# **Rapport labo 8**

1. La classe abstraite Move

Afin de factoriser au mieux le code commun des pièces, il a été décidé de créer une classe abstraite Move possédant une fonction abstraite, isValid(). Ainsi, 3 sous-classes ont été créées :

* Diagonal
* Horizontal
* Vertical

Nous avons fait le choix de séparer le mouvement horizontal du mouvement vertical car la classe Pawn n’utilise que le mouvement vertical, ceci afin d’éviter un comportement ambigu.

Graphical user interface, text, application

Description automatically generated

Figure 1 : sous-classe Diagonal

Text

Description automatically generated

Figure 2 : sous-classe Horizontal

Text

Description automatically generated

Figure 3 : sous-classe Horizontal

2. La classe abstraite Piece

Pour définir les pièces du jeu, une première classe abstraite Piece a été créée. Celle-ci est implémentée par les pièces suivantes :

* Le fou
* Le cavalier
* La reine

La raison pour laquelle ces pièces implémentent la classe abstraite Piece est due au fait que les autres pièces héritent d’une autre classe abstraite PieceMove, dont nous discuterons plus tard.

Il est important de noter que le seul attribut de la classe est la couleur de la pièce qui possède un getter. Les autres méthodes de cette classe sont toutes abstraites :

* getPieceType()
* move()
* isMoveValid()

2.1 La méthode getPieceType()

getPieceType() est une fonction qui permet de retourner le type de la pièce.

2.2 La méthode move()

move() est une fonction qui permet d’effectuer le mouvement de la pièce sur le Board.

Celle-ci prend en paramètre le Board, ce qui permet de le mettre à jour mais également la vue.

Cette méthode ne peut être appelée que si le mouvement a été validé.

2.3 La méthode isMoveValid()

isMoveValid() est une fonction qui permet de vérifier si une pièce peut se déplacer sur une case souhaitée. Celle-ci prend en paramètre le Board, ce qui permet de l’interroger sur la position des autres pièces.

Cette fonction prend également en paramètre un booléen checkLegality, qui permet de choisir si l’on désire ou non, vérifier que le mouvement que l’on souhaite réaliser, ne nous met pas en échec.

Graphical user interface, text, application

Description automatically generated

Figure 4 : Exemple pour getPieceType(), move() et isMoveValid()

3. La classe abstraite PieceMove

Pour définir les autres pièces du jeu, une seconde classe abstraite PieceMove a été créée. Cette dernière est implémentée par les pièces restantes :

* Le roi
* Le pion
* La tour

PieceMove hérite de la classe abstraite Piece et possède en plus, un booléen hasMoved qui permet de savoir si la pièce a déjà bougé. De plus, elle possède un getter et un setter pour cette variable. L’implémentation de cette classe abstraite permet de valider l’exécution des mouvements suivants :

* Le roque pour le roi et la tour
* Le déplacement de 2 cases pour le pion

Créer une classe abstraite permet de factoriser le code commun à ces trois classes qui l’implémentent.

4. La classe Board

Cette classe permet de mémoriser l’état du jeu et fournit des méthodes utilitaires que les pièces vont pouvoir utiliser. Par exemple, mettre à jour les pièces dans la matrice et les déplacer sur la vue. Les principales fonctions du Board utilisées par les pièces sont :

* isMoveAuthorized()
  + Vérifie qu’un joueur ne s’attaque pas lui-même
* isMoveFree()
  + Vérifie qu’un mouvement est libre
* isMoveLegal()
  + Vérifie qu’un mouvement est légal, c’est-à-dire qu’un joueur ne se met pas en échec

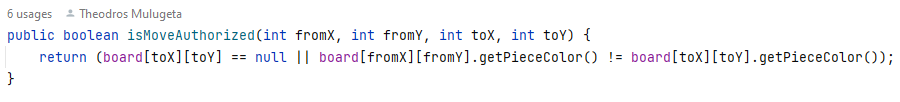


Figure 5 : Exemple pour isMoveAuthorized()

Graphical user interface, text, application, email

Description automatically generated

Figure 6 : Exemple pour isMoveFree()

Graphical user interface, text, application, email

Description automatically generated

Figure 7 : Exemple pour isMoveLegal()

Le Board est également utilisé pour mémoriser le dernier pion ayant avancé de 2 cases et le tour durant lequel, ce mouvement a été effectué. Ainsi, lorsqu’un pion souhaitera effectuer une prise en passant, il pourra questionner le Board.

Graphical user interface, text, application, email

Description automatically generated

Figure 8 : getters et setters qui servent à la mise en place de la prise en passant dans la classe Board

Text

Description automatically generated

Figure 9 : Fonction qui détermine si la prise en passant a lieu dans la classe Pawn

Le Board permet aussi à la classe Pawn d’effectuer la promotion. Pour ce faire, nous avons implémenté un record interne à la classe Board qui implémente l’interface **ChessView.UserChoice** et dont l’attribut est un objet de type Piece, ce qui permet son retour, une fois la promotion choisie.

Graphical user interface, text, application, email

Description automatically generated

Figure 10 : record dans la classe Board

Text

Description automatically generated

Figure 11 : fonction autorisant la promotion dans la classe Board

Une autre classe interne, KingPosition a également été implémentée dans la classe Board pour mémoriser la position des 2 rois.

Graphical user interface, text, application

Description automatically generated

Figure 12 : classe interne KingPosition dans la classe Board