Une image contenant texte, capture d’écran, livre, cercle

Description générée automatiquement

TABLE DE MATIERES

[INTRODUCTION 3](#_Toc181466996)

[PROGRAMMES INDENTES ET COMMENTES 4](#_Toc181466997)

[Main.java 4](#_Toc181466998)

[Joueur.java 6](#_Toc181466999)

[JEU D’ESSAI COMMENTE 10](#_Toc181467000)

[CONCLUSION 15](#_Toc181467001)

# INTRODUCTION

Dans ce projet, nous allons créer un jeu d'alignement en Java avec NetBeans. Ce jeu peut être joué seul ou à deux. Chaque joueur a une grille et reçoit deux pions de couleurs différentes parmi quatre : vert, rouge, jaune et bleu. L'objectif est d'aligner quatre pions de la même couleur sur une grille de 4\*4, que ce soit horizontalement, verticalement ou en diagonale. Chaque fois qu'un joueur réalise un alignement, il gagne un jeton de la couleur des pions alignés. Après avoir obtenu un jeton, la grille se réinitialise pour de nouveaux placements. La partie se termine quand un joueur a un jeton de chaque couleur.

Cette année, nous n'utilisons pas de bibliothèques, alors nous nous concentrons sur l'interface utilisateur avec du texte. Nous allons créer des menus pour encourager les joueurs à utiliser notre programme. Dans de nombreux jeux, les graphismes et les fonctionnalités sont aussi importants que l'objectif du jeu. Par exemple, un Pong simple sans scores sera moins intéressant qu'un Pong moderne avec une belle interface et des scores. Nous évitons également les pages d'erreur. La plupart des erreurs viennent d'entrées invalides. Pour éviter cela, nous avons mis en place des vérifications qui obligent l'utilisateur à fournir des réponses correctes. Cela garantit le bon fonctionnement du programme.

Pour développer le programme, nous utiliserons une matrice de 4\*4 pour chaque joueur. À chaque tour, une fonction choisit deux chiffres différents parmi les quatre couleurs. Quand le joueur fait un choix, il place son pion sur la grille en utilisant des coordonnées XY : X est une lettre de A à D et Y est un chiffre de 1 à 4. La grille s'affiche sous forme de matrice. Un coefficient de zéro signifie qu'il n'y a pas de pion sur cette case. Lorsqu'un alignement est trouvé, la grille se réinitialise et le joueur gagne un jeton. Nous suivons aussi le nombre de grilles utilisées et affichons les statistiques à la fin de la partie. La partie se termine quand un joueur a au moins un jeton de chaque couleur.

Le défi principal de ce projet est le mode solo avec l'IA. Nous pouvons créer une IA simple qui joue ses pions de façon aléatoire, mais cela serait sans intérêt. Une telle IA aurait presque aucune chance de gagner. Elle pourrait rarement réaliser des alignements.

Pour rendre l'IA plus compétitive, nous pouvons lui faire prioriser les couleurs pour lesquelles elle n'a pas encore gagné de jeton. Ensuite, avant de placer une couleur, l'IA vérifiera les possibilités d'alignement. Si un alignement est possible, elle choisira cette position. Sinon, elle placera son pion au hasard. Après placement, l'IA vérifiera si un alignement a été réalisé. Si c'est le cas, elle obtiendra le jeton. Enfin, l'IA gardera une trace des jetons gagnés et de ses placements pour influencer ses décisions futures.

# PROGRAMME INDENTE ET COMMENTE

## Main.java

|  |
| --- |
| // Bryan THIRIMANNA & Thomas MOROT-GAUDRY  // 2A - Semestre 3  // Projet 1 Java  // Octobre - Novembre 2024  package tp1;  import java.util.Random;  import java.util.Scanner;  import java.util.InputMismatchException;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Bienvenue dans le jeu d'alignement !");  boolean quitter = false;  while (!quitter) {  afficherMenuPrincipal();  int choix = saisirEntier(sc, "Erreur, veuillez saisir un choix valide.");  switch (choix) {  case 1 -> {  boolean retour = false;  while (!retour) {  afficherMenuModeDeJeu();  int mode = saisirEntier(sc, "Erreur, veuillez saisir un choix valide.");  switch (mode) {  case 1, 2 -> {  System.out.print("Entrez le pseudo du joueur 1 : ");  String pseudo1 = sc.next();  Joueur joueur1 = new Joueur(pseudo1);    Joueur joueur2 = (mode == 2) ? new Joueur("Ordinateur") : new Joueur(saisirPseudo(sc));  jouerPartie(joueur1, joueur2, mode == 2);  retour = true;  }  case 3 -> retour = true; // Retour au menu principal  default -> System.out.println("Erreur, veuillez saisir un choix valide.");  }  }  }  case 2 -> afficherCredits(); // Affichage de texte  case 3 -> afficherReglesDuJeu(); // Affichage de texte  case 4 -> {  quitter = true;  System.out.println("A bientot !"); // Ferme le jeu  }  default -> System.out.println("Erreur, veuillez saisir un choix valide.");  }  }  sc.close();  }  public static void afficherMenuPrincipal() {  System.out.println("===============TP Java Semestre 3================");  System.out.println("[1] Nouvelle Partie");  System.out.println("[2] A propos");  System.out.println("[3] Regles du jeu");  System.out.println("[4] Quitter");  System.out.println(" ");  }  public static void afficherMenuModeDeJeu() {  System.out.println("========== Choisissez le mode de jeu ==========");  System.out.println("[1] Mode 1vs1");  System.out.println("[2] Mode humain vs IA");  System.out.println("[3] Retour");  System.out.println(" ");  }  public static void afficherCredits() {  System.out.println("=============A propos=============");  System.out.println("Par deux etudiants de l'EPF : Thomas MOROT-GAUDRY et Bryan THIRIMANNA");  System.out.println("Projet Java - Semestre 3 (IDE Netbeans)");  System.out.println(" ");  }  public static void afficherReglesDuJeu() {  System.out.println("=============Regles du jeu :=============");  System.out.println("Le jeu d'alignement oppose deux joueurs, chacun recevant une grille vide de 4\*4.");  System.out.println("A chaque tour, deux pions de couleurs differentes sont tires au sort.");  System.out.println("Le but est d'aligner quatre pions de la meme couleur pour gagner un jeton de cette couleur.");  System.out.println("La grille d'un joueurs se renitialise pour chaque alignement effectues ou si la grille du joueur est remplie sans alignements");  System.out.println("Le premier a collecter un jeton de chaque couleur gagne la partie.");  System.out.println(" ");  }  public static int saisirEntier(Scanner sc, String messageErreur) {  int entier = -1;  boolean valide = false;  while (!valide) {  try {  entier = sc.nextInt();  valide = true;  } catch (InputMismatchException e) {  System.out.println(messageErreur);  sc.next();  }  }  return entier;  }  public static String saisirPseudo(Scanner sc) {  System.out.print("Entrez le pseudo du joueur 2 : ");  System.out.println(" ");  return sc.next();  }  public static void jouerPartie(Joueur joueur1, Joueur joueur2, boolean modeIA) {  System.out.println("Debut de la partie !");  System.out.println(" ");  System.out.println("Grille du joueur 1 ");  joueur1.afficherGrille();  System.out.println(" ");  System.out.println("Grille du joueur 2 ");  joueur2.afficherGrille();  boolean partieTerminee = false;  while (!partieTerminee) {  joueur1.tourJeu(joueur2);  if (joueur1.verifGagne()) {  System.out.println(joueur1.getPseudo() + " a gagne la partie !");  System.out.println(" ");  partieTerminee = true;  break;  }  if (modeIA) {  joueur2.tourJeuIA(joueur1);  } else {  joueur2.tourJeu(joueur1);  }  if (joueur2.verifGagne()) {  System.out.println(joueur2.getPseudo() + " a gagne la partie !");  System.out.println(" ");  partieTerminee = true;  }  }  }  } |

## Joueur.java

|  |
| --- |
| // Bryan THIRIMANNA & Thomas MOROT-GAUDRY  // 2A - Semestre 3  // Projet 1 Java  // Octobre - Novembre 2024  package tp1;  import java.util.Random;  import java.util.Scanner;  class Joueur {  private final String pseudo;  private final int[][] grille = new int[4][4];  private boolean jetonBleu, jetonVert, jetonJaune, jetonRouge;  private int nbreVides = 16;  private final boolean estOrdinateur;  public Joueur(String pseudo) {  this.pseudo = pseudo;  this.estOrdinateur = pseudo.equals("Ordinateur");  }  public String getPseudo() {  return pseudo;  }  public void afficherGrille() {  System.out.println(" 1 2 3 4");  for (int i = 0; i < 4; i++) {  System.out.print((char) ('A' + i) + " |");  for (int j = 0; j < 4; j++) {  String symbole = switch (grille[i][j]) {  case 1 -> "B";  case 2 -> "V";  case 3 -> "J";  case 4 -> "R";  default -> " ";  };  System.out.print(" " + symbole + " |");  }  System.out.println();  }  }  public Joueur verifTermine(Joueur adversaire) {  if (this.verifGagne()) return this;  if (adversaire.verifGagne()) return adversaire;  return null;  }  public int[] tirageCouleurs() {  Random rand = new Random();  int couleur1 = rand.nextInt(4) + 1;  int couleur2;  do {  couleur2 = rand.nextInt(4) + 1;  } while (couleur1 == couleur2);  return new int[]{couleur1, couleur2};  }  public int choixCouleur(int[] couleurs) {  if (estOrdinateur) {  return choixCouleurIA(couleurs);  } else {  Scanner sc = new Scanner(System.in);  int choix;  System.out.println("Choisissez une couleur : 0 - " + couleurEnLettres(couleurs[0]) + ", 1 - " + couleurEnLettres(couleurs[1]));  choix = Main.saisirEntier(sc, "Erreur, veuillez saisir 0 ou 1.");  return choix == 0 ? couleurs[0] : couleurs[1];  }  }  private int choixCouleurIA(int[] couleurs) {  // l'IA choisit en priorité une couleur d'un jeton qu'elle ne possède pas  if (!jetonBleu && (couleurs[0] == 1 || couleurs[1] == 1)) return 1;  if (!jetonVert && (couleurs[0] == 2 || couleurs[1] == 2)) return 2;  if (!jetonJaune && (couleurs[0] == 3 || couleurs[1] == 3)) return 3;  if (!jetonRouge && (couleurs[0] == 4 || couleurs[1] == 4)) return 4;  // Par défaut, elle choisira la 1re couleur proposée  return couleurs[0];  }  public int[] placeCouleur(int couleur) {  if (estOrdinateur) {  return placeCouleurIA(couleur);  } else {  Scanner sc = new Scanner(System.in);  int ligne = 0, colonne = 0;  while (true) {  System.out.println("Entrez une position (XY ; avec X une lettre entre A et D - Y un chiffre entre 1 et 4):");  String coordonnee = sc.next();    // Ferme le jeu si "stop" ou "quit" est tapé  if (coordonnee.equalsIgnoreCase("stop") || coordonnee.equalsIgnoreCase("quit")) {  System.out.println("La partie a ete interrompue.");  System.out.println(" ");  System.exit(0);  }  // Vérifie que l'entrée est exactement de longueur 2  if (coordonnee.length() == 2) {  char ligneChar = coordonnee.charAt(0);  char colonneChar = coordonnee.charAt(1);    // Vérifie que le premier caractère est une lettre entre 'A' et 'D' et le second est entre '1' et '4'  if (ligneChar >= 'A' && ligneChar <= 'D' && colonneChar >= '1' && colonneChar <= '4') {  ligne = ligneChar - 'A';  colonne = colonneChar - '1';    // Vérifie si la case est vide  if (grille[ligne][colonne] == 0) {  break; // Sortie de la boucle si l'entrée est valide  } else {  System.out.println("Position invalide ou case deja utilisee, veuillez reessayer.");  }  } else {  System.out.println("Erreur : Position invalide ou case deja utilisee. Entrez une lettre entre A et D suivie d'un chiffre entre 1 et 4.");  }  } else {  System.out.println("Format incorrect. Entrez une lettre entre A et D suivie d'un chiffre entre 1 et 4.");  }  }  grille[ligne][colonne] = couleur;  nbreVides--;  return new int[]{ligne, colonne};  }  }  private int[] placeCouleurIA(int couleur) {  // Tentative de trouver une case libre proche d'un pion de même couleur pour faciliter un alignement  for (int i = 0; i < 4; i++) {  for (int j = 0; j < 4; j++) {  if (grille[i][j] == 0 && caseFavorisantAlignement(i, j, couleur)) {  grille[i][j] = couleur;  nbreVides--;  return new int[]{i, j};  }  }  }  // Si aucune case stratégique n'est trouvée, elle place son pion dans une case libre de façon aléatoire  Random rand = new Random();  int ligne, colonne;  do {  ligne = rand.nextInt(4);  colonne = rand.nextInt(4);  } while (grille[ligne][colonne] != 0);  grille[ligne][colonne] = couleur;  nbreVides--;  return new int[]{ligne, colonne};  }  // On vérifie si le placement est stratégique  private boolean caseFavorisantAlignement(int ligne, int colonne, int couleur) {  int totalAligne = 0;  // l'IA vérifie les lignes pour compter les pions des mêmes couleurs  for (int j = 0; j < 4; j++) {  if (grille[ligne][j] == couleur) totalAligne++;  }  if (totalAligne >= 2) return true;  // l'IA vérfie les colonnes  totalAligne = 0;  for (int i = 0; i < 4; i++) {  if (grille[i][colonne] == couleur) totalAligne++;  }  if (totalAligne >= 2) return true;  // l'IA vérifie les diagonales  if (ligne == colonne) { // Diagonale principale  totalAligne = 0;  for (int d = 0; d < 4; d++) {  if (grille[d][d] == couleur) totalAligne++;  }  if (totalAligne >= 2) return true;  }  if (ligne + colonne == 3) { // Diagonale secondaire  totalAligne = 0;  for (int d = 0; d < 4; d++) {  if (grille[d][3 - d] == couleur) totalAligne++;  }  if (totalAligne >= 2) return true;  }  return false;  }  public boolean verifAligne(int[] coordonnees) {  int ligne = coordonnees[0], colonne = coordonnees[1], couleur = grille[ligne][colonne];  return (grille[ligne][0] == couleur && grille[ligne][1] == couleur && grille[ligne][2] == couleur && grille[ligne][3] == couleur) ||  (grille[0][colonne] == couleur && grille[1][colonne] == couleur && grille[2][colonne] == couleur && grille[3][colonne] == couleur) ||  (grille[0][0] == couleur && grille[1][1] == couleur && grille[2][2] == couleur && grille[3][3] == couleur) ||  (grille[3][0] == couleur && grille[2][1] == couleur && grille[1][2] == couleur && grille[0][3] == couleur);  }  public void videGrille() {  for (int i = 0; i < 4; i++) {  for (int j = 0; j < 4; j++) {  grille[i][j] = 0;  }  }  nbreVides = 16;  }  public void recoitJeton(int couleur) {  switch (couleur) {  case 1 -> jetonBleu = true;  case 2 -> jetonVert = true;  case 3 -> jetonJaune = true;  case 4 -> jetonRouge = true;  }  }  public void tourJeu(Joueur adversaire) {  System.out.println("C'est au tour de " + pseudo);  // Si la grille est pleine sans alignement, on la vide pour continuer le jeu  if (nbreVides == 0) {  System.out.println(pseudo + " a rempli la grille sans alignement. Reinitialisation de la grille.");  System.out.println(" ");  videGrille();  }  int[] couleurs = tirageCouleurs();  int couleur = choixCouleur(couleurs);  int[] position = placeCouleur(couleur);  if (verifAligne(position)) {  recoitJeton(couleur);  System.out.println("Felicitations, " + pseudo + " a aligne 4 pions " + couleurEnLettres(couleur) + " et gagne un jeton " + couleurEnLettres(couleur) + " !");  System.out.println(" ");  videGrille();  }  afficherGrille();  }  public void tourJeuIA(Joueur adversaire) {  System.out.println("C'est au tour de l'" + pseudo + ".");  // Si la grille est pleine sans alignement, on la vide pour continuer le jeu  if (nbreVides == 0) {  System.out.println(pseudo + " a rempli la grille sans alignement. Reinitialisation de la grille.");  System.out.println(" ");  videGrille();  }  int[] couleurs = tirageCouleurs();  int couleur = choixCouleurIA(couleurs);  int[] position = placeCouleurIA(couleur);  // Vérifie si un alignement est formé après le tour  if (verifAligne(position)) {  recoitJeton(couleur);  System.out.println(pseudo + " a aligne 4 pions " + couleurEnLettres(couleur) + " et gagne un jeton " + couleurEnLettres(couleur) + " !");  System.out.println(" ");  videGrille(); // Réinitialise la grille si un alignement est détecté  }  afficherGrille();  }  public boolean verifGagne() {  return jetonBleu && jetonVert && jetonJaune && jetonRouge;  }  private String couleurEnLettres(int couleur) {  return switch (couleur) {  case 1 -> "bleu";  case 2 -> "vert";  case 3 -> "jaune";  case 4 -> "rouge";  default -> "inconnue";  };  }  } |

# JEU D’ESSAI COMMENTE

Dans cette partie, nous allons illustrer le programme par des images que l’utilisateur reçoit dans sa console.

En lançant *Main.java*, on tombe sur le premier menu :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

La console demande alors une valeur à l’utilisateur (entre 1 et 4). Une valeur incorrecte retourne un message d’erreur : « Erreur, veuillez saisir un choix valide. » et affiche de nouveau le menu principal.

Si l’utilisateur entre une valeur égale à 2 ou 3 à la console, dans le menu principal, alors le programme renvoie des textes. Si l’utilisateur entre 4, le programme affiche « A bientôt » et ferme le jeu.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Une image contenant texte, Police, capture d’écran, algèbre

Description générée automatiquement

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Concentrons-nous sur le jeu désormais. Lorsque l’utilisateur entre « 1 », le programme affiche un nouveau menu ; « Choisissez le mode de jeu » avec 2 modes. Pour jouer au mode 1vs1, le joueur doit entrer « 1 ». Pour joueur au mode humain vs IA, le joueur doit entrer « 2 ». Si le joueur entre « 3 », alors le programme renvoie le joueur au menu principal.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

Pour simplifier l’explication du code, nous allons le mode IA du jeu.

Une image contenant texte, diagramme, capture d’écran, ligne

Description générée automatiquement

*On prendra en compte que dans ce programme, les caractères spéciaux tels que les accents ou les cédilles car ce projet NetBeans n’utilise pas l’encodage UTF-8.*

Dans le mode IA, on demande au joueur son pseudo, puis la partie commence. 2 grilles 4\*4 s’affichent ; la 1re pour le joueur, et la 2ème pour l’IA. Dans le mode 1vs1, c’est la même chose mais on demande deux pseudos à la place d’un.

Puis on entre dans la vraie boucle du jeu. A tour de rôles, on propose exactement deux couleurs aléatoires au joueur. Le joueur doit choisir une des deux couleurs en entrant « 0 » ou « 1 »[[1]](#footnote-1). Ensuite, il doit entrer des coordonnées de la forme XY (avec X une lettre entre A et D, et Y un chiffre entre 1 et 4). Si le joueur entre des coordonnées dans un format incorrect, l’erreur « Format incorrect.» s’affiche. Si le joueur entre des coordonnées où une case a déjà été utilisée, alors l’erreur « Erreur : Position invalide ou case deja utilisee. » s’affiche. Lorsque le joueur a posé son pion, la grille de ce joueur est mise à jour et est affiché sur la console. C’est ensuite au tour de l’adversaire. Dans le mode IA, on affiche seulement « C’est au tour de l’Ordinateur » ainsi que la grille après que l’IA ait joué.

Lorsqu’un joueur gagne un jeton, après que le programme a détecté un alignement (vertical, horizontal ou diagonal), un message s’affiche sur la console « Felicitations, Bryan a aligne 4 pions rouge et gagne un jeton rouge ! » et une nouvelle grille vide s’affiche.

Dans cette image ci-dessous, nous avons aligné quatre pions rouges.

Une image contenant texte, nombre, capture d’écran, document

Description générée automatiquement

On remarque que l’IA essaye d’effectuer un alignement diagonal…

Lorsque le joueur gagne la partie, un message s’affiche sur la console : « Bryan a gagne la partie » ; le programme annonce qui a gagné la partie et la grande boucle du jeu se ferme. On retourne finalement au menu.

Dans la dernière partie que nous avons joué contre l’IA, on note 4 jetons gagnés pour nous et 2 pour l’IA. Un alignement horizontal de 4 pions bleu, ainsi qu’un alignement vertical de 4 pions verts.

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

L’IA que nous avons conçue reste facile à battre. Elle essaie de gagner les jetons dans un ordre précis : bleu, vert, jaune, puis rouge. Ce choix la rend lente et inefficace. Par exemple, dans la dernière partie, nous avions gagné notre troisième jeton alors qu’elle n’en avait qu’un. Elle se concentrait uniquement sur la première couleur qu’elle voulait gagner. Elle avait pourtant assez de pions d’une autre couleur pour aligner, mais elle les a mal placés. Elle ne suit pas de stratégie alternative si elle reçoit des pions inattendus.

L'IA est également facile à battre car, parfois, elle place ses pions de manière aléatoire. Cela arrive lorsque les deux couleurs proposées ne l'intéressent pas. Cependant, notre IA reste intelligente dans sa logique. Elle cherche des alignements différents. Ce n'est pas une IA qui choisit simplement une couleur parmi les deux proposées et la place n’importe où.

Nous avons aussi testé combien de tours il lui faut pour gagner. Pour cela, nous avons toujours choisi le premier pion parmi les deux proposés. Nous avons placé nos pions sans faire d'alignements, en suivant la séquence A1-A2-A3-A4-B1-B2-B3-B4-C1-C2-C3-C4. C’était satisfaisant de voir l’IA gagner. Elle a réussi à effectuer deux alignements horizontaux, un alignement vertical et un alignement diagonal. En moyenne, l’IA a besoin d’environ une cinquantaine de tours pour gagner. Pour rappel, une partie peut se gagner en 16 tours, mais l’effet de la probabilité du choix des jetons fait qu’on a besoin de plus de tours.

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Le jeu d’alignement reste relativement long. Il aurait été utile d’ajouter une commande « save » pour quitter le jeu et le reprendre plus tard. De notre côté, nous avons ajouté une commande « quit» ou « stop», que le joueur peut utiliser au moment de saisir les coordonnées du pion à placer dans la grille. Cette commande permet de quitter la partie instantanément, avec un message s’affichant dans la console : « La partie a été interrompue ! » avant que le programme ne s’arrête.

Une image contenant texte, capture d’écran, Police, diagramme

Description générée automatiquement

# CONCLUSION

Pour conclure, ce projet a permis d'appliquer des concepts de programmation en Java tout en créant un jeu d'alignement (traduction matrices en affichage textes). Les joueurs s'affrontent sur une grille de 4\*4 pour aligner quatre pions de la même couleur. Le jeu propose diverses fonctionnalités, comme un mode IA et des options pour afficher les règles et les crédits.

Le principal défi a été de développer une IA stratégique. Elle est actuellement facile à battre, car elle priorise les couleurs non acquises et optimise ses placements. Toutefois, ses comportements aléatoires réduisent son efficacité. Pour rendre l'IA plus compétitive, elle pourrait adopter un plan B, en réalisant plusieurs alignements sans suivre un ordre précis de couleurs. Elle devrait également éviter de placer un pion de couleur non prioritaire, afin de ne pas compromettre un alignement prévu.

Aujourd’hui, la logique et les graphismes sont essentiels pour attirer les joueurs. Nous aurions pu ajouter des fonctions telles qu'un tableau de statistiques à la fin du jeu, indiquant le nombre de grilles utilisées et les jetons gagnés pour chaque couleur. Une option pour un mode en anglais aurait également été bénéfique, sans nécessiter de traduire chaque sortie du code. Pour améliorer l'expérience du joueur, l'utilisation de bibliothèques graphiques aurait vraiment fait la différence. Cela aurait permis d'enrichir l'interface du jeu et de la rendre plus attrayante. On note tout de même que les jeux actuels sont programmés en C/C++/C# (des logiciels utilisant du C et ses dérivées).

Nous avons intégré une commande "quit" pour interrompre le jeu, mais une fonctionnalité "save" améliorerait l'accessibilité[[2]](#footnote-2). Ce projet a été une excellente occasion d'approfondir nos compétences en Java et d'explorer l'intelligence artificielle dans les jeux.

Le projet est disponible sur GitHub : (<https://github.com/bryanthrmn/TP_CPO1>).

*Nous avons également utilisé la logique d'affichage de la grille du projet de l'année dernière en Python, Gomoku (puissance 5) : (*[*https://github.com/bryanthrmn/Gomoku*](https://github.com/bryanthrmn/Gomoku)*).*

1. Si le joueur entre une valeur incorrecte, alors un message d’erreur s’affiche. On force le joueur à entrer des vraies valeurs. L’avantage d’intégrer cette fonction, c’est d’éviter que le programme s’interrompe en pleine partie [↑](#footnote-ref-1)
2. Permettre au joueur de quitter la partie et de la reprendre plus tard. Il stocker un fichier qui enregistre les données de chaque tableau des joueurs, leur pseudo ainsi que les variables permettant de vérifier l’état du jeu (jetons gagnés, tour de quel joueur, etc...) [↑](#footnote-ref-2)