Projet Gomoku

Erkin Tunc Boya

Avril - Mai 2025

1 Structure du Projet

Pour organiser notre travail, on a opté pour une architecture en plusieurs dossiers, chacun ayant un rôle assez précis. Voici un rapide tour d'horizon :

- **app**/ : C'est ici que tout commence. Ce dossier s'occupe de coordonner le déroulement du jeu : mise en place de la partie, affichage terminal, appels aux fonctions de sauvegarde, etc. Un peu le "chef d'orchestre" du projet.
- **model**/ : Ici, on trouve les classes qui décrivent le cœur du jeu : la grille, les jetons, les joueurs... et surtout la façon dont tout ce petit monde interagit entre eux.
- **ai**/ : Ce dossier contient tout ce qui touche à l'intelligence artificielle. Rien de très poussé, mais suffisant pour donner un peu de fil à retordre au joueur humain.
- **save**/ : Les fonctions qui permettent d'enregistrer ou de reprendre une partie en cours. On s'appuie sur la sérialisation pour garder l'état du jeu.
- **util**/ : Quelques utilitaires pratiques pour améliorer l'affichage en console, notamment pour gérer les couleurs et les styles.

On a essayé de garder une organisation simple et cohérente, pour s'y retrouver facilement même après plusieurs jours sans toucher au code.

2 Fonctionnalités du Projet

Notre version de Gomoku propose plusieurs options sympa :

- Grille dynamique qui peut s'agrandir si besoin.
- Vérification efficace des alignements (lignes, colonnes, diagonales).
- Joueur contre IA de base.
- Sauvegarde et reprise de partie.
- Interface terminal avec des couleurs pour plus de clarté.
- Possibilité de personnaliser la taille de la grille et le nombre de pions à aligner.

3 Comment Compiler et Exécuter

Pour compiler et lancer le jeu, deux solutions
Utiliser directement les commandes classiques :

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

Ou tout simplement passer par les scripts fournis :

- `runGomoku.bat` pour Windows
- `runGomoku.sh` pour Linux ou Mac

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

4 Génération de la Documentation

La JavaDoc peut être générée via :
— `generateDocs.bat` pour Windows
— `generateDocs.sh` pour Linux ou Mac
Cela créera une documentation HTML dans le dossier `/doc/`
Ces scripts créent une documentation HTML dans le dossier `/doc/`.

5 Sauvegardes des Parties

Les parties sauvegardées se trouvent dans le dossier `/data/`, sous forme de fichiers `.dat`.

6 Organisation 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

Un diagramme UML présente les liens entre les classes principales du projet, notamment celles du `model/` et la classe `GameEngine` du dossier `app/`.

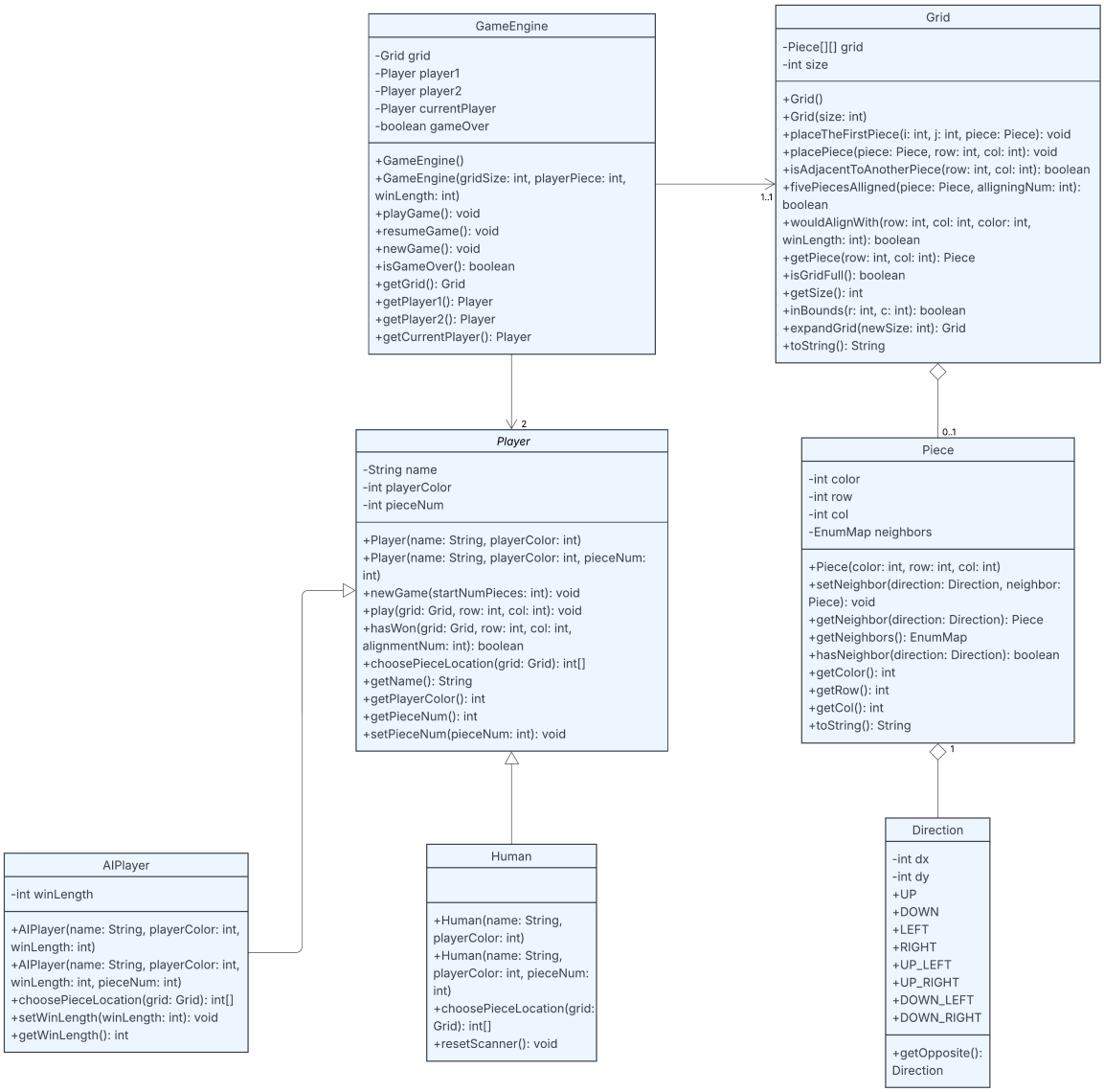

FIGURE 1 – Diagramme UML de la logique du projet Gomoku