Mini-Projet

Exercice 1 (Gomoku)

On se propose ici de réaliser une version du jeu « gomoku ». Ce jeu se joue sur une grille de 15 lignes et 15 colonnes. Le but est d'aligner cinq jetons de la même couleur, en ligne, colonne ou diagonale. Les joueurs disposent initialement de 60 jetons chacun. Le premier coup se fait obligatoirement sur la case centrale. Chaque joueur joue ensuite à tour de rôle, en ajoutant un jeton sur une case de son choix, avec la contrainte qu'elle doit être libre et adjacente (en 8-connexité) d'une case déjà occupée.

La grille sera représentée par une matrice doublement chaînée en 8-connexité, c'est-à-dire que chaque case contiendra une référence vers ses 8 cases voisines Cette structure est illustrée figure 1.

En particulier, une case doit maintenir la cohérence du maillage de ses voisines. Par exemple, dans l'exemple figure 2 page suivante, le seul ajout de la case D comme voisine (en diagonale) de B doit associer B comme voisine de D (réciproque), mais également A comme voisine de D (en ligne) et C comme voisine de D (en colonne), et réciproquement.

Une case maintient aussi la présence éventuelle d'un jeton.

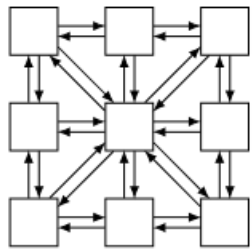

FIGURE 1 – Illustration des liens entre les cases d'une grille 3×3

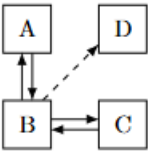

FIGURE 2 – Illustration des liens entre les cases d'une grille 3×3

Question 1 (Structure) *:* Créer les différentes classes impliquées dans la structure décrite précédemment. Vous ajouterez les différentes méthodes jugées utiles. Créer une méthode d'initialisation de la grille. On pourra par exemple utiliser une énumération pour les directions et une `EnumMap`[1] pour stocker les voisins.

Question 2 (Vérification) *:* Ajouter aux cases une méthode vérifiant si elle est dans un alignement de 4 jetons de la même couleur qu'elle-même. Vous pourrez utiliser judicieusement la récursivité.

Question 3 (Affichage) *:* Définir une méthode d'affichage de la grille dans un terminal. Vous pourrez utiliser les symboles unicodes ainsi que les codes d'échappement ANSI pour les couleurs.

Question 4 (Généralisation) *:* Rendez paramétrable la dimension de la grille, le nombre de jetons de départ et le nombre de jetons à aligner pour gagner.

Question 5 (Programme principal) *:* Écrire un programme principal qui permet à deux joueurs ayant donné leur nom et choisi leur couleur de jouer alternativement. Le programme affichera l'état courant de la grille, puis lira le numéro de la ligne et de la colonne où le joueur courant veut positionner son jeton, tant qu'il n'y a pas un gagnant et qu'il reste des jetons.

Les paramètres de jeux (dimension, etc.) seront lus sur la ligne de commande.

Question 6 (Sauvegarde et restauration) *:* Créer deux méthodes symétriques `save(Path)` et `load(Path)`, permettant d'enregistrer l'état courant de la partie dans un fichier texte d'une part, et de recharger un état précédemment enregistré d'autre part.

Question 7 (Bonus) *:* Implémentez une intelligence artificelle de jeux, afin de pouvoir jouer contre l'ordinateur. Vous pouvez utiliser des stratégies naïves au coup par coup. Vous pouvez aussi essayer de construire l'arbre des possibilités afin de choisir le meilleur coup possible à chaque tour.

Question 8 (Bonus 2) *:* Rendez le jeu infini : le nombre de jetons à disposition n'est pas limité. De plus, quand un jeton est positionné en limite de grille, une nouvelle ligne ou colonne est automatiquement ajoutée pour étendre la grille.

1. https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/EnumMap.html#