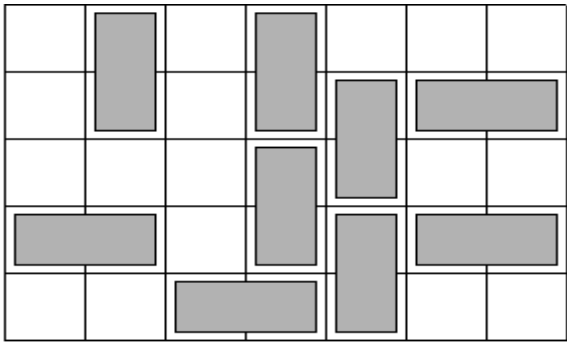
Département de Génie Informatique Cycle d’ingénieur Logiciels & Systèmes intelligents

METHODOLOGIE DE L’INTELLIGENCE ARTIFICIELLE



Domineering game

Réalisé par:
KAISSI Houda
Essalhi Sara

I.But:

Développement d ’une application Java pour le jeu" Domineering" en implémentant l’algorithme alphaBeta.

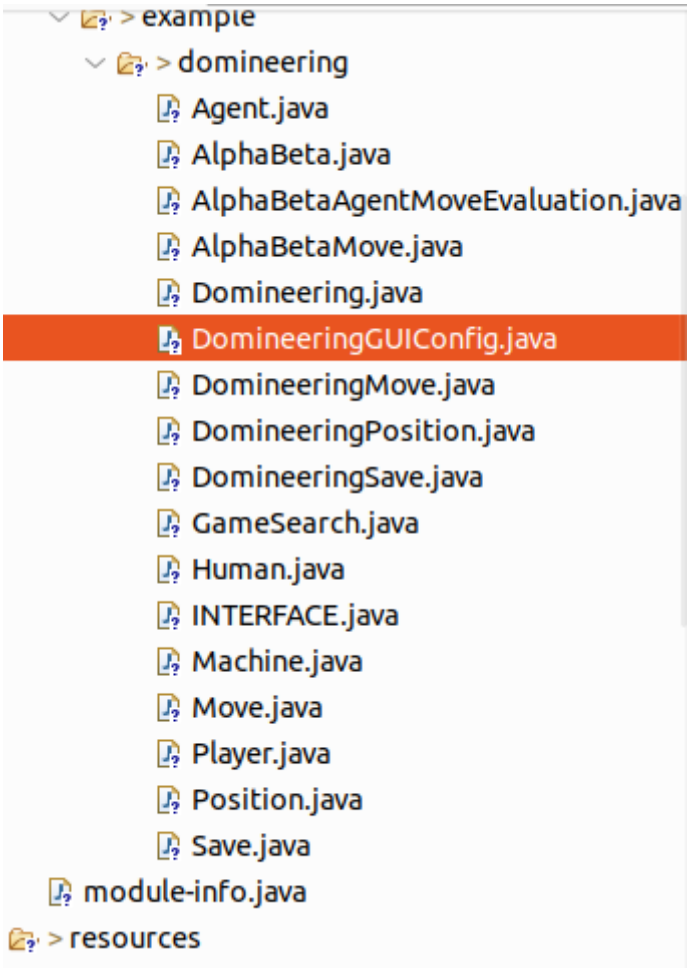
II.Introduction

Le jeu de Domineering est un jeu de société abstrait qui se joue à deux joueurs sur une grille rectangulaire. Il appartient à la famille des jeux combinatoires purs, ce qui signifie qu'il se déroule en alternant les tours entre les joueurs, et le résultat dépend uniquement des choix stratégiques des joueurs, sans élément de hasard.

La particularité du jeu de domineering réside dans sa simplicité apparente, mais qui cache des défis tactiques intéressants. La grille de jeu est divisée en cellules rectangulaires, et chaque joueur place alternativement des dominos (pièces de deux cellules adjacentes) de manière à dominer les cases de la grille. L'objectif est de forcer l'adversaire à ne pas pouvoir jouer, soit en bloquant complètement son accès à la grille, soit en contrôlant de manière stratégique les espaces disponibles.

III.Code

La hiérarchie du projet:



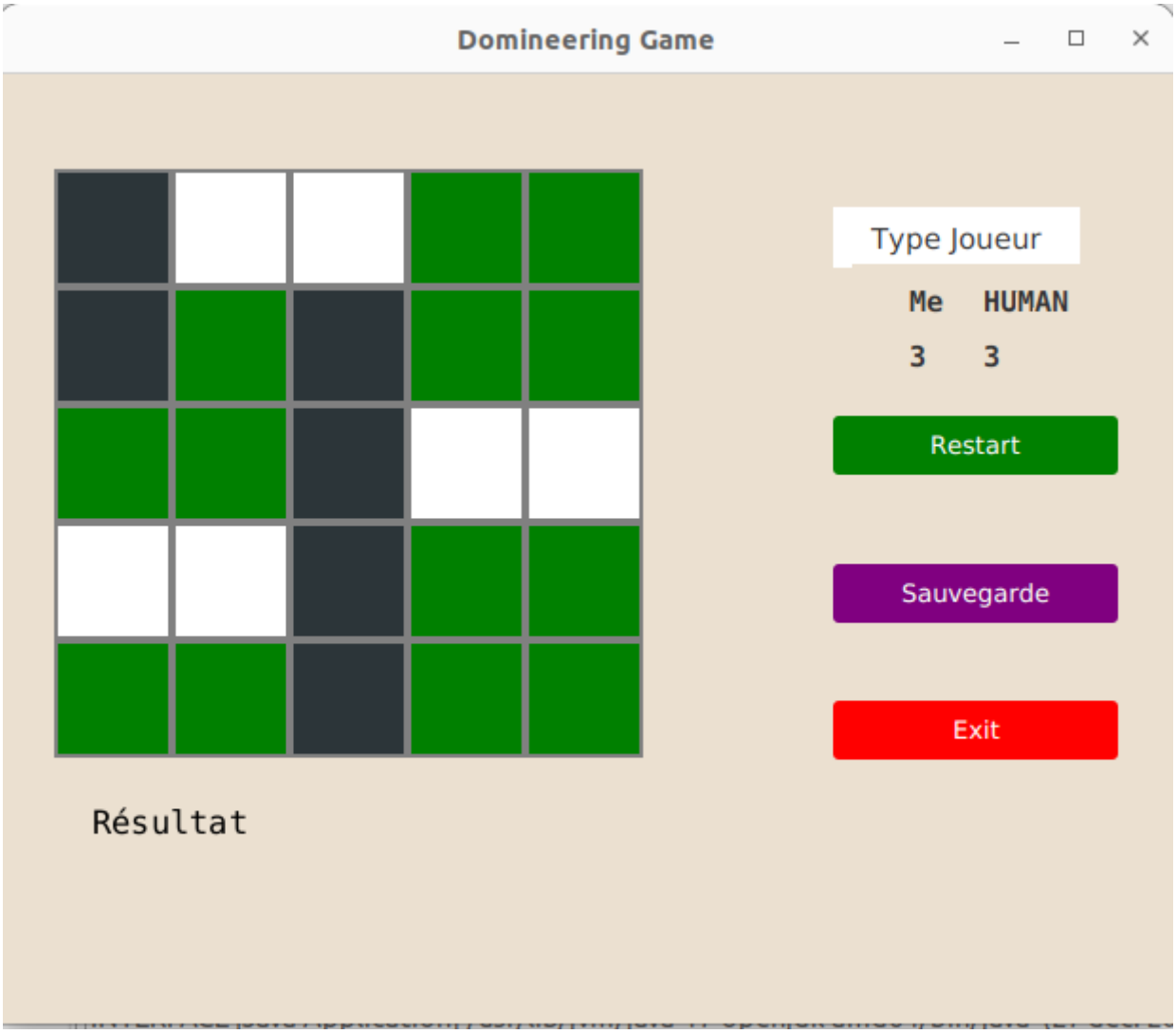
l’algorithme Alpha-Beta

L'application est dotée d'une approche stratégique avancée grâce à l'algorithme Minimax et à la technique de coupure Alpha-Beta Pruning, offrant ainsi une prise de décision réfléchie lors des parties de Domineering. Cet algorithme de recherche adversariale permet à la machine d'explorer diverses séquences de mouvements, anticipant ainsi les conséquences à court et long terme. L'efficacité globale de l'algorithme est considérablement améliorée grâce à l'utilisation judicieuse du mécanisme de coupure Alpha-Beta, qui élimine les branches non prometteuses de l'exploration.

En parallèle, l'application intègre des heuristiques sophistiquées 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, permettant à la machine de choisir des options plus avantageuses. La personnalisation de la stratégie de la machine est une caractéristique distinctive, offrant aux utilisateurs la possibilité de configurer la stratégie selon leurs préférences lors du lancement du jeu. Cela assure une expérience de jeu adaptée et engageante, où la machine s'ajuste aux préférences individuelles de chaque utilisateur.

VI.Interface

1.Choisir le niveau de difficulté.
2.Jouer une nouvelle partie.
3.Choisir la manière du jeu (Humain-Machine ou Humain-Humain).
4.Enregistrer une partie achevée
5.Annuler un coup joué.
6.Visualiser son score ainsi quele score de son adversaire.
7.Se déconnecter.
8.Quitter le jeu.



Explication du code:

Gamesearch

```java
package com.example.domineering;

public abstract class GameSearch {


    public abstract boolean wonPosition(Position p);


    public abstract AlphaBetaMove[] possibleMoves(AlphaBetaMove p, int player);

    public abstract Move getNeighbourMove(Position position, Move move, int player);

    public abstract Move makeMove(Position gamePosition, GameSearch gameSearch);
}
```

la classe abstraite GameSearch définit des méthodes génériques pour la recherche de solutions dans le contexte du jeu de Domineering. Les détails spécifiques de la mise en œuvre de ces méthodes sont laissés à des classes concrètes qui étendent cette classe abstraite

Class Interface

```java
import javafx.scene.text.Font;

public class INTERFACE extends Application {

    // the state of the game
    private final Position gamePosition = new DomineeringPosition(Player.HUMAN);
    private final Label adversaryLabel = new Label(gamePosition.getCurrentPlayerType().toStrir
    private final Move movePlayer1 = new DomineeringMove((BOX_SIZE * gamePosition.getNumSquare
    private final Text resultText = new Text("Résultat");
    private final Text resultText1 = new Text("Domineering Game");
    private Label movesPlayer1Label;
    private Label maxPossibleMovesPlayer1Label;
    private Label movesPlayer2Label;
    private Label maxPossibleMovesPlayer2Label;
    private final Move movePlayer2 = new DomineeringMove((BOX_SIZE * gamePosition.getNumSquare
    //     game search
```

cette classe représente l'interface graphique d'un jeu, mais pour obtenir une vue plus complète, il serait nécessaire d'examiner

d'autres parties du code qui définissent les actions des boutons et la logique du jeu.

Class alphabeta

```java
package com.example.domineering;
import javafx.scene.control.Alert;

public class AlphaBeta extends Agent {

    private static final int MAX_DEPTH = 5;



     @Override
    public Move makeMove(Position gamePosition, GameSearch domineeringGameSearch) {
        // transform position to board
        // -1 = program, 1 = human, 0 = blank
        AlphaBetaMove alphaBetaAgentMove = new AlphaBetaMove(gamePosition);


        AlphaBetaMove alphaBetaAgentMoveResult = alphaBeta(
```

Le code est une implémentation de l'algorithme Alpha-Bêta pour jouer au jeu de Domineering.
Domineering est un jeu à deux joueurs où les joueurs placent tour à tour des dominos horizontaux ou verticaux sur une grille. L'objectif est de créer un chemin connecté d'un côté de la grille à l'autre.

Class Machine

```java
package com.example.domineering;

import java.util.List;

public class Machine extends Agent {
    @Override
    public Move makeMove(Position gamePosition, GameSearch domineeringGameSearch) {
        List<Move> unPlayedMoves = gamePosition.getGridPane().getChildren().stream().filter(noc
            Move neighbourSquare = domineeringGameSearch.getNeighbourMove(gamePosition, (Move)
            return !node.isDisable() && neighbourSquare != null && !neighbourSquare.isDisable()
        }).map(node -> (Move) node).toList();

        if (!unPlayedMoves.isEmpty()) {
            return unPlayedMoves.get((int) (Math.random() * unPlayedMoves.size()));
        }
        return null;

    }
}
```

Le code fourni est une implémentation de la classe Machine , qui étend la classe abstraite Agent . La classe Machine est destinée à représenter une entité automatisée prenant des décisions dans le contexte du jeu Domineering.

Class Domineering

```java
package com.example.domineering;

import com.example.domineering.Agent.*;

public class Domineering extends GameSearch {


    @Override
    public boolean wonPosition(Position position) {
        int numSquares = position.getNumSquares();
        for (int row = 0; row < numSquares; row++) {
            for (int col = 0; col < numSquares; col++) {
                Move move = position.getSquare(row, col);
                Move neighbourMove = getNeighbourMove(position, move, position.getCurrentPlaye
                if (move != null && !move.isDisable() && neighbourMove != null && !neighbourMo
                    return false;
            }
        }
```

la classe Domineering sest un composant central pour la gestion des règles du jeu de Domineering, fournissant des fonctionnalités telles que la vérification des positions gagnantes, la génération de mouvements possibles, et la sélection de mouvements en fonction du joueur actue

class DomineeringSave

```java
package com.example.domineering;

import javafx.scene.control.Alert;

public class DomineeringSave extends Save {
    @Override
    public void saveGame(Position position) {

        // saveGame function (using a file.txt)

        // Create a FileChooser object
        FileChooser fileChooser = new FileChooser();

        // Set the title of the FileChooser
        fileChooser.setTitle("Save Game");

        // Set the initial directory
        fileChooser.setInitialDirectory(new File("."));

```

Le code est une implémentation de la classe DomineeringGame, qui étend la classe abstraite Save. Cette classe est conçue pour gérer la sauvegarde du jeu de Domineering dans un fichier texte et afficher des alertes correspondantes. La fonction principale est saveGame, qui sauvegarde l'état actuel du jeu dans un fichier texte.

VII Conclusion

Cette application est basée sur l’algorithme MinMax et la méthode d’élagage AlphaBeta, qui constituent la base du moteur d’intelligence artificielle de tous les jeux de réflexion tel que Domineering. Le principe de recherche d’un coup suivant ces méthodes repose sur le développement d’un arbre de jeu dont les feuilles ont une note obtenue par l’utilisation d’une fonction dévaluation