POO1 - Jeu d'échecs

Alexandre Duc

— Durée du laboratoire : 12 périodes.

— Rendez vos fichiers sur Cyberlearn.

— Imprimez votre rapport ainsi que votre code source.

— Nous allons potentiellement annoncer des modifications du code fourni et de l'énoncé sur cyberlearn ou en classe. Restez informés !

1 Tâche

Le but de ce laboratoire est d'implémenter un jeu d'échecs fonctionnel. Une interface graphique ainsi qu'un mode console vous sont fournis. Les règles à implémenter sont les suivantes : [1]

— Les mouvements et les prises de toutes les pièces (pions, tours, cavaliers, fous, dames, rois).

— Le **petit et le grand roque** doivent être fonctionnels. Leur mouvement est initié en bougeant le roi de deux cases vers la droite ou vers la gauche. Ce coup ne peut être effectué si le roi est en échec, s'il a déjà bougé, si la tour concernée a déjà bougée ou si une des cases sur lesquelles le roi passe est en danger.

— La **prise en passant** doit être fonctionnelle. Ce coup s'effectue en prenant un pion ayant avancé de deux cases au tour précédent comme s'il n'avait avancé que d'une case.

— La **promotion de pions** doit être implémentée. Les types de promotions possibles sont tour, cavalier, fou, dame.

— Si un roi est en échecs, vous pouvez envoyer un message à la vue à l'aide de la méthode `displayMessage(String)` disant *Check !*.

— Il n'est **pas** nécessaire de détecter l'échec et mat ni les différents match nuls.

2 Points bonus

Toute implémentation supplémentaire conséquente donnera des points bonus. Par exemple, l'implémentation de l'échec et mat ou des matchs nuls par pat ou impossibilité de mater.

1. Des détails sont donnés sur Wikipedia : https://en.wikipedia.org/wiki/Rules_of_chess.

3 Implémentation

Nous vous fournissons les classes et interfaces suivantes (qui ne devraient pas être modifiées) :

- `PieceType` est une enum listant les différents types de pièces.
- `PlayerColor` est une enum listant les couleurs des joueurs (blanc, noir)
- `ChessView` est une interface permettant de représenter une **vue**. Nous vous fournissons l'implémentation de deux vues : la vue GUI `GUIView` et la vue Console `ConsoleView`. Tout le code de ces vues se trouve dans les packages views et assets.
- `ChessController` est une interface permettant de contrôler le jeu d'échecs depuis la vue. Il s'agit de **l'interface que vous devez implémenter** dans le contrôleur.

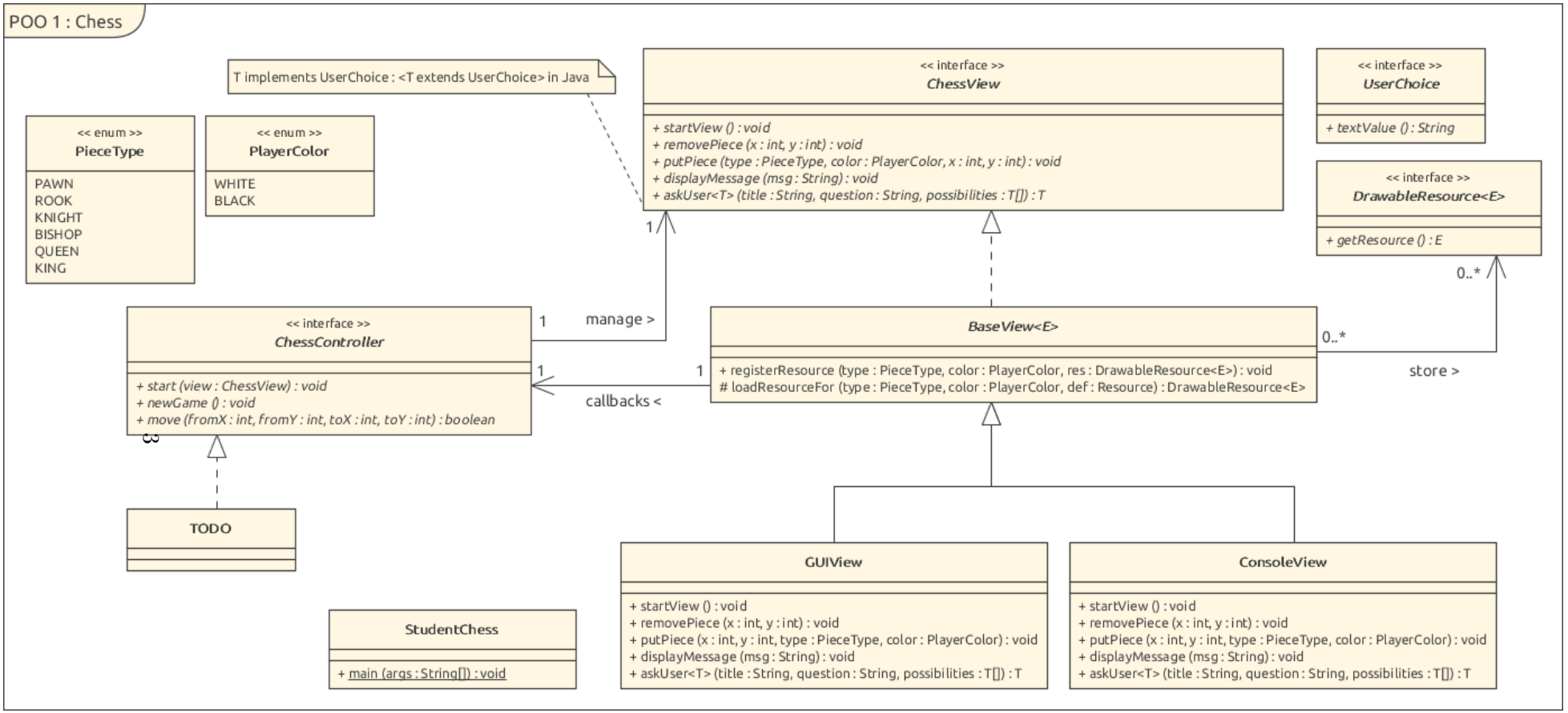
Nous vous conseillons de coder toutes vos classes dans un package `engine`.
Pour utiliser le code fourni, votre main doit ressembler à ça :

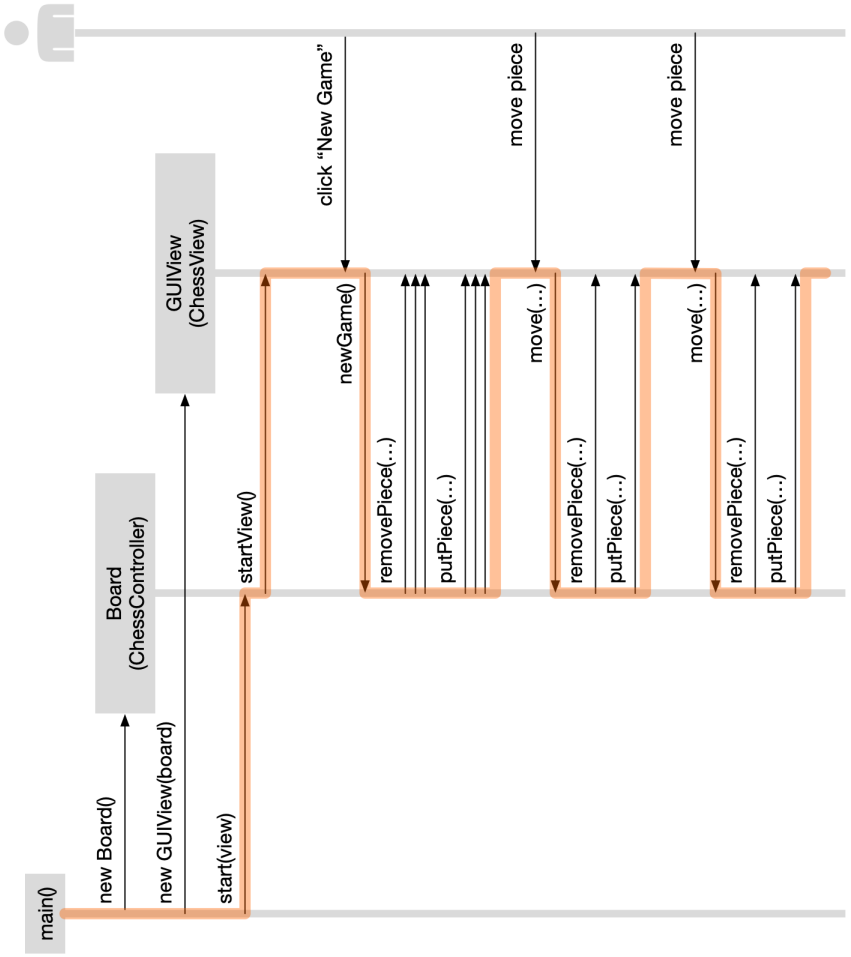
```
public static void main(String[] args) {
    // 1. Création du contrôleur pour gérer le jeu d'échec
    ChessController controller = new ... // Ici, vous devez instancier un
    ChessController

    // 2. Création de la vue désirée
    ChessView view = new GUIView(controller); // mode GUI
    //ChessView view = new ConsoleView(controller); // ou mode Console

    // 3. Lancement du programme.
    controller.start(view);
}
```

Attention à l'encapsulation et l'aspect OO de votre modélisation.


POO 1 : Chess
T implements UserChoice : <T extends UserChoice> in Java
<< interface >>
ChessView
+ startView () : void
+ removePiece (x : int, y : int) : void
+ putPiece (type : PieceType, color : PlayerColor, x : int, y : int) : void
+ displayMessage (msg : String) : void
+ askUser<T> (title : String, question : String, possibilities : T[]) : T
<< interface >>
UserChoice
+ textValue () : String
<< enum >>
PieceType
PAWN
ROOK
KNIGHT
BISHOP
QUEEN
KING
<< enum >>
PlayerColor
WHITE
BLACK
<< interface >>
DrawableResource<E>
+ getResource () : E
0..*
1
manage >
<< interface >>
ChessController
+ start (view : ChessView) : void
+ newGame () : void
+ move (fromX : int, fromY : int, toX : int, toY : int) : boolean
1
callbacks <
1
BaseView<E>
+ registerResource (type : PieceType, color : PlayerColor, res : DrawableResource<E>) : void
# loadResourceFor (type : PieceType, color : PlayerColor, def : Resource) : DrawableResource<E>
0..*
store >
TODO
GUIView
+ startView () : void
+ removePiece (x : int, y : int) : void
+ putPiece (x : int, y : int, type : PieceType, color : PlayerColor) : void
+ displayMessage (msg : String) : void
+ askUser<T> (title : String, question : String, possibilities : T[]) : T
ConsoleView
+ startView () : void
+ removePiece (x : int, y : int) : void
+ putPiece (x : int, y : int, type : PieceType, color : PlayerColor) : void
+ displayMessage (msg : String) : void
+ askUser<T> (title : String, question : String, possibilities : T[]) : T
StudentChess
+ main (args : String[]) : void


main()
new Board()
Board
(ChessController)
new GUIView(board)
GUIView
(ChessView)
start(view)
startView()
click “New Game”
newGame()
removePiece(...)
putPiece(...)
move piece
move(...)
removePiece(...)
putPiece(...)
move piece
move(...)
removePiece(...)
putPiece(...)