Rapport de Projet : Jeu d'Échecs en Java

Introduction

Ce rapport présente notre implémentation d'un jeu d'échecs en Java. Notre objectif était de créer une application permettant de jouer aux échecs selon les règles classiques, avec une interface graphique intuitive et la possibilité de jouer contre une intelligence artificielle (IA).

Dans ce projet, nous avons mis en œuvre plusieurs concepts de programmation orientée objet appris durant le cours de LIFAPOO, notamment l'utilisation de patterns de conception et une architecture modulaire.

Analyse du problème

Le jeu d'échecs est un jeu de stratégie pour deux joueurs se déroulant sur un plateau de 64 cases. Chaque joueur contrôle 16 pièces (blanches ou noires) avec des règles de déplacement spécifiques.

Pour implémenter ce jeu, nous avons dû relever plusieurs défis :

- Représenter les différentes pièces
- Implémenter les règles de déplacement de chaque type de pièce (Décorateur)
- Gérer les règles spéciales (roque, prise en passant, promotion des pions)
- Détecter les situations particulières (échec, échec et mat, pat, fin par répétition)
- Mise en place d'extensions : Historique, score, affichage fin de la partie , menu Principale et développement d'IA.

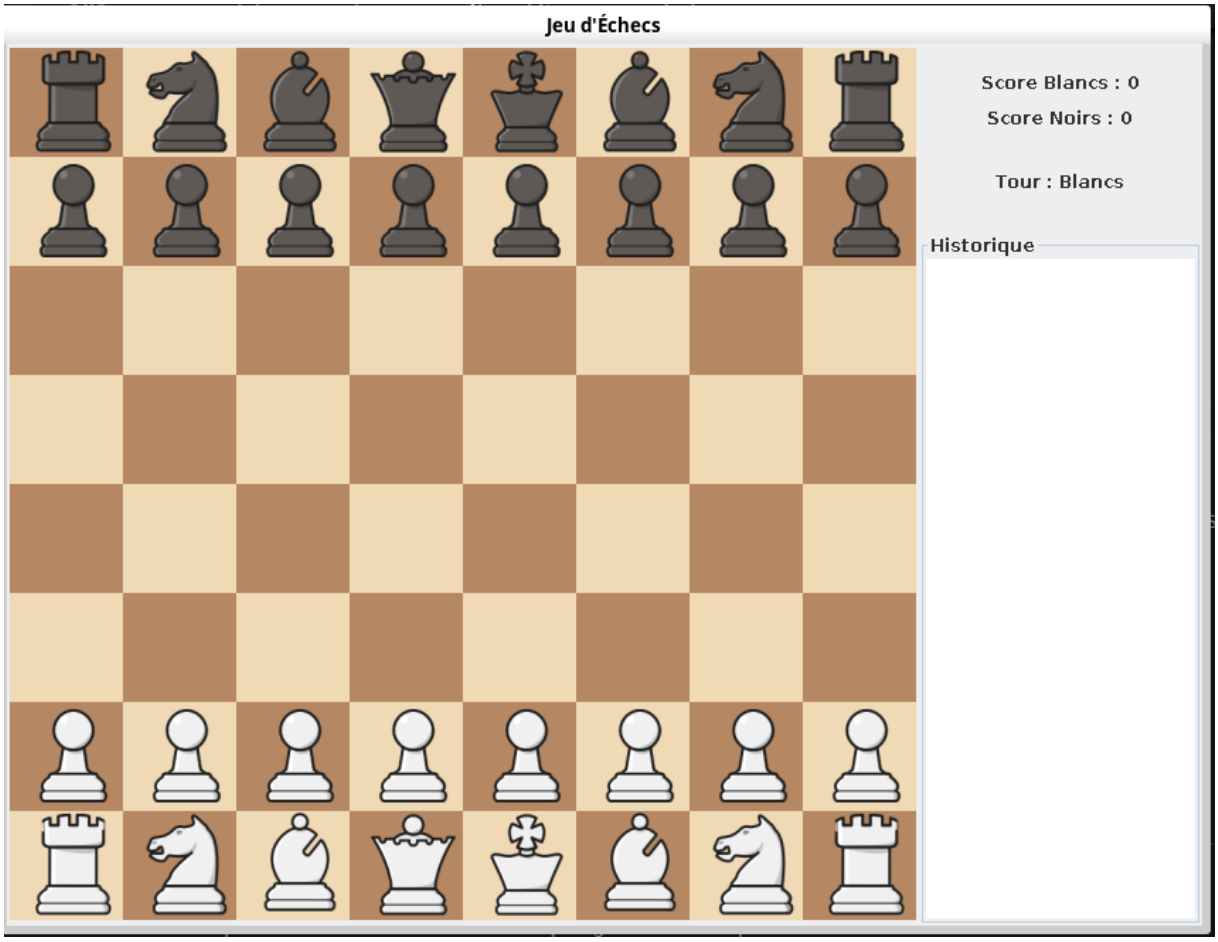
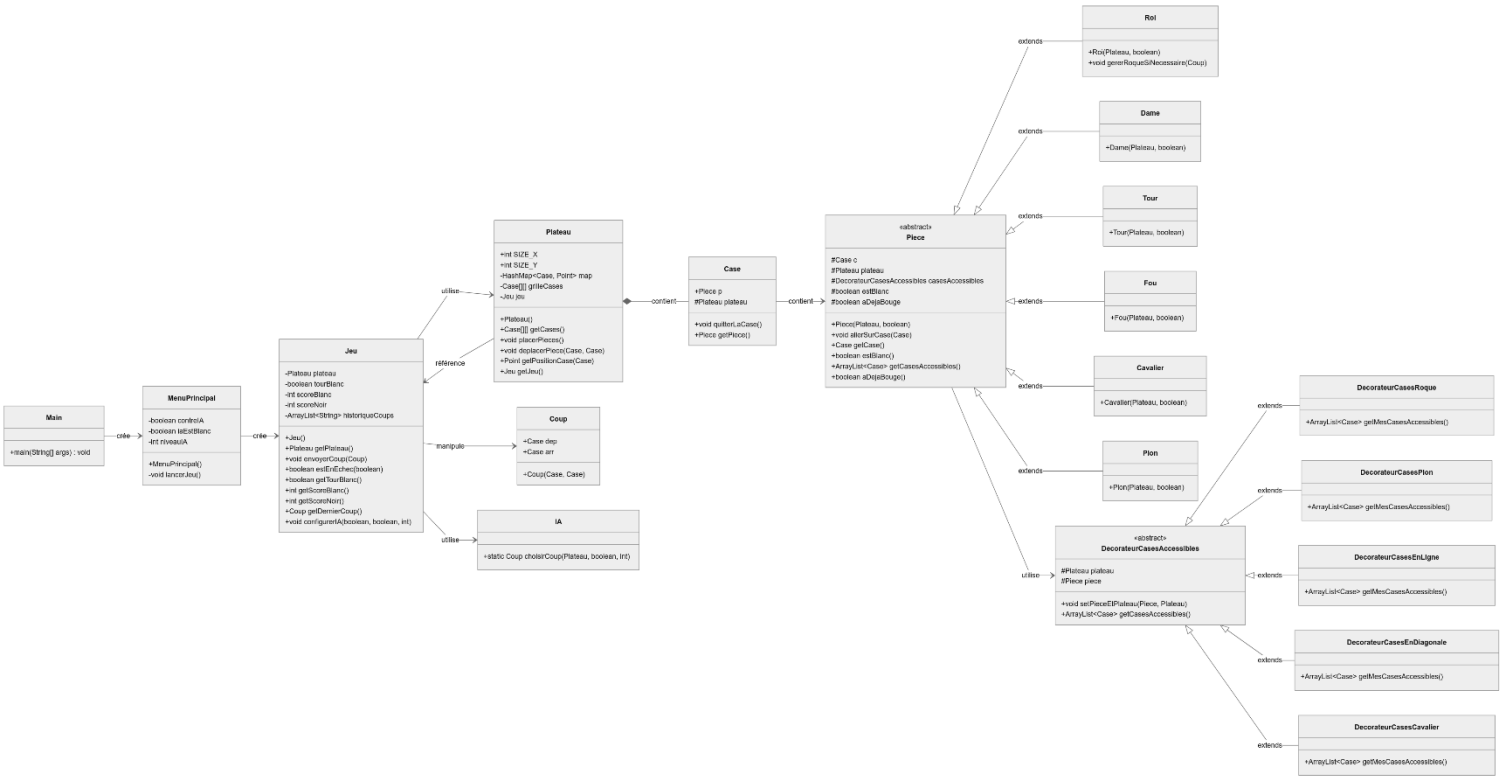


Diagramme de classes

Voici le diagramme de classes simplifié de notre application :



La structure de notre code s'articule autour des composants suivants :

- La classe Jeu qui orchestre le déroulement de la partie
- La classe Plateau qui représente l'échiquier
- Les classes Piece et ses sous-classes (Roi, Dame, Tour, etc.) qui définissent les comportements spécifiques de chaque pièce
- Le système de décorateurs (DecorateurCasesAccessibles et ses sous-classes) qui gère les règles de déplacement
- La classe IA qui implémente l'intelligence artificielle

Patterns de conception utilisés

Modèle MVC

Nous avons utilisé le pattern MVC pour séparer clairement les responsabilités :

- Le **modèle** (package modele) contient la logique du jeu
- La **vue** (classe VueControleur) gère l'affichage graphique
- Le **contrôleur** (aspects de MenuPrincipal et VueControleur) traite les interactions utilisateur

Cette séparation nous a permis de développer et tester la logique du jeu indépendamment de l'interface graphique.

Pattern Décorateur

Le pattern Décorateur a été utilisé pour implémenter les règles de déplacement des pièces. La classe abstraite DecorateurCasesAccessibles sert de base, et différentes sous-classes ajoutent des comportements spécifiques :

- DecorateurCasesEnLigne : déplacements horizontaux et verticaux (tour, dame)
- DecorateurCasesEnDiagonale : déplacements diagonaux (fou, dame)
- DecorateurCasesCavalier : déplacements en L (cavalier)
- DecorateurCasesPion : déplacements spécifiques des pions
- DecorateurCasesRoque : gestion du roque pour le roi

Ce pattern nous a permis de combiner différents types de déplacements (par exemple, la dame combine les déplacements en ligne et en diagonale) et d'ajouter facilement de nouvelles règles.

Pattern Observer

Le pattern Observer a été implémenté pour maintenir la vue synchronisée avec l'état du jeu. La classe Plateau hérite de Observable et notifie ses observateurs (dont VueControleur) lorsque l'état du jeu change.

Fonctionnalités implémentées

Règles des échecs

Notre implémentation couvre toutes les règles standard des échecs :

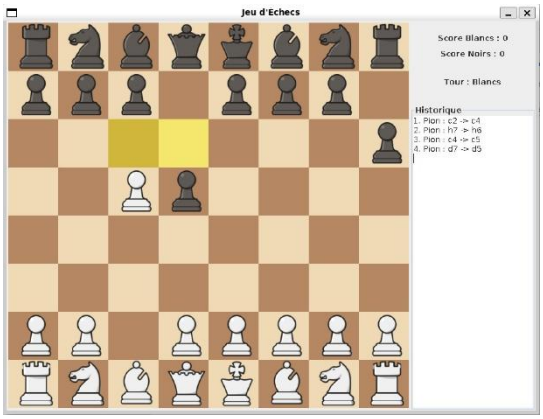
- Déplacements spécifiques à chaque type de pièce
- Captures de pièces
- Échec et échec et mat

- Pat (match nul quand un joueur ne peut plus jouer mais n'est pas en échec)

- Match nul après répétition de trois situations égales

- Roque (petit et grand)

- Prise en passant



- Promotion des pions

Intelligence artificielle

Notre IA comporte trois niveaux de difficulté :

1. **Facile** : L'IA joue des coups aléatoires parmi les coups légaux
2. **Moyen** : L'IA privilégie les captures avantageuses
3. **Difficile** : L'IA évalue les positions et cherche les coups stratégiques

L'algorithme utilisé pour l'IA de niveau difficile évalue :

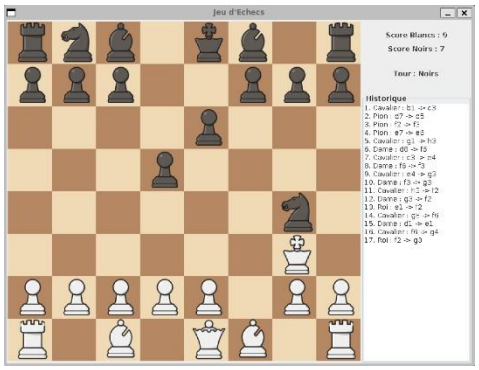
- La valeur des pièces capturées
- Le contrôle du centre de l'échiquier
- Le développement des pièces

Interface graphique

L'interface graphique comprend :

- Un menu principal permettant de choisir le mode de jeu (contre un ami ou contre l'IA)

- Un panneau d'information affichant le score et l'état du jeu
- Un historique des coups joués



- La mise en évidence des cases accessibles lors de la sélection d'une pièce

- Des dialogues pour la promotion des pions et l'annonce de fin de partie

Difficultés rencontrées et solutions

Implémentation des règles spéciales

La mise en œuvre de règles comme le roque et la prise en passant a posé des défis, car ces mouvements impliquent plus qu'un simple déplacement de pièce. Nous avons résolu ce problème en :

- Créant des décorateurs spécifiques (DecorateurCasesRoque, logique spéciale dans DecorateurCasesPion)
- Ajoutant des méthodes spéciales comme gererRoqueSiNecessaire dans la classe Roi
- Conservant l'historique des coups pour détecter les situations de prise en passant

Détection des situations d'échec et mat

La vérification de l'échec et mat a nécessité de simuler tous les coups possibles pour un joueur et de vérifier s'ils permettent d'échapper à l'échec. Cette opération étant coûteuse, nous avons :

- Implémenté des méthodes d'annulation de coup pour les tests
- Créé des versions "sans notification" des méthodes critiques pour éviter des mises à jour inutiles de l'interface

Conception de l'IA

La conception d'une IA représentait un défi important. Nous avons adopté une approche par niveaux :

- Niveau facile : coups aléatoires (simple à implémenter)
- Niveau moyen : captures avantageuses (utilisation d'une fonction d'évaluation simple)
- Niveau difficile : stratégie plus avancée (évaluation des positions et des menaces)

Conclusion

Ce projet nous a permis d'appliquer plusieurs concepts importants de la programmation orientée objet, comme l'héritage, le polymorphisme et l'utilisation de patterns de conception. L'architecture modulaire que nous avons conçue a facilité le développement incrémental et le test des différentes fonctionnalités.