Rapport - Échecs

Auteurs: Bleuer Rémy, Changanaqui Yoann & Duruz Florian

1. Introduction

Le but de ce laboratoire est d'implémenter un jeu d'échecs fonctionnel. Ce rapport résume les choix d'implémentation effectués pour la réalisation de ce projet.

2. Architecture générale

- ChessGame (contrôleur) :
  - Met en place la logique centrale du jeu.
  - Stocke le plateau (`Board`), le joueur courant (`currentPlayerColor`), le dernier coup joué (`lastMove`) et l'état de la partie (`isGameOver`).
  - Gère la méthode `move()` et relègue les mouvements spécifiques aux pièces concernées.
- Board :
  - Contient un tableau de `Piece` de taille 8x8.
  - Initialise le plateau avec les pièces à leur position initiale.
  - Méthodes pour manipuler les pièces (`movePiece()` et `removePiece()`).
- Position :
  - Gère les coordonnées d'une case sur le plateau `(x, y)`
  - Vérifie si une position se trouve dans les limites du plateau.
- Move :
  - Encapsule deux positions : `from` et `to`.
- Piece (abstract) :
  - Représente une pièce d'échec avec une couleur (`PlayerColor`) et une position (`Position`).
  - Méthodes communes à toutes les pièces : `isValidMove()`, `executeMove()`.
  - Sous-classes : `Pawn`, `Rook`, `Knight`, `Bishop`, `Queen`, `King`.
- FirstMovePiece (abstract) :
  - Super-Classe : `Piece`.
  - Représente les pièce ayant un movement spécial nécessitant de ne pas avoir bouger.
  - Sous-classes : `Pawn`, `Rook`, `King`.

3. Gestions des mouvements

3.1 Méthodes isValidMove()

- Polymorphisme : chaque pièce implémente sa propre logique de déplacement.
- Pièces linéaires (fou, tour, dame) : vérifient la trajectoire (pas d'obstacles).
- Roi : autorise un déplacement d'une case et la logique de roque.
- Pion : gère l'avance d'une ou deux cases, la capture diagonale, la prise en passant et la promotion.

3.2 Méthode executeMove(...)

- Chaque pièce dispose d'une méthode `executeMove(Move, Board, ChessView, Move lastMove)` qui réalise le déplacement.
- Roi : gère le roque.
- Pion : gère la prise en passant et la promotion.

3.2.1 Roque

- On vérifie si le roi veut se déplacer de deux case à droite ou à gauche de sa position (sans qu'il y ait de pièce entre le roi et la tour)
- Lorsque les deux pièces concernées n'ont jusqu'alors pas bougé de la partie elles sont autorisées à exécuter le roque

3.2.2 Promotion

- Teste si le pion atteint la dernière ligne.
- Utilise `view.askUser()` pour demander la pièce choisie au joueur.
- Remplace le pion par une nouvelle pièce dans le `Board` et met à jour la vue.

3.2.3 Prise en passant

- Flag `doublePawnMove` pour signaler qu'un pion a avancé de 2 cases.
- `isEnPassant` est mis à true si le déplacement est validé en diagonale vers une case vide et que le dernier coup était un double pas d'un pion adverse qui se place à côté du pion allié.
- Après le déplacement, on retire le pion adverse juste derrière la case d'arrivée.

3.3 Echec et mat

- On Vérifie si le roi est échec.
- On regarde si le roi a un coup valide (roque n'est pas permis si le roi est en échec selon les règles officielles).
- On Vérifie si une pièce alliée peut tuer l'attaquant, si on ne peut pas et que l'attaquant est un cavalier alors c'est checkmate
- On Vérifie selon la direction(horizontal ou vertical) de l'attaque si une pièce peut se mettre entre le roi et l'attaquant.
- échec et mat si aucune des vérfications ci-dessus ne passe.

3.4 Pat

- Vérifie que le roi n'est ps en échec.
- On parcourt toutes les pièces pour vérifier qu'aucune à un coup légal en testant toutes les destinatiosn possibles.
- Si une destination est un coup valide (`isValidMove`) et qu'après avoir simulé le déplacement le roi n'est pas en échec, ce n'est pas un pat.
- Il y a pat si aucun déplacement permet d'éviter ou d'entrer en échec.

4. Tests

Nous avons réalisé chaque cas de figure spécifique demandé dans l'énoncé, voici une liste des tests effectués (seulement sur les pièces blancs, mais les pièces noires fonctionnent de la même manière) :

Scénario de test	Résultat attendu
Pions	
Avancer un pion blanc d'une case en avant lorsque la case est libre (pas encore déplacé)	OK
Avancer un pion blanc de deux cases en avant sur son premier mouvement (les deux cases libres)	OK
Avancer le pion blanc en diagonale d'une case avec une pièce noire sur la destination (capture)	OK
Avancer le pion blanc en diagonale d'une case sans pièce noire sur la destination (ni en passant)	KO
Prendre en passant un pion noir qui vient de faire un double pas et s'est arrêté à côté de nous	OK
Prise en passant alors que le pion adverse a bougé deux tours avant (plus éligible)	KO
Promotion d'un pion blanc qui atteint la 8 ème rangée (dame, tour, fou, cavalier)	OK
Roque	
Roquer (petit roque,chemin vide, pas en échec)	OK
Roquer (grand roque, chemin vide, pas en échec)	OK
Roquer alors que la tour ou le roi ont déjà bougé	KO
Roquer alors que le chemin est bloqué par une pièce	KO
Roquer alors que le roi est en échec	KO
Roi	
Avancer le roi blanc de deux cases en avant sur un emplacement vide et sans obstacle	KO
Avancer le roi blanc d'une case en avant sur un emplacement vide	OK
Avancer le roi blanc d'une case en diagonale sur un emplacement vide	OK
Avancer le roi sur une case occupée par une pièce noire (capture)	OK
Fou	
Déplacer un fou en diagonale tant que le chemin est libre	OK
Déplacer un fou en diagonale si une pièce (alliée ou ennemie) se trouve sur la trajectoire	KO
Déplacer un fou sur une case avec un adversaire (capture)	OK

Scénario de test	Résultat attendu
Cavalier	
Déplacer un cavalier en « L » sur une case vide (2 cases dans une direction et 1 dans l'autre)	OK
Déplacer un cavalier en « L » sur une case vide (1 case dans une direction et 2 dans l'autre)	OK
Déplacer un cavalier sur une case avec une pièce noire (capture)	OK
Tour	
Déplacer la tour blanche en ligne droite (horizontalement ou verticalement) sans pièce sur le chemin	OK
Déplacer la tour blanche si une pièce (alliée ou ennemie) bloque la trajectoire	KO
Tenter de bouger la tour blanche en diagonale	KO
Déplacer la tour blanche sur une case avec une pièce noire (capture)	OK
Dame	
Déplacer la dame blanche en ligne droite (chemin libre)	OK
Déplacer la dame en diagonale (chemin libre)	OK
Déplacer la dame avec une pièce (alliée ou ennemie) sur sa trajectoire	KO
Déplacer la dame sur une case avec une pièce noire (caputre)	OK
Échec	
Mettre le roi adverse en échec (déplacer une pièce pour qu'elle puisse capturer le roi s'il ne bouge pas)	"Check!"
Déplacer une pièce qui laisserait son propre roi en échec	KO
Déplacer un pion qui découvre un échec	KO
Échec et mat	
Roi adverse en échec et aucun déplacement possible du roi qui évite l'échec	"Checkmate!"
Aucune pièce alliée ne peut capturer la pièce qui donne l'échec ou bloquer le chemin	"Checkmate!"
Déplacer le roi en échec alors qu'il n'a aucune case valide	KO
Pat	
Roi non en échec, mais aucune pièce alliée ne possède de coup légal	"Stalemate!"

Pour les tests des pats, nous avons utilisé les configurations suivantes :

- https://www.chess.com/forum/view/game-showcase/fastest-stalemate-known-in-chess
- https://www.chess.com/forum/view/general/stalemate-on-move-12-with-all-the-pieces-on-the-board

5. Conclusion

L'implémentation utilise une architecture orientée objet, où :

1. `ChessGame` gère le jeu et appelle les méthodes nécessaires,
2. `Board` pour déplacer/manipuler les pièces, et
3. Chaque `Piece` gère son déplacement et son exécution spécifique.

Difficultés rencontrées

- L'encapsulation de chaque mouvement spécifique dans les classes des pièces a été un défi pour garantir la modularité et la lisibilité du code.
- L'echec et mat a été un défi pour nous car il a fallu vérifier si le roi était en echec et si il pouvait bouger pour sortir de cette situation d'après toutes les pièces adverses.
- La prise en passant a été un défi pour nous car il a fallu vérifier si le pion adverse avait bougé de deux cases et si le pion allié pouvait le prendre.