Classe Game :
. Attributs :
    - nbActions : int symbolisant le nombre d'actions que l'on peut effectuer par tour.
    - nbTours : int symbolisant le nombre de tours effectués par les deux joueurs.
    - player1 : Player désigne le premier joueur (par convention il jouera en bas)
    - player2 : Player désigne le second joueur (par convention il jouera en haut)
    - currentPlayer : Player désigne le joueur qui joue actuellement
    - gameBoard : GameBoard symbolise le plateau de jeu, la gestion des déplacements et autres actions "physique" lui est déléguée
. Méthodes :
    - Game(String valeursDuJeu) : constructeur de la classe Game à partie de données en String (Json...).
    - hasWon() : fonction qui renvoit true si current_player a fait un mouvement gagnant, false sinon.

Classe GameBoard extends Do :
. Attributs :
    - BOUNDARY : constante correspondant à la largeur du plateau de jeu.
    - undoable_move : pile composée d'ActionCoord contenant les potentiels mouvement que l'on peut annuler.
    - redoable_move : pile composée d'ActionCoord contenant les potentiels mouvement que l'on souhaite refaire.
    - board : list de Pawn représentant l'état du jeu. Ils seront rangés de façon à ce qu'un pion situé en (x,y) soit accessible à la case x+y*7 (x et y allant de 0 à 6)
. Méthodes :
    - GameBoard(p1 : Player, p2 : Player) : constructeur de la classe Gameboard. Créé en plus les pions qui seront affectés aux deux joueurs.
    - getPawn(c : Coordinate) : fonction qui renvoit un Optionnal<Pawn> composé d'un pion si celui-ci possède la coordonnée donnée en paramètre, ou null sinon.

Classe abstraite Do implements Undo :
. Méthodes :
    - move(player : Player, coords : ActionCoord) : fonction qui effectue le mouvement symbolisé par coords pour player. Renvoie true si le mouvement effectué est valide, renvoie false sinon (le mouvement n'était pas valide).
    - canMove(player : Player, coords : ActionCoord) : fonction qui renvoit un booléen selon si coords constitue un mouvement légal pour player.

Interface Undo :
. Méthodes :
    - undo() : défait le dernier mouvement effectué par le Player courant.
    - redo() : refait le dernier mouvement annulé par le Player courant.

Classe abstraite Player :
. Attributs :
    - nom : nom du joueur sous format String.
    - colour : couleur des pions du joueur sous format booléen : true si le joueur a les pions blanc, false si il a les pions noir.
    - pieces : List de Pawn, symbolisant les différentes pièces du joueur. Est complémentaire avec le board du gameBoard dans le sens où ce dernier permet un accès rapide à une case bien déterminée, alors que pieces permet un accès rapide à la liste des pions d'un joueur bien déterminé (liste non nécessaire mais pratique et efficace pour programmer des IA)
    - ball : Pawn symbolisant la balle du joueur
. Méthodes :
    - addPawn(p : Pawn) : ajoute p au tableau pieces afin que le joueur connaisse ses pions
    - setBall(p : Pawn) : setter de l'attribut ball avec le Pawn p, afin de spécifier que c'est ce pion qui possède la balle en ce moment
    - getBall() : getter de ball.
    - getMove() : fonction qui renvoit le mouvement d'un joueur sous la forme d'une instance de ActionCoord
    - waitEndOfTurn() : booléen notifiant de la fin du tour de Player. Il s'agit d'un processus d'attente que le Game lance afin que le joueur puisse appuyer sur le bouton "fin du tour".

Classe HumanPlayer extends Player :
. Méthodes :
    - HumanPlayer(nom : String, colour : boolean) : constructeur de HumanPlayer lui affectant

```
    un nom et une couleur

Classe AIPlayer extends Player :
. Attributs :
    - current_turn : int symbolisant le nombre de tours joués.
    - TURNS_BEFORE_SWAP : int symbolisant le nombre de tours à partir duquel la difficulyé de
    l'IA doit changer dans le cas d'une IA progressive
    - algo : instance de la classe Algo, correspondant à l'algorithme qui va déterminer les
    mouvements de l'IA
. Méthodes :
    - AIPlayer(type : EAiType) : constructeur de AIPlayer. Prend en paramètre le niveau de l'IA
    souhaité parmis un Enum créé pour
    - setAlgo(type : EAiType) : setter de algo. Précise le niveau de l'IA souhaité
    - setboard(board : GameBoard) : méthode qui transmet board à algo

Classe Algo :
. Attributs :
    - gameBoard : GameBoard représente l'état du jeu. L'algo en a besoin afin de proposer des
    mouvements plus ou moins futés
. Méthodes :
    - decideMove() : attend la réflexion de l'IA et renvoie sa décision sous forme d'un
    ActionCoord
    - Algo(player : Player) : constructeur définissant pour l'algorithme le joueur pour lequel
    il fera ses calculs
    - setBoard(board : gameBoard) : set le board à l'algo

Classe Pawn :
. Attributs :
    - position : Coordinate représente la position actuelle du pion
    - haveBall : boolean est à true si le pion a la balle de son joueur, false sinon
    - player : Player est le joueur qui possède ce pion
. Méthodes :
    - Pawn(p : Player, c : Coordinate) : constructeur du pion qui lui affecte un joueur et des
    coordonnées
    - getPosition() : Coordinates renvoie la position du pion sous forme de coordonnées
    - haveBall() : boolean renvoie true si le pion a la balle de son joueur
    - getPlayer() : Player renvoie le joueur contrôlant le pion
    - setBall(have : boolean) : permet de dire au pion s'il a la balle ou non

Classe Coordinate :
. Attributs :
    - posX : position X des coordonnées (horizontal)
    - posY : position Y des coordonnées (vertical)
. Méthodes :
    - getX() : int renvoie la coordonnée X
    - getY() : int renvoie la coordonnées Y
    - Coordinate (x : int, y : int) : constructeur du pion avec les coordonnées initiales
    - moveTo(c : Coordinate) : permet de déplacer les coordonnées de façon relative

Classe ActionCoord :
. Attributs :
    - source : Coordinate représente la position d'origine d'un déplacement
    - target : Coordinate représente la position d'arrivée d'un déplacement
. Méthodes :
    - getSource() : Coordinate permet d'obtenir l'origine d'un déplacement
    - getTarget() : Coordinate permet d'obtenir l'arrivée d'un déplacement
    - ActionCoord(s : Coordinate, t : Coordinate) : constructeur créant ActionCoord avec la
    source et l'arrivée

Classe PlayerFactory :
. Attributs :
    - PLAYERLIMIT : int détermine le nombre max de joueurs (2)
. Méthodes :
    - createHuman(nom : char, colour : boolean) : Player crée un joueur de type humain avec un
    nom et une couleur
    - createAi(type : EAiType) : Player crée un joueur de type AI avec l'enum adapté
```

enum : EAiTypes :
. Valeurs possibles :
    -AI_NOOB
    -AI_STARTING
    -AI_PROGRESSIVE