est affiché.

- **Méthode playLoadedGame:**

  La méthode playLoadedGame est une méthode utilisée par la méthode loadGame pour reprendre une partie sauvegardée. Elle prend en compte les paramètres chargés tels que la position initiale, le tour du joueur, et le nombre d'aides utilisées par chaque joueur.
  À l'aide d'une boucle while, le jeu est repris jusqu'à ce qu'une condition de fin de partie soit atteinte (victoire d'un joueur).

Conclusion:

En conclusion, notre application Java propose une expérience complète du jeu Domineering, intégrant des algorithmes de recherche adversariale avancés et des options de personnalisation étendues.
Les choix de conception, y compris l'utilisation d'heuristiques modifiables et la prise en compte des préférences des joueurs, ont été faits dans le but d'offrir une expérience de jeu riche et adaptative.