**class** Board

**types**

**public** PieceType = **<LANCE>** | **<BISHOP>** | **<GOLDGENERAL>** | **<SILVERGENERAL>** | **<KING>** | **<KNIGHT>** | **<PAWN>** | **<ROOK>** | **<NONE>**;

**public** Player = **<White>** | **<Black>**;

**values**

**private** startPositions = [

[**<LANCE>**, **<KNIGHT>**, **<SILVERGENERAL>**, **<GOLDGENERAL>**, **<KING>**, **<GOLDGENERAL>**, **<SILVERGENERAL>**, **<KNIGHT>**,**<LANCE>**],

[**<NONE>**, **<BISHOP>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<ROOK>**, **<NONE>** ],

[**<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>** ],

[**<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>** ],

[**<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>** ],

[**<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>** ],

[**<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>**, **<PAWN>** ],

[**<NONE>**, **<ROOK>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<NONE>**, **<BISHOP>**,**<NONE>** ],

[**<LANCE>**, **<KNIGHT>**, **<SILVERGENERAL>**, **<GOLDGENERAL>**, **<KING>**, **<GOLDGENERAL>**, **<SILVERGENERAL>**, **<KNIGHT>**,**<LANCE>**]

]

**instance** **variables**

-- TODO Define instance variables here

**private** capturedPieces : **map** Player **to** **seq** **of** Piece;

**private** lines: **map** **nat1** **to** BoardLine;

**public** turn: Player;

**public** gameEnded: **bool**;

**inv** **card**(**dom**(lines)) = 9 **and** **dom**(lines) = {1,...,9};

**private** **static** board:Board := **new** Board();

**operations**

**protected** Board: () ==> Board

Board() ==

(

lines:= {y |-> **new** BoardLine( mapStartSeq(startPositions(y), 1, y), y ) | y **in** **set** {1,...,9}};

capturedPieces := {**<White>**|->[], **<Black>**|->[]};

turn:= **<White>**;

gameEnded:=**false**;

);

**public** **static** getInstance: () ==> Board

getInstance() == **return** board;

**public** getCell: **nat1**\***nat1** ==> Cell

getCell(x,y) == **return** lines(y).getCell(x);

**public** dropPiece: Cell\*Piece ==> ()

dropPiece(c,p) ==

(

c.piece:= p;

capturedPieces:= capturedPieces ++ {turn |-> capturedPieces(turn)(1,...,getIndex[Piece](p,capturedPieces(turn))-1) ^

capturedPieces(turn)(getIndex[Piece](p,capturedPieces(turn)) + 1,...,**len** capturedPieces(turn))};

**if** turn = **<White>** **then** turn := **<Black>** **else** turn := **<White>**

)

**pre** c.piece = **nil** **and** p **in** **set** **elems**(capturedPieces(turn)) --Falta verificar condições mais complexas referentes a checkmates e pawns

**post** p **not** **in** **set** **elems**(capturedPieces(turn)) **and** c.piece = p **and** turn **<>** turn~;

**public** move: Cell\*Cell ==> ()

move(c1,c2) ==

(

**if** c2.piece **<>** **nil**

**then** (c2.piece.capture(); capturedPieces:= capturedPieces ++ {turn |-> capturedPieces(turn) ^ [c2.piece]}; );

c2.piece:= c1.piece;

c1.piece:= **nil**;

**if** turn = **<White>** **then** turn := **<Black>** **else** turn := **<White>**

)

**pre** c1 = getCell(c1.x, c1.y) **and** c2 = getCell(c2.x,c2.y) **and** c1.piece **<>** **nil** **and** c1.piece.validMovement(c1,c2) **and** c1.piece.belongTo = turn

**post** c1.piece = **nil** **and** turn **<>** turn~;

**public** getCapturedPieces: () ==> **map** Player **to** **seq** **of** Piece

getCapturedPieces() == **return** capturedPieces;

**public** check: Player ==> **bool**

check(p) == **if** **exists** c **in** **set** getAllCellsFromPlayer(getOtherPlayer(p),1) & c.piece.validMovement(c,getKingFromPlayer(p))

**then** **return** **true**

**else** **return** **false**;

**public** endGame:()==>()

endGame() == gameEnded:=**true**

**pre** check(turn) **and** **not** **exists** c **in** **set** getAllEmptyCells(1) & getKingFromPlayer(turn).piece.validMovement(getKingFromPlayer(turn),c)

**post** gameEnded=**true**;

**private** getAllEmptyCells: **nat1** ==> **set** **of** Cell

getAllEmptyCells(y) ==

(

**if** y = 9 **then** **return** lines(y).getEmptyCells()

**else** **return** lines(y).getEmptyCells() **union** getAllEmptyCells(y+1);

);

**public** getAllCellsFromPlayer: Player\***nat1** ==> **set** **of** Cell

getAllCellsFromPlayer(p,y) ==

(

**if** y = 9 **then** **return** lines(y).getCells(p)

**else** **return** lines(y).getCells(p) **union** getAllCellsFromPlayer(p,y+1);

);

**public** getKingFromPlayer: Player ==> Cell

getKingFromPlayer(p) ==

(

**let** c **in** **set** getAllCellsFromPlayer(p,1) **be** **st** **isofclass**(King,c.piece)

**in** **return** c;

)

**functions**

**private** **static** getOtherPlayer: Player -> Player

getOtherPlayer(p) == **if** p = **<White>** **then** **<Black>** **else** **<White>**;

**public** **static** getIndex[@T]: @T \* **seq** **of** @T -> **nat1**

getIndex(t,s) == **let** n **in** **set** **inds**(s) **be** **st** s(n) = t **in** n

**pre** **exists1** n **in** **set** **inds**(s) & s(n) = t;

**private** mapStartSeq: **seq** **of** PieceType\***nat1**\***nat1** -> **seq** **of** Cell

mapStartSeq(s, x, y) ==

(

**if** s = [] **then** []

**elseif** **hd** s = **<LANCE>** **then** [**new** Cell(x,y,**new** Lance(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<KNIGHT>** **then** [**new** Cell(x,y,**new** Knight(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<SILVERGENERAL>** **then** [**new** Cell(x,y,**new** SilverGeneral(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<GOLDGENERAL>** **then** [**new** Cell(x,y,**new** GoldGeneral(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<KING>** **then** [**new** Cell(x,y,**new** King(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<BISHOP>** **then** [**new** Cell(x,y,**new** Bishop(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<ROOK>** **then** [**new** Cell(x,y,**new** Rook(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**elseif** **hd** s = **<PAWN>** **then** [**new** Cell(x,y,**new** Pawn(zone(y)))] ^ mapStartSeq(**tl** s, x+1, y)

**else** [**new** Cell(x,y)] ^ mapStartSeq(**tl** s, x+1, y)

)

**pre** y > 0 **and** y < 10 **and** x > 0 **and** x < 10 **and** **len** s = 10 - x;

**private** zone: **nat1** -> [Player]

zone(y) ==

(

**if** y < 4 **then** **<White>**

**elseif** y > 6 **then** **<Black>**

**else** **nil**

);

**end** Board