Développement orientée objet - Java
TP 06 - Partie 4

Lionnel Conoir, Isabelle Delignières, Wajdi Elleuch
Rémi Synave et Franck Vandewiele

Analyse de la situation

En ayant terminé les trois premières parties, vous avez un jeu quasiment fonctionnel. Vous avez un plateau sur lequel vous pouvez déplacer les pièces.

L'objectif de cette partie est de développer une méthode permettant de vérifier la mise en échec du roi. Il ne restera plus ensuite qu'à modifier votre application graphique pour gérer l'alternance des joueurs et un avertissement en cas de mise en échec.

La classe `Plateau`

Dans cette classe, vous allez ajouter une méthode `estEchec` prenant en paramètre une couleur. La méthode doit retourner vrai si le roi dont la couleur est passée en paramètre est en échec.

Avant de développer cette méthode, il est conseillé de créer une méthode `getRoi` prenant en paramètre une couleur. Cette méthode doit simplement parcourir le tableau de pièces et retourner le roi de la couleur choisie.

Pour vérifier si le roi est en échec, vous pouvez simplement récupérer la liste des pièces de la couleur adverse (en utilisant les méthodes `getPiecesBlanches` ou `getPiecesNoires`) puis de parcourir leurs coups possibles. Si l'un des coups possibles de l'une des pièces adverses est la prise du roi alors celui ci est en échec.

Le classe `MainGraphique`

Modifiez votre interface graphique pour que la partie se joue avec alternance des deux joueurs et que la mise échec d'un roi soit notifié par un message dans le terminal ou une mise en évidence sur l'échiquier (en changeant la couleur de fond de la case par exemple).

Question subsidiaire

Il n'est pas demandé de coder la réponse à cette question (sauf si vous êtes motivés). Le roque consiste à "échanger" les positions du roi et d'une tour. Le roque ne peut se faire que sous plusieurs conditions dont : le roi ne doit pas avoir été mis en échec précédemment dans la partie et ne doit pas avoir bougé.

Comment feriez vous pour savoir si, à un moment donné dans la partie, ces deux conditions sont respectées ?

Pour aller encore plus loin

Vous pouvez encore ajouter les deux fonctionnalités suivantes :
— Obliger le joueur dont le roi est en échec de se déséchequiser par son prochain coup.
— Empêcher le joueur de jouer un coup qui mettrait son roi en échec.

Pour ajouter ces deux fonctionnalités, vous devrez "prévoir" la situation après le coup demandé. Vous allez donc avoir besoin de copier le plateau pour y jouer le coup et vérifier la mise en échec ou non d'un roi. La copie du plateau doit être effectuée au travers du constructeur par copie. Toutefois, un objet de type `Plateau` contient un tableau d'objets de type `Piece` qui doit être copié. Hors, les pièces contenues dans ce tableau sont toutes des pions, tours, cavaliers, fous, rois ou dames.
Deux possibilités :
— Vous pouvez connaitre le type précis d'une pièce en utilisant l'opérateur `instaceof` et ainsi créer les bonnes pièces.
— Vous pouvez également vous tourner vers la méthode `clone` de l'objet `ArrayList`. Il faudra, dans ce cas, que la classe `Piece` implémente l'interface `Cloneable` en redéfinissant la méthode `clone` dans `Piece` et toutes ses sous-classes.