HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Enzo Allemano
Malo Romano

Graf Marcel
Franchetti Thibaud
07.01.2021

Programmation Orientée Objet

Rapport Laboratoire 8 :
Echecs

Description des classes

L'arborescence des classes que nous avons implémentée se présente de la façon suivante. Comme demandé dans la donnée du laboratoire nous avons créé un package `engine` dans lequel se trouve toutes les classes nécessaires au contrôleur.

Le package `engine` contient la classe `Case`, `Plateau` et `Main`. Il contient aussi un autre package, `pieces`, qui nous sert à mieux séparer les différentes classes utilisées.

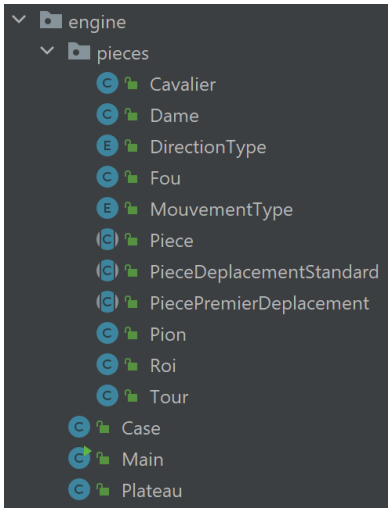


Figure 1 - Arborescences des classes de pièces

Description de l'arborescence d'une pièce

Pour simplifier et factoriser le plus possible la gestion des pièces, nous avons décidé d'implémenter une arborescence comme ci-dessous :

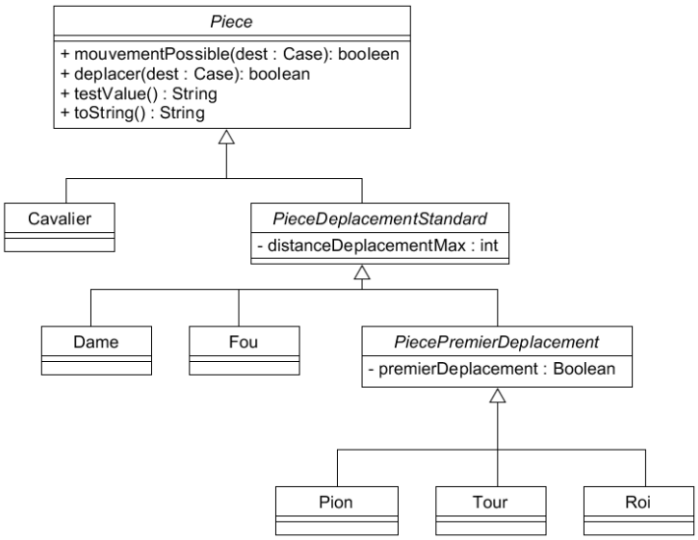


Figure 2 - Arborescence des classes des pièces dans l'UML

De ce fait, la méthode `mouvementPossible` peut être implémentée de manière très spécifique à chaque pièce. Chaque pièce implémentant ses propres règles en plus des règles de ses classes parentes.

Enum « MouvementType »

Cet enum implémente tous les types de mouvements qu'une pièce pourra faire durant une partie. Il implémente également les type de mouvements spéciaux réservés à certaines pièces.

```
public enum MouvementType {
    CLASSIQUE,
    NON_VALIDE,
    GRAND_ROQUE,
    PETIT_ROQUE,
    PROMOTION,
    DOUBLE,
    EN_PASSANT
}
```

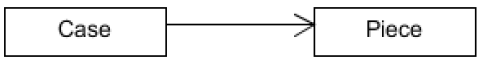
Figure 3 - Liste des types de mouvements

La fonction `mouvementPossible` retourne un `MouvementType`. Cela facilite la lisibilité du code dans la classe plateau et permet d'aiguiller la suite du programme en fonction du mouvement retourné par la fonction.

Case et Plateau

La classe `Case` possède un position x et y ainsi qu'une pièce et un pion fantôme. Nous pouvons donc peupler l'échiquier de 64 cases. Cette classe possède notamment diverses fonctions pour récupérer et définir les informations sur ladite case.

En ce qui concerne la navigabilité `Case` <-> `Piece`, nous avons choisi de l'implémenter de cette manière :



La raison est qu'une pièce existe uniquement dans une case, et fournit les méthodes permettant de savoir si un déplacement est possible. L'échiquier est manipulé par les cases et non pas par les pièces.

Implémentation

Les mouvements simples

Chaque pièce héritant de la classe `PieceDeplacementStandard` possède une liste de mouvements parmi l'enum `DirectionType`. Par exemple :

- Le roi : `TOUS`
- La dame : `TOUS`
- Le pion : `TOUS`
- La tour : `DROIT`
- Le fou : `DIAGONALE`

Chaque pièce de la classe `PieceDeplacementStandard` a également une valeur de déplacement maximum, stockée dans l'attribut `distanceDeplacementMax`.

On peut donc facilement encadrer les déplacements d'une pièce pour la restreindre aux déplacements autorisés.

Pour vérifier si un déplacement est valide, nous utilisons principalement les méthodes suivantes.

- `trajectoireLibre`
- `mouvementPossible`

Fonction `trajectoireLibre`

C'est la première vérification qu'effectue le plateau lorsqu'un mouvement est demandé. Elle répond à la question « est-ce que le chemin entre la case de source et de destination est libre ? ». Cette vérification n'est pas effectuée si la pièce est un cavalier, qui peut par définition passer au-dessus des autres pièces.

La vérification s'effectue dans tous les cas, même si le mouvement est impossible pour la pièce, par exemple un pion qui veut se déplacer de 5 cases en diagonale. La principale raison est que la vérification de trajectoire est moins gourmande que la vérification du mouvement du pion. Il est donc préférable d'effectuer celle-ci dans tous les cas.

Fonction `mouvementPossible`

La classe Piece et toutes les classes, abstraites ou non, qui en hérite implémente cette méthode, en commençant toujours par appeler celle de la classe parente. Cela permet d'ajouter, en cascade, des règles pour les types de pièces.

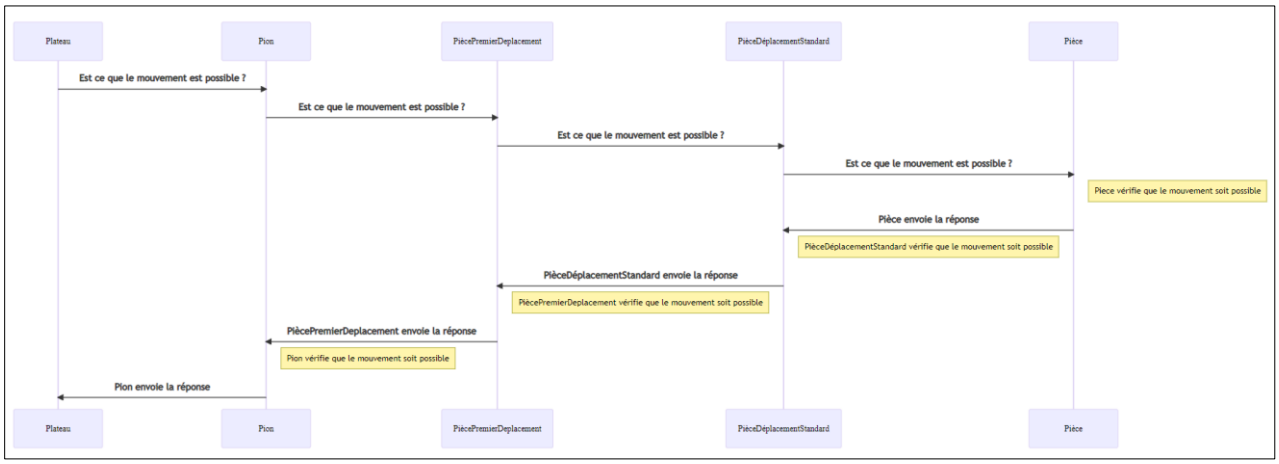


Figure 4 - Exemple dans le cas de l'appel d'un mouvement d'un pion

Dans l'exemple ci-dessus, la première vérification qui va être effectuée est celle de la classe `Piece`, qui va uniquement tester si la case de destination n'est pas occupée par une pièce de la même couleur. Ensuite, la classe `PieceDeplacementDirectionnel` va vérifier si le déplacement est légal, en prenant en compte la valeur de l'enum `DirectionType` de la pièce et sa distance max de déplacement.

La classe `PiecePremierDeplacement` n'ajoute pas de vérification, mais la classe `Pion` effectue toutes les vérifications, afin de voir s'il s'agit dune prise, d'un premier avancement de 2 cases, d'une prise en passant, etc.

A la fin, la fonction retourne au plateau une enum `MouvementType`, indiquant le type de mouvement.

Les mouvements complexes (Pions)

Le pion à des déplacement bien spéciaux. Premièrement, il peut se déplacer de 2 uniquement lors du premier tour (c'est pourquoi il hérite de la classe `PiecePremierDeplacement`).

Un booléen indique s'il a déjà effectué son premier déplacement. Par défaut, il est à `true`, mais est passé à `false` dès qu'il a effectué un déplacement.

Lors du déplacement, la fonction `mouvementPossible` vérifie s'il est en train de faire un mouvement `CLASSIQUE` (pour les mouvements normaux et les prise), `DOUBLE` ou `PROMOTION`. Si aucun de ces mouvements n'est détecté, c'est que le pion essaye de manger tout droit ou qu'il fait un mouvement `NON_VALIDE`.

Les mouvements complexes (cavalier)

Le déplacement du cavalier doit respecter une forme de L, de 2 sur 1, dans n'importe quelle direction. La fonction `mouvementPossible` du cavalier calcule donc 2 deltas, sur x et y, qui représente la différence entre la source et la destination. Pour qu'un mouvement soit valide, le delta x doit être de 2 et le delta y de 1, ou vice-versa.

La promotion

La promotion d'un pion se déroule comme suit :

La fonction `mouvementPossible` de la classe `Pion` test si le mouvement du pion courant est une promotion. Si non, on retourne un mouvement de type `CLASSIQUE` ou `NON_VALIDE` (selon le mouvement). Si oui, il s'agit bien d'une promotion, alors on retourne un mouvement de type promotion.

Pour tester si on promotion aura lieu il suffit de regarder si le pion courant s'apprête à aller sur la première ou la dernière ligne de l'échiquier. Il n'y a pas besoin de faire de distinction entre le joueur noir et blanc car les pions ne peuvent reculer dans aucun cas.

Si la fonction `move` du plateau détecte un mouvement de type promotion, alors on rentre dans la fonction `promouvoir` du plateau. Dans ce cas, on propose à l'utilisateur une dame, un cavalier, une tour ou un fou. Le pion est supprimé et remplacé par la sélection de l'utilisateur.

Mise en échec de son propre roi

Afin d'éviter de mettre en échec son propre roi, il faut tester si le mouvement actuel met le roi en échec. Dans le cas où le mouvement met ou laisse son roi en échec, il faut annuler le mouvement. En réalité, annuler un mouvement pose de nombreux problèmes d'implémentations, notamment pour les pièces qui peuvent effectuer un mouvement particulier lors de leur premier déplacement, et qui ne pourront alors pas parcourir le chemin en sens inverse. Par exemple un pion qui avance de 2 cases et met son roi en échec ne pourra pas reculer de 2 cases pour annuler son mouvement pour 2 raisons : il ne peut pas reculer et il ne peut pas se déplacer de 2 cases après son premier déplacement.

Pour résoudre ce problème, il a été décidé d'effectuer en premier lieu chaque mouvement dans une copie temporaire du plateau. Une fois le plateau effectué dans le plateau de test, l'échec de son propre roi est vérifié dans ce plateau. Si le mouvement est légal, le mouvement est effectué dans le vrai plateau. Si le mouvement ne l'est pas, un message indique que le mouvement de l'utilisateur met en échec son propre roi et le mouvement n'est pas effectué.

Cas de mise en échec durant une promotion

Ce cas est relativement rare mais il est toutefois nécessaire de le gérer correctement. En effet, une pièce ne peut pas être bougée si elle protège son roi. Dans le cas d'un pion, il ne faut pas qu'il puisse être promu s'il est actuellement en train de protéger son roi.

Notre solution est la suivante.

Dans le cas d'une promotion simple, on regarde si notre mouvement met notre propre roi en échec. Sinon, on promeut et ensuite on regarde si on met le roi adverse en échec.

Dans le cas d'une promotion qui met en danger notre roi, on interrompt la promotion grâce à la première condition (notre mouvement met notre propre roi en échec) et on ne bouge pas le pion.

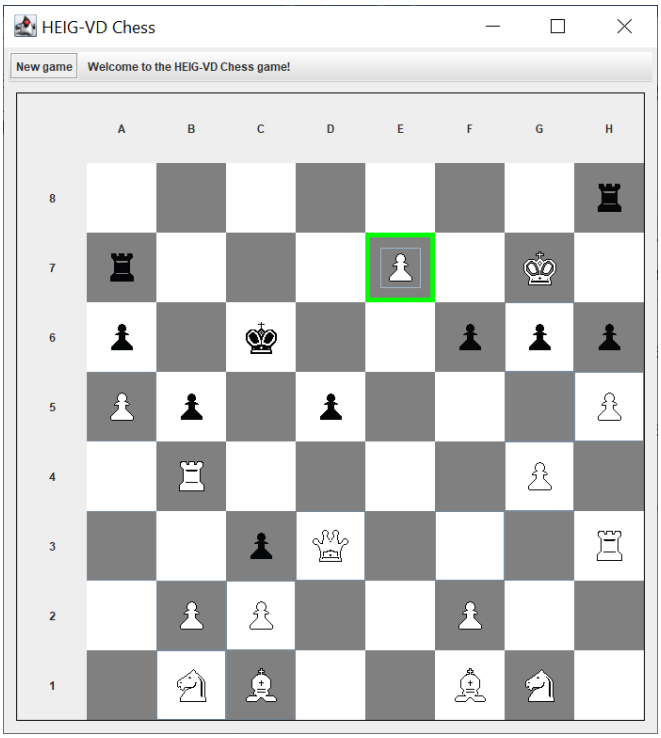


Figure 5 – Le pion blanc (E7) n'ose pas avancer en E8 et faire sa promotion car il protège son roi (G7) de la tour noir (A7)

Dans tous les cas le plateau temporaire est supprimé par le ramasse-miettes.

Détection d'un échec du roi adverse

Pour détecter l'échec d'une couleur, il faut se poser, pour chacune des pièces adverses, la question « Est-ce que cette pièce peut atteindre le roi ? ». Concrètement, il faut donc parcourir les pièces, et voir si le mouvement vers la case du roi est possible.

A chaque fin de tour, la case du roi adverse est d'abord trouvée, en parcourant l'échiquier.

Ensuite l'échiquier est parcouru à nouveau, et lorsqu'une case contient une pièce de la couleur qui vient de jouer, la pièce teste si elle pourrait se rendre sur la case du roi. Dès qu'une pièce peut atteindre le roi, un échec est détecté et la boucle s'arrête.

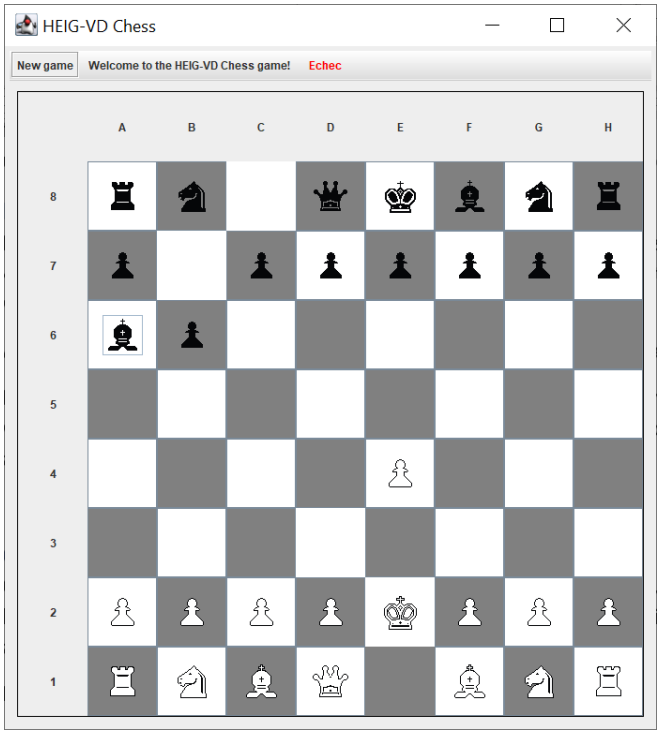


Figure 6 - Le roi blanc (E2) est mis en échec par le fou noir (A6)

La prise en passant

Lorsqu'un pion avance de 2 cases lors de son premier déplacement, le tour suivant, un pion adverse peut manger ce pion en se plaçant sur la case par laquelle il est passé.

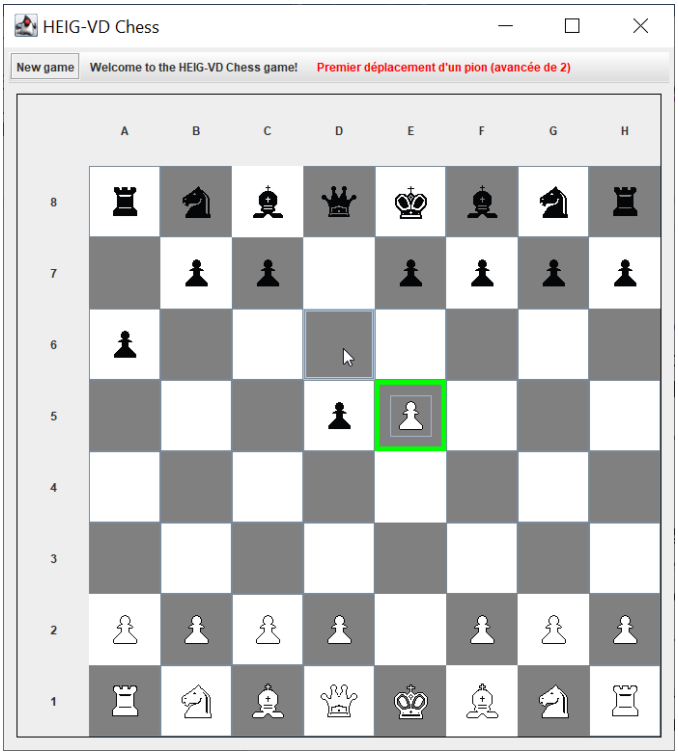


Figure 7 - Le pion blanc (E5) peut manger le pion noir (D5) en se plaçant en D6 car le pion noir a avancé de 2 cases au tour précédent

Afin d'implémenter cette règle, une case stocke à la fois une pièce courante et un pion fantôme. Le pion fantôme garde une référence sur un pion qui est passé par cette case en avançant de 2. Cette case est attribuée par le plateau, quand le type de mouvement du pion est un mouvement DOUBLE.

A chaque tour, l'échiquier est parcouru et tous les pions fantômes sont effacés.

Lorsqu'un pion se place sur une case, en plus des vérifications de prise habituelles, il est aussi vérifié si la case contient un pion fantôme, dans quel cas le pion originel est mangé.

Le grand et le petit roque

Le roi ne peut se déplacer que d'une case dans toutes les directions, sauf dans le cas où il effectue un roque. Le roque se produit lorsque lors de son premier déplacement, il se déplace de 2 cases sur la gauche ou la droite. Si la tour du côté du déplacement n'a pas encore bougé, et que ni la case parcourue par le roi, ni la destination ne mettent le roi en échec, et qu'il n'y a aucune autre pièce entre le roi et la tour, alors la tour vient se placer de l'autre côté du roi. Sinon, le roi ne peut pas se déplacer de 2 cases. Le petit roque se produit quand le roi se dirige vers la droite, et le grand roque quand le roi se dirige vers la gauche.

C'est la fonction `mouvementPossible` du `Roi` qui détecte une tentative de tentative de roque, lorsque le roi n'a pas encore bougé et qu'il tente de faire un mouvement de 2 sur la gauche ou de 2 sur la droite. Le plateau teste alors d'effectuer le roque dans une copie temporaire du plateau, en vérifiant que la tour n'ait pas encore bougéet que les cases entre la tour et le roi soient libres et ne mette pas en échec le roi. Si c'est le cas, le plateau déplace manuellement la tour vers sa nouvelle case.

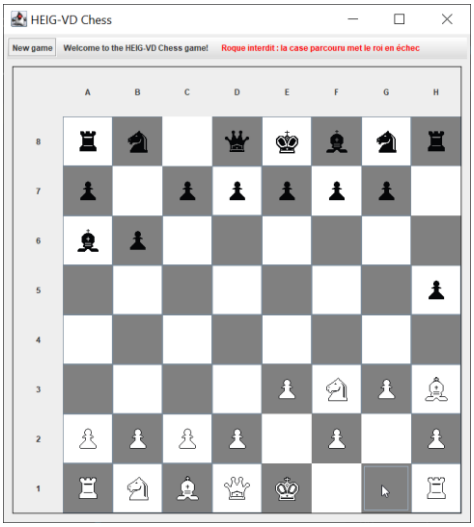


Figure 8 - Dans cet exemple, le roi blanc ne peut pas effectuer de petit roque (G1) car la case qu'il parcourrait sans s'y arrêter (F1) le mettrait en échec par le fou noir (A6)

La fonction « move » du plateau

Cette fonction nous sert principalement à vérifier si le tour qui est en train d'être joué est correct.

Elle nous permet notamment de :

- Savoir qui est en train de jouer (blanc ou noir)
- Traiter tous les types de mouvements
- Tester si les trajectoires des mouvements sont correctes
- Mettre à jour l'échiquier et l'interface graphique
- Détecter les prises en passant
- Incrémenter le nombre de tours
- Détecter un échec et restreindre les mouvements en fonction

Cette fonction retourne `true` quand elle valide un tour. Ainsi le tour effectué et on passe au prochain tour. Elle retourne false si elle détecte quelque chose de non valide et recommence le tour.

Spécificités

La classe `Piece` implémente `ChessView.UserChoice` pour nous permettre de donner le choix de sélectionner une pièce à l'utilisateur lors d'une promotion.

```
public abstract class Piece implements ChessView.UserChoice{
```

Remarque et conclusion

Arrivé au moment de l'implémentation de la détection d'un échec, nous nous sommes questionnés sur la création d'une classe `Mouvement`. En effet pour vérifier la mise en échec du roi on doit tester un coup en avance. Si le coup n'est pas permis on doit faire revenir la pièce en arrière à l'aide d'une éventuelle fonction "annuler". Or, ça n'a pas de sens d'annuler une pièce, mais plutôt un mouvement. Cela justifierait donc la création d'une classe mouvement.

Malheureusement, notre projet était déjà trop avancé et cela nous aurait pris trop de temps à tout devoir réimplémenter. Nous avons donc pu constater une éventuelle autre direction à prendre, mais nous avons décidés de nous diriger vers la solution expliquée dans la section « Mise en échec de son propre roi ».

L'ensemble de la réalisation de ce laboratoire s'est bien passé. La réalisation du rapport ainsi que la création de l'UML nous ont permis d'avoir une bonne vue d'ensemble sur notre projet.

Test de comportements

Test	Contexte	Résultat attendu	Résultat obtenu
Cliquer sur « New game »		L'échiquier est rempli	OK
Jouer 2 coups d'affilés avec la même couleur de pièces		Le deuxième coup ne s'effectue pas	OK
Déplacer un pion d'une case en avant	La case devant le pion est libre	Le pion avance	OK
Déplacer un pion de deux cases en avant et le pion n'a jamais avancé	Les 2 cases devant le pion sont libres	Le pion avance La vue affiche que c'est le premier déplacement d'un pion	OK
Déplacer un pion de deux cases en avant et le pion a déjà avancé pendant la partie		Le pion n'avance pas	OK
Déplacer un pion de deux cases en avant	La case juste devant le pion est occupée	Le pion n'avance pas	OK
Déplacer le pion en diagonale en avant d'une case	La case en diagonale en avant est vide	Le pion n'avance pas	OK
Déplacer le pion en diagonale en avant d'une case	La case en diagonale en avant est occupée par une pièce adverse	Le pion avance et mange la pièce adverse	OK
Déplacer le pion de 2 en avant et de 1 sur la gauche ou la droite (mouvement en L)	La case de destination est vide	Le pion n'avance pas	OK
Déplacer le pion de 2 en avant et de 1 sur la gauche ou la droite (mouvement en L)	La case de destination est occupée par une pièce ennemie	Le pion n'avance pas	OK
Le pion blanc avance de deux cases et le tour suivant, un pion noir se place sur la case par laquelle le pion blanc est passé (prise en passant)		Le pion noir mange le pion blanc La vue indique que c'est une prise en passant	OK
Le pion blanc avance de deux cases et le tour suivant, une pièce noire autre qu'un pion se place sur la case par laquelle le pion blanc est passé (prise en passant)		La pièce noire ne mange pas le pion blanc	OK
Le pion blanc avance de deux cases et 2 tours après, un pion noir se place sur la case par laquelle le pion blanc est passé (prise en passant)		Le pion noir mange le pion blanc	OK
Déplacer un pion d'une case en avant sur la dernière ligne de l'échiquier	Le pion protège son propre roi de la mise en échec	Le pion n'avance pas	OK

		La vue indique qu'il est interdit de mettre son roi en échec	
Déplacer un pion d'une case en avant sur la dernière ligne de l'échiquier		La boîte de dialogue permet à l'utilisateur de choisir une nouvelle pièce La vue indique que c'est une promotion	OK
L'utilisateur choisit une nouvelle pièce	(Lorsque le pion a atteint la dernière ligne)	Le pion est remplacé par la nouvelle pièce	OK
Déplacer le fou en ligne droite		Le fou n'avance pas	OK
Déplacer le fou en diagonale	La case de destination et les cases parcouru sont vides	Le fou avance	OK
Déplacer le fou en diagonale	La case de destination est vide mais une des cases parcourues est occupée par une pièce	Le fou n'avance pas	OK
Déplacer le fou en diagonale	La case de destination est occupée par une pièce adverse	Le fou avance et mange la pièce	OK
Déplacer la tour en diagonale		La tour n'avance pas	OK
Déplacer la tour en ligne droite	La case de destination et les cases parcouru sont vides	La tour avance	OK
Déplacer la tour en ligne droite	La case de destination est vide mais une des cases parcourues est occupée par une pièce	La tour n'avance pas	OK
Déplacer la tour en ligne droite	La case de destination est occupée par une pièce adverse	La tour avance et mange la pièce	OK
Déplacer le cavalier en ligne droite		Le cavalier n'avance pas	OK
Déplacer le cavalier en diagonale		Le cavalier n'avance pas	OK
Déplacer le cavalier en forme de L de 2 de haut et 1 de large, ou l'inverse	La case de destination est vide	Le cavalier se déplacer	OK
Déplacer le cavalier en forme de L de 2 de haut et 1 de large, ou l'inverse	La case de destination est occupée par une pièce de la couleur de cavalier	Le cavalier ne se déplace pas	OK
Déplacer le cavalier en forme de L de 2 de haut et 1 de large, ou l'inverse	La case de destination est occupée par une pièce adverse	Le cavalier mange la pièce et se déplace	OK
Déplacer la reine en diagonale	La case de destination et le chemin parcouru sont vides	La reine avance	OK

Déplacer la reine en ligne droite	La case de destination et le chemin parcouru sont vides	La reine avance	OK
Déplacer la reine en forme de L		La reine n'avance pas	OK
Déplacer la reine en ligne droite ou en diagonale	La case de destination est occupée par une pièce de la même couleur	La reine n'avance pas	OK
Déplacer la reine en ligne droite ou en diagonale	Le chemin parcouru est occupé par une ou plusieurs pièces	La reine n'avance pas	OK
Déplacer le roi d'une case en ligne droite	La case de destination est vide	Le roi avance	OK
Déplacer le roi d'une case en diagonale	La case de destination est vide	Le roi avance	OK
Déplacer le roi d'une case en ligne droite	La case de destination est occupée par une pièce adverse	Le roi avance et mange la pièce	OK
Déplacer le roi d'une case en diagonale	La case de destination est occupée par une pièce adverse	Le roi avance et mange la pièce	OK
Déplacer le roi d'une case en ligne droite	La case de destination est occupée par une pièce de la même couleur	Le roi n'avance pas	OK
Déplacer le roi d'une case en diagonale	La case de destination est occupée par une pièce de la même couleur	Le roi n'avance pas	OK
Déplacer le roi de plus d'une case en ligne droite		Le roi n'avance pas	OK
Déplacer le roi de plus d'une case en diagonale		Le roi n'avance pas	OK
Déplacer le roi de 2 cases sur la droite	Le roi n'a pas encore bougé La tour du coin droit n'a pas encore bougé Les 2 cases sur la droite du roi sont libres La case sur la droite du roi ne mettrait pas le roi en échec	Le roi se déplace de 2 cases sur la droite et la tour du coin droit vient se placer à sa gauche La vue indique que c'est un petit roque	OK
Déplacer le roi de 2 cases sur la droite	Une des conditions du test précédent n'est pas respectée	Le roi ne se déplace pas La vue indique la raison de l'invalidité du roque	OK
Déplacer le roi de 2 cases sur la droite	Le roi n'a pas encore bougé	Le roi se déplace de 2 cases sur la gauche et la	OK

	La tour du coin gauche n'a pas encore bougé Les 2 cases sur la gauche du roi sont libres La case sur la gauche du roi ne mettrait pas le roi en échec	tour du coin gauche vient se placer à sa droite La vue indique que c'est un grand roque	
Déplacer le roi de 2 cases sur la gauche	Une des conditions du test précédent n'est pas respectée	Le roi ne se déplace pas La vue indique la raison de l'invalidité du roque	OK
Déplacer n'importe quelle pièce	Le roi adverse est en échec après le déplacement	La vue indique que le roi adverse est en échec	OK
Déplacer n'importe quelle pièce	Le roi de la même couleur est mis en échec	La pièce ne se déplace pas La vue indique qu'il est interdit de mettre son roi en échec	OK

Malo Romano & Enzo Allemano
POO - Labo 8 Échecs
13.01.2022