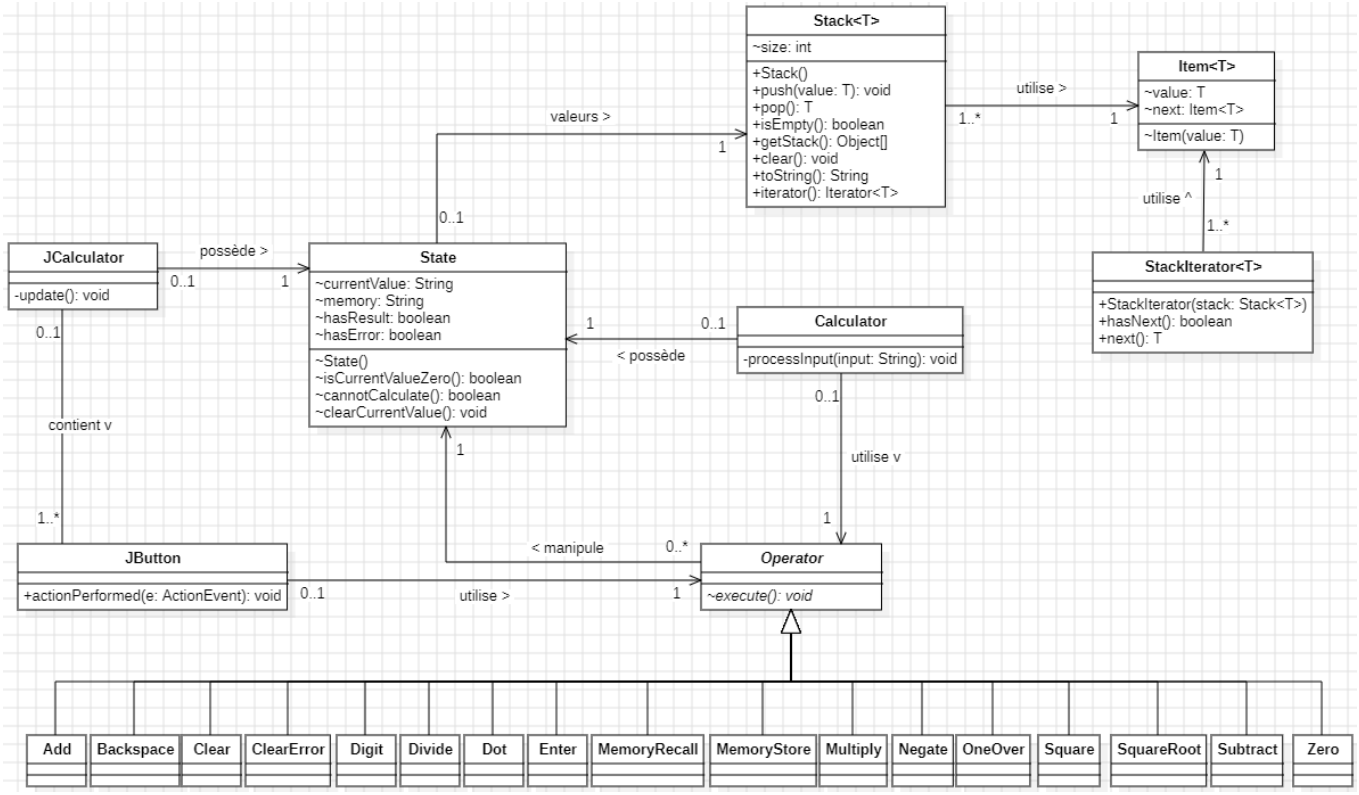
Labo 8 - Jeu d'échecs

- Groupe : L08GrJ
- Etudiants : Calum Quinn, Dylan Ramos

Diagramme de classes



Choix d'implémentation

- La classe abstraite `SpecialPiece` regroupe les pièces ayant des intéractions avec d'autres pièces durant un de leur déplacement.
  - `Pawn` doit évaluer l'état d'un autre pion pour savoir s'il peut effectuer une capture "en passant".
  - `Rook` doit ne pas avoir bougé depuis le débût du jeu pour pouvoir effectuer un roque.
  - `King` doit être immobile depuis le débût de la partie pour pouvoir procéder à un roque.
- La position de chaque pièce est stockée en tant qu'index sur le plateau. Ceci à comme avantage de ne garder qu'une seule structure de données pour lister les pièces et les cases.
- La cardinalité de 0..32 sur `Piece` nous permet de créer un `Board` sans pièces et rappelle le nombre de pièces maximum dans un jeu d'échecs.
- Les interfaces `AddListener`, `CaptureListener`, `PromoteListener` et `CastleListener` ont été déclarées à l'intérieur de la classe `Board` pour mieux les encapsuler. Ces interfaces sont principalement utilisées pour appeler les méthodes de la classe `ChessView` sans avoir un lien direct entre la classe utilitaire `Board` et la classe de gestion `Game`.
- L'attribut `BOARD_SIZE` de `Board` est constant et permet de définir la taille du plateau de jeu qui est de 8x8. De plus, cet attribut est statique car il est commun à toutes les instances de `Board`.
- La classe abstraite `Piece` contient une methode `textValue` qui sert à rendre un String du nom de la pièce pour l'affichage graphique. Ceci permet notemment de proposer à l'utilisateur, les différents choix de promotion pour un pion.
- La classe `Pawn` à un attribut `lastMoveDist` pour savoir si un pion à bougé d'une ou deux cases lors de son déplacement précédent. Ceci sert bien sûr à contrôler si une capture "en passant" et possible.
- La classe `Board` contient la méthode `isCheckMate` car, même si ce n'était pas demandé, c'est une des deux seuls façon de finir une partie d'échecs. L'autre manière étant le match nul, ce qui demande une analyse profonde de nombreux coups en avance pour prédire. La méthode `isCheckMate` contrôle donc qu'il n'y a aucun déplacement d'aucune pièce qui puisse protéger le roi d'une attaque le coup suivant.

Tests effectués

- Pour chaque pièce nous avons testé que seuls les mouvements autorisés sont possibles.
  - Pour toutes les pièces une `capture` survient quand la pièce se déplace sur une case où il y a déjà une autre pièce. Ceci n'est que possible si l'autre pièce est de la couleur opposée.
  - Pion:
    - 1 case vers l'avant si aucune pièce se trouve à l'arrivée.
  - 2 cases vers l'avant si aucune pièce se trouve à l'arrivée et que c'est le premier déplacement du pion.
  - 1 case en diagonal si c'est une capture.
  - La prise en passant n'est possible que si le pion a capturé s'est déplacé de deux cases le tour d'avant et est arrivé à côté du pion avec laquelle on va faire la capture.
  - Tour (le roque n'est pas pris en compte ici car ça doit être commencé par le roi):
    - Autant de cases que voulu tant que le déplacement et parallèle aux cases et qu'il n'y a pas de pièces entre le départ et l'arrivée.
  - Chevalier:
    - N'importe quel déplacement qui consiste de 2 cases soit horizontalement soit verticalement et 1 case dans la direction opposé.
  - Fou:
    - Autant de cases que voulu tant que le déplacement et diagonal aux cases et qu'il n'y a pas de pièces entre le départ et l'arrivée.
  - Reine:
    - Les mêmes conditions que pour la tour.
  - Les mêmes conditions que pour le fou.
  - Roi:
  - 1 case en parallèle ou en diagonal.
  - Grand et petit roque. Pour faire le roque l'utilisateur doit cliquer sur le roi et ensuite cliquer deux cases à droite ou à gauche de sa position ou bien il peut cliquer sur le roi et ensuite sur la tour avec lequel il veut effectuer le roque.
  - Le roque n'est que possible selon les conditions satandards:
    - Le roi et la tour en question ne se sont jamais déplacés avant.
    - Il n'y a aucune pièce entre le roi et la tour.
    - Le roi ne passe sur aucune case où il serait en échec.
- Si l'utilisateur essaie de faire un déplacement illégal, la pièce n'est simplement pas déplacée et l'utilisateur doit rechoisir un déplacement.
- Lorsqu'un pion arrive à la dernière case du plateau, un menu déroulant s'affiche et l'utilisateur choisi en quel pièce il veut promouvoir le pion, la pièce affichée change et peut se déplacer avec les règles adaptées au type de pièce choisi.
- Si l'utilisateur clique sur une pièce à déplacer et ensuite clique à nouveau sur la même case, rien n'est déplacé et ça reste le tour de la même couleur.
- Seulement le joueur en blanc peut jouer le premier tour.
- Après un déplacement seulement le joueur opposé peut déplacer une pièce.
- Avant chaque déplacement, le message " to play" est affiché en haut de l'écran.
- Lors de chaque déplacement nous controllons que le roi du joueur actuel n'est pas mis en échec, si c'est le cas la pièce n'est pas déplacée.
- Pour chaque déplacement nous controllons si le roi du joueur opposé est mis en échec, un message "Check!" s'affiche si c'est le cas.
- Si le roi du joueur actuel est en échec, les seuls déplacements possibles sont ceux qui sortent le roi de l'échec.
- Après chaque déplacement, si le roi est en échec on contrôle s'il y a un quelconque déplacement légal permettant de sortir le roi de l'échec, si ce n'est pas le cas le message "Checkmate wins!" s'affiche et plus personne ne peut jouer.
- Quand l'utilisateur clique sur "New Game", les pièces se remettent à la bonne place et c'est à nouveau aux blancs de jouer.