Projet Gomoku

Erkin Tunc Boya

Avril - Mai 2025

1 Structure du Projet

Le projet Gomoku suit une architecture orientée objet organisée en plusieurs dossiers ayant chacun une responsabilité spécifique :

- **app/** : Gère le flux général du jeu, en orchestrant l'ensemble des fonctionnalités principales telles que l'initialisation du jeu, l'interface utilisateur et l'utilisation des modules de sauvegarde et utilitaires.
- **model/** : Définit et maintient les données fondamentales du jeu, telles que la grille, les jetons, les joueurs, leurs interactions et les relations spatiales (directions).
- **ai/** : Implémente les mécanismes d'intelligence artificielle permettant à un joueur de jouer contre l'ordinateur.
- **save/** : Permet d'enregistrer et de restaurer l'état d'une partie en cours. Les classes de ce dossier sont utilisées principalement dans le flux du jeu (`app/`).
- **util/** : Fournit des outils pour améliorer l'affichage et la lisibilité du jeu dans le terminal. Ces utilitaires sont également utilisés par les classes du `app/`.

2 Fonctionnalités du Projet

Le projet Gomoku propose les fonctionnalités suivantes :

- Grille dynamique pouvant s'agrandir automatiquement.
- Vérification récursive efficace des alignements dans toutes les directions.
- Intelligence Artificielle simple pour jouer contre l'ordinateur.
- Sauvegarde et chargement des parties grâce à la sérialisation.
- Interface utilisateur en terminal avec affichage clair et en couleur.
- Paramètres dynamiques permettant de personnaliser la taille de grille, le nombre de jetons requis pour gagner et le nombre de jetons initiaux.

3 Comment Compiler et Exécuter

Vous pouvez compiler et exécuter le projet avec les commandes suivantes :

```
javac -d target/classes src/main/java/*/*.java
java -cp target/classes app.Gomoku
```

Alternativement, utilisez les scripts fournis :

- **runGomoku.bat** (Windows)
- **runGomoku.sh** (Linux/macOS)

Ces scripts compilent automatiquement tous les fichiers sources et lancent le jeu.

4 Générer la Documentation JavaDoc

Vous pouvez générer une documentation JavaDoc complète avec les scripts suivants :

- **generateDocs.bat** (Windows)
- **generateDocs.sh** (Linux/macOS)

Ces scripts créent une documentation HTML dans le dossier `/doc/`.

5 Sauvegardes des Parties

Les parties sauvegardées se trouvent dans le dossier `/data/`. Les fichiers portent l'extension `.dat` et sont générés automatiquement lors de la sauvegarde depuis le jeu.

6 Structure Visuelle du Projet

Voici l'organisation visuelle des dossiers et fichiers du projet :

```
Gomoku-Game-Projet/
|-- .vscode/                    # Parametres VSCode (optionnel)
|-- Gomoku/
|   |-- generateDocs.bat         # Windows : Generer JavaDoc
|   |-- generateDocs.sh          # Linux/Mac : Generer JavaDoc
|   |-- runGomoku.bat            # Windows : Compiler et executer Gomoku
|   |-- runGomoku.sh             # Linux/Mac : Compiler et executer
   Gomoku
|   |-- pom.xml                  # Configuration Maven (optionnel)
|   |-- data/                    # Sauvegardes des parties
|   |   +-- ErkinVsAI.dat
|   |-- src/
|   |   |-- main/
|   |   |   +-- java/
|   |   |       |-- ai/           # Logique de l'IA
|   |   |       |-- app/          # Application principale
|   |   |       |   +-- Gomoku.java
|   |   |       |-- model/        # Modeles du jeu
|   |   |       |-- save/         # Gestion sauvegarde/chargement
|   |   |       +-- util/         # Utilitaires
|   |   +-- test/                 # Reserve aux tests
|   |       +-- java/
|   +-- target/
|       |-- classes/              # Fichiers .class compiles
|       +-- test-classes/
|-- rapport/                      # Rapport LaTeX
|   +-- rapport.tex
|-- README.md                      # Documentation
+-- .gitignore                     # Regles Git
```

7 Diagramme UML

Le diagramme suivant représente uniquement la structure de la logique du jeu, c'est-à-dire les classes du dossier `model/` ainsi que la classe `GameEngine` située dans `app/`.

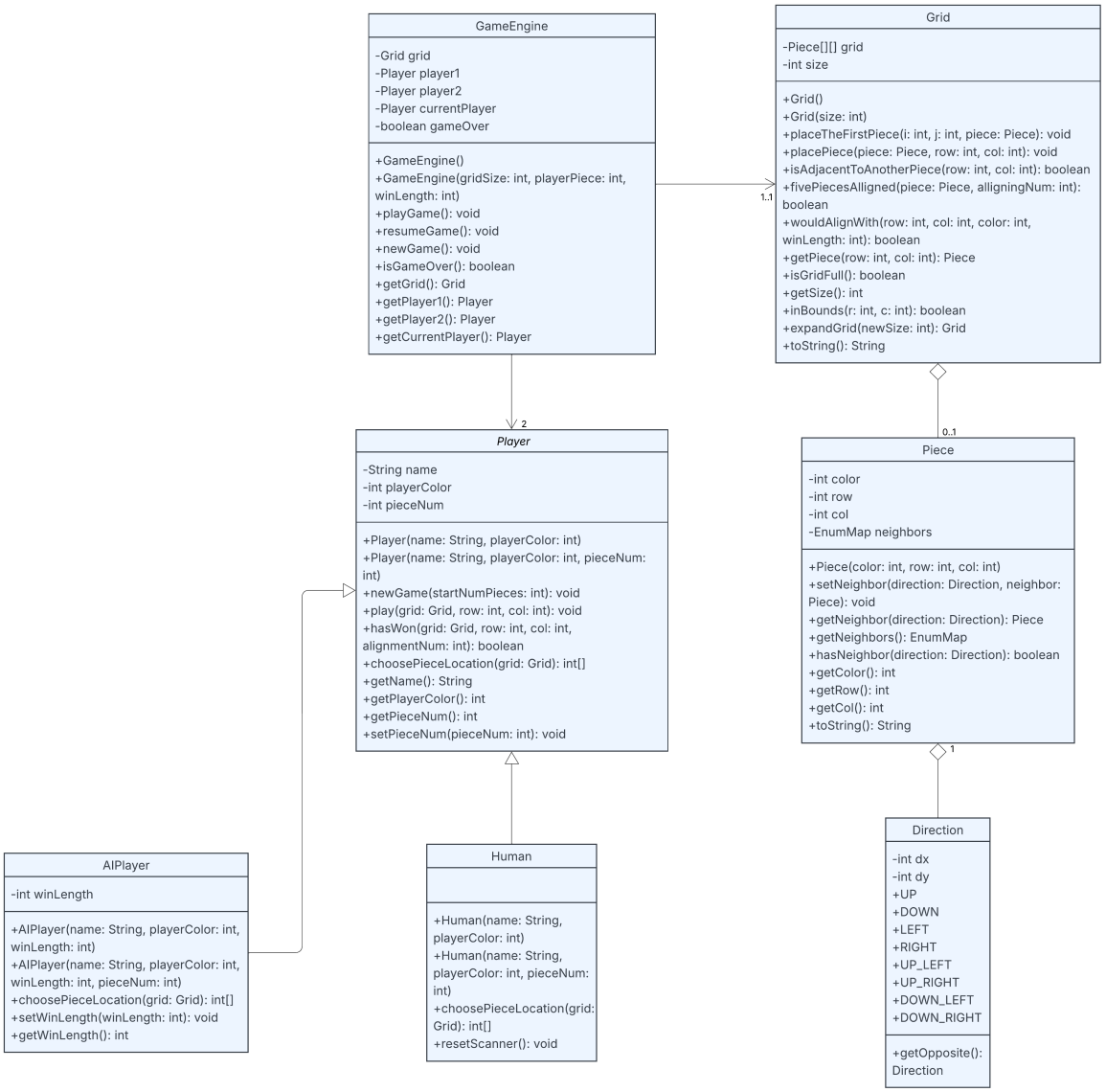

FIGURE 1 – Diagramme UML de la logique du projet Gomoku