**Rapport de Projet – JavaChess**

# Présentation

JavaChess est un jeu d’échecs développé en Java avec une interface graphique basée sur Swing. Le projet respecte une architecture MVC et implémente les règles classiques d’un jeu d’échecs (déplacements, roque, prise en passant, promotion, échec et mat, pat). Plusieurs extensions ont également été ajoutées.

Le projet a été structuré avec Maven, permettant notamment de faciliter la création d’un exécutable (*.jar*).

# Fonctionnalités de base

|  |  |  |
| --- | --- | --- |
| Fonctionnalité | Note | Temps de travail |
| Déplacement des pièces | Calcul des cases accessibles avec des Décorateurs | 10h |
| Application des coups | Application d’un coup joué | 6h |
| Calcul des fins de partie | Détection par échec et mat ou pat | 3h |
| Vue graphique Swing | Affichage du plateau et des pièces | 6h |

# Extensions

|  |  |  |
| --- | --- | --- |
| Extension | Note | Temps de travail |
| Vue console | Une vue permettant de jouer dans la console | 5h |
| Intelligence artificielle | Possibilité de jouer contre une IA avec niveau de difficulté ajustable | 2h |
| Sauvegarde / chargement PGN | Exportation & importation des parties au format PGN (historique des coups) | 7h |
| Sauvegarde / chargement FEN | Exportation & importation des parties au format FEN (position statique des pièces) | 3h |
| Fin de partie par répétition | Détection d’une répétition de position à l’aide d’une HashMap | 2h |
| Échecs 960 (Échecs aléatoires Fischer) | Variante avec position initiale aléatoire | 4h |
| Menu de sélection du mode de jeu | Interface de sélection du mode de jeu : IA, 2 joueurs, Chess960 et chargement d’une partie | 5h |

# Architecture

Le projet suit une architecture MVC (Modèle Vue Contrôleur). Les sources (fichiers java et ressources) sont placées dans le dossier ***src*** et la documentation dans le dossier ***doc***.

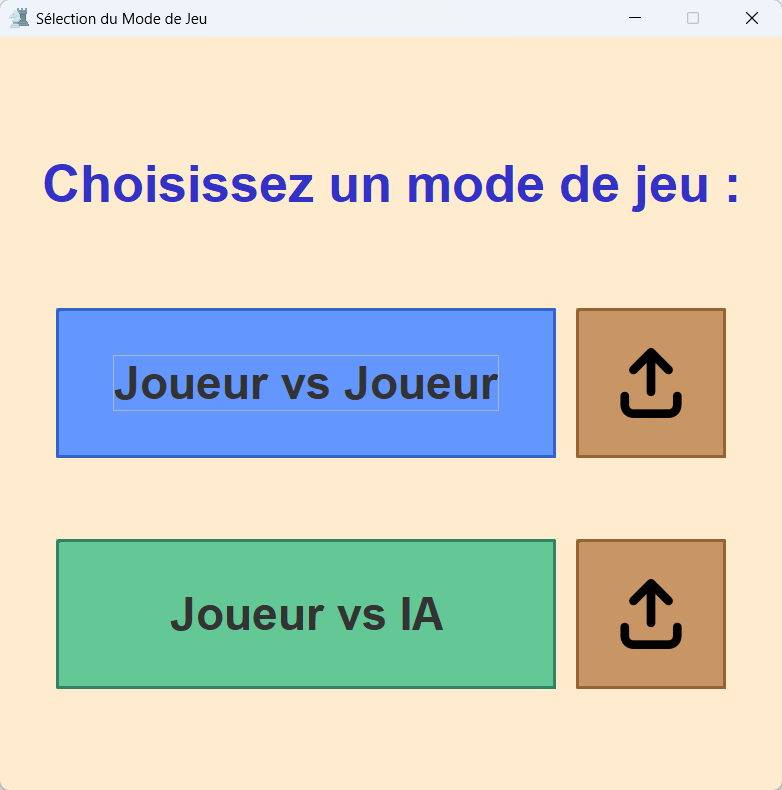
L’application a été séparé en plusieurs packages et classes :

* **modele**
  + **jeu** : Gère la logique principale du jeu.
    - *Jeu* : Classe centrale du modèle, elle contient la boucle principale du jeu.
    - *Joueur* & *JoueurIA* : Représente un joueur humain et une IA respectivement.
    - *Coup* : Modélise un coup avec une position de départ et d’arrivée.
    - *Couleur* : Enum définissant les couleurs des joueurs et des pièces (BLANC, NOIR).
    - *GameEvent* : Enum représentant les événements de jeu.
  + **pieces** : Contient les pièces d’un jeu d’échec.
    - *Piece* : Classe abstraite définissant les caractéristiques des pièces.
    - *Pion*, *Tour*, *Cavalier*, *Fou*, *Dame*, *Roi* : Sous classe représentant chaque pièce avec ses mouvements associés.
  + **mouvement** : Implémente les règles de déplacement via des décorateurs.
    - *DecoratorCasesAccessibles* : Classe abstraite définissant une interface commune pour les décorateurs.
    - *DecoratorLigne*, *DecoratorDiag*, *DecoratorCavalier*, *DecoratorPion*, *DecoratorRoi* : gèrent les mouvements en ligne, en diagonale, du cavalier, du pion et du roi.
  + **plateau** : Modélise le plateau et ses cases
    - *Plateau* : Représente le plateau et permet l’accès et la modification des cases.
    - *Case* : Représente une case.
* **vue**
  + *MenuSelection* : Choix du mode de jeu (2 joueurs, IA, Chess960, chargement de partie).
  + *VueControleur* : Interface graphique Swing (affichage du plateau, gestion des clics).
  + *MenuSelectionConsole* & *VueControleurConsole* : Version console.
  + *SoundPlayer* : Classe outil permettant de jouer un son.
* *Main* : point d’entrée de l’application

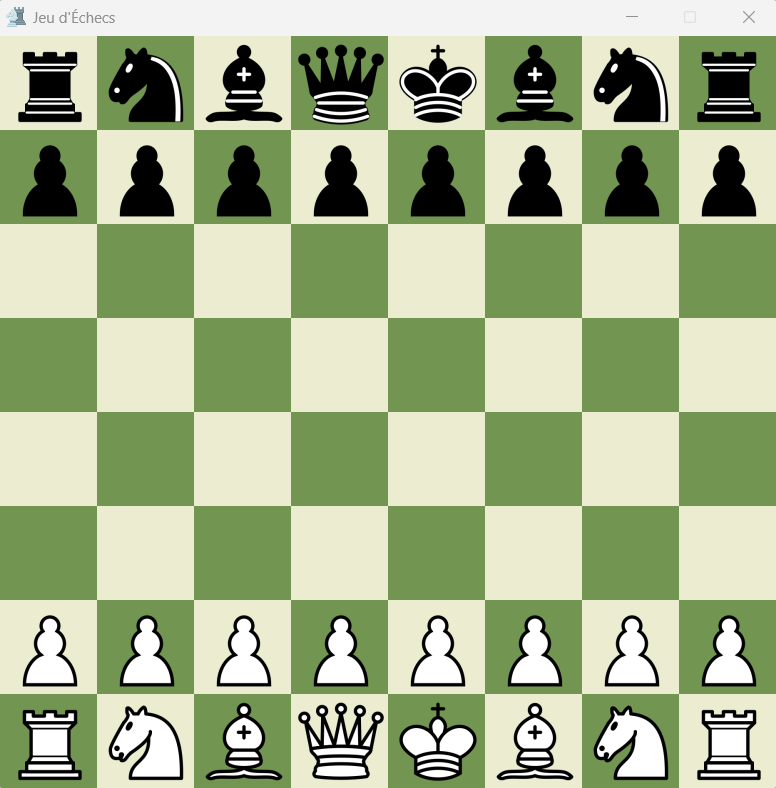
*Voir diagramme des classes en annexe ou dans le dossier* ***doc****.*

# Captures d’écran

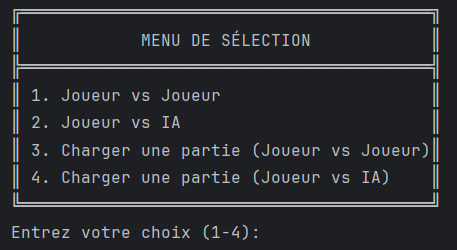
**Menu de sélection du mode de jeu :**



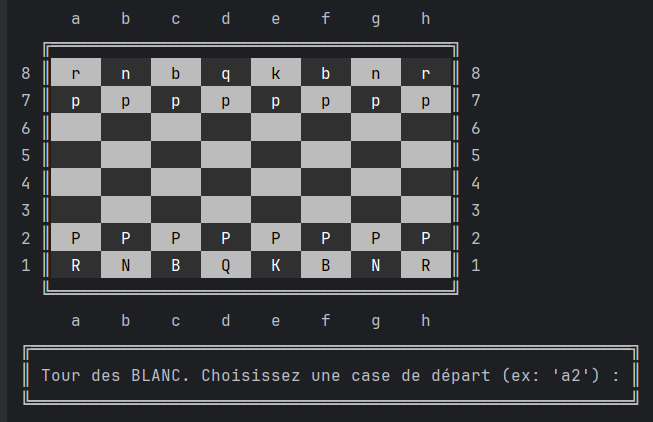
**Plateau version graphique :**



**Menu de sélection version console :**

****

**Plateau version console :**



# Annexes

