**Joan Estop Cepero**

**Dimecres 10:30**

**SHOGUI**

**Funcionalitat**: Constructor de la classe Abstracte Piece

**Localització**: Model/Piece.java/piece()

**Test**: ModelTests/PieceTest.java/testConstructor()

En la primera versió només faig els tests per comprovar que els valors del constructor són correctes, però unes hores més tard em vaig adonar que havia permès que el valor de les Y i de les X no fossin entre 0-8. S’actualitza tant els tests com el codi per no permetre valors fora del rang. Es fan els tests necessaris per assegurar el funcionament tant en els valors límit i frontera com en les particions equivalents.

**Funcionalitat**: SetPosition (mètode de la classe Piece). Permet donar-li unes coordeandes a una peça.

**Localització**: Model/Piece.java/SetPosition(int newX, int newY)

**Test**: ModelTests/PieceTest.java/testSetPosition()

Desenvolupem els tests per comprovar que no podem ficar una fitxa fora del tauler. S’assegura el correcte funcionament del codi en les particions equivalents i en els valors límit i frontera. Codi igual que l’utilitzat per fer el constructor.

**Funcionalitat**: move (mètode de la classe Piece). Canvia les coordenades d’una peça.

**Localització**: Model/Piece.java/move(int newX, int newY)

**Test**: ModelTests/PieceTest.java/testmove()

Exactament igual que en el mètode SetPiece.

**Funcionalitat**: constructor subclasse Pawn (mètode de la subclasse Pawn)

**Localització**: Model/pieces/Pawn.java/pawn()

**Test**: ModelTests/pieces/PawnTest.java/testConstructorPawn()

Utilitzem el mateix codi tant en el test com en el codi normal implementat en la classe mare.

**Funcionalitat**: isValidMove() (mètode de la subclasse Pawn). Aquest mètode identifica si el moviment que vol fer el jugador és vàlid o no, en funció dels moviments vàlids dels peons i de la resta de caselles del tauler.

**Localització**: Model/pieces/Pawn.java/isValidMove()

**Test**: ModelTests/pieces/PawnTest.java/testIsValidMove()

Per a la implementació dels tests d’aquest mètode desenvolupem un mockObject de la classe Board, que encara no està implementada. D’aquesta manera ens estalviem implementar-la i amés a més ens serveix per col·locar les fitxes d’una manera que ens permetin realitzar els tests per valors límit i frontera, particions equivalents, statement coverage, decision coverage, condition coverage i path coverage. En els tests fem path coverage.

Diagrama

Descripción generada automáticamente Texto

Descripción generada automáticamente

1,2,3: *assertTrue*(peo\_00.isValidMove(0, 1, taulell\_auxiliar.MockBoard), "no es mou bé endavant");

1,5: *assertFalse*(peo\_54.isValidMove(6, 4, taulell\_auxiliar.MockBoard), "permet moviment horitzontal");

1,2,6: *assertFalse*(peo\_21.isValidMove(2, 2, taulell\_auxiliar.MockBoard), "permet menjar un altre peça blanca");

4: *assertFalse*(peo\_00.isValidMove(0, -1, taulell\_auxiliar.MockBoard), "permet sortir del tauler");

**Funcionalitat**: Constructor de la subclasse King)

**Localització**: Model/pieces/King.java/King()

**Test**: ModelTests/pieces/KingTest.java/testConstructorKing()

Mateix codi i test que el que hem utilitzat en la subclasse Pawn.

**Funcionalitat**: isValidMove() de la subclasse King. Mateixa idea que en el cas del Peó. No és en aquest mètode on comprovarem si el rei es troba en hake o hake mate.

**Localització**: Model/pieces/King.java/isValidMove()

**Test**: ModelTests/pieces/KingTest.java/testisValidMoveKing(). He utilitzat dos mock objects per simular dos estats del tauler diferents per tal de comprovar si el mètode funciona correctament. Un mock simula una situació específica per a les blanques i l’altre per a les negres.

Diagrama

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1,2,3: *assertTrue*(rei\_24.isValidMove(2, 3, taulell\_aux\_w.MockBoardW), "no deixa mov correcte");

1,2,6: *assertFalse*(rei\_54.isValidMove(5, 3, taulell\_aux\_w.MockBoardW), "permet menjar peo blanc");

1,5: *assertFalse*(rei\_24.isValidMove(2, 6, taulell\_aux\_w.MockBoardW), "permet moure's de 2 en 2");

4: *assertFalse*(rei\_00.isValidMove(-1, -1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");

**Funcionalitat**: Constructor de la subclasse SilverGeneral.

**Localització**: Model/pieces/SilverGeneral.java/SilverGeneral()

**Test**: ModelTests/pieces/SilverGeneralTests.java/testConstructorSG().

**Funcionalitat**: isValidMove() de la subclasse SilverGeneral.

**Localització**: Model/pieces/SilverGeneral.java/isValidMove()

**Test**: ModelTests/pieces/SilverGeneralTests.java/testisValidMoveSG(). Com pot costar una mica més que en els casos dels peons i els reis, aquí demostro que he realitzat els tests per fer statement, path, decision i condition. En l’últim return hi ha dos requadres, ja que simbolitzen 2 camins diferents, quan retorna true i quan retorna false.

Texto

Descripción generada automáticamente

1:

*assertFalse*(sg\_00.isValidMove(1, -1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");  
*assertFalse*(sg\_00.isValidMove(-1, 1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");

*assertTrue*(sg\_24.isValidMove(2, 5, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

2:

*assertFalse*(sg\_24.isValidMove(2, 6, taulell\_aux\_w.MockBoardW), "permet moviments dobles");

*assertTrue*(sg\_24.isValidMove(2, 5, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

*assertFalse(sg\_24.isValidMove(0, 4, taulell\_aux\_w.MockBoardW), "permet moviments dobles");*

3:

*assertFalse*(sg\_24B.isValidMove(1, 4, taulell\_aux\_b.MockBoardB), "permet moviment horitzontal");  
*assertFalse*(sg\_24B.isValidMove(2, 5, taulell\_aux\_b.MockBoardB), "permet moviment cap abaix");

4:

*assertTrue*(sg\_54.isValidMove(6, 5, taulell\_aux\_w.MockBoardW), "no permet menjar");

*assertTrue*(sg\_24.isValidMove(2, 5, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

5:

*assertFalse*(sg\_54.isValidMove(5, 5, taulell\_aux\_w.MockBoardW), "permet menjar blanques");

**Funcionalitat**: Constructor de la subclasse GoldGeneral.

**Localització**: Model/pieces/GoldGeneral.java/GoldGeneral()

**Test**: ModelTests/pieces/GoldGeneralTests.java/testConstructorGG().

**Funcionalitat**: isValidMove() de la subclasse GoldGeneral.

**Localització**: Model/pieces/GoldGeneral.java/isValidMove()

**Test**: ModelTests/pieces/GoldGeneralTests.java/testisValidMoveGG(). Demostració path, statement, decision i condition coverage

Texto

Descripción generada automáticamente

1:

*assertFalse*(gg\_00.isValidMove(1, -1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");  
*assertFalse*(gg\_00.isValidMove(-1, 1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");

*assertTrue*(gg\_24.isValidMove(2, 5, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

2:

*assertTrue*(gg\_24.isValidMove(2, 3, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");  
*assertFalse*(gg\_24.isValidMove(2, 6, taulell\_aux\_w.MockBoardW), "permet moviments dobles");  
*assertFalse*(gg\_24.isValidMove(0, 4, taulell\_aux\_w.MockBoardW), "permet moviments dobles");

3:

*assertTrue*(gg\_24.isValidMove(1, 4, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

*assertFalse*(gg\_24.isValidMove(1, 3, taulell\_aux\_w.MockBoardW), "permet moviment invalid");

*assertTrue*(gg\_24B.isValidMove(1, 3, taulell\_aux\_b.MockBoardB), "no permet moviment correcte");

*assertFalse*(gg\_24B.isValidMove(3, 5, taulell\_aux\_b.MockBoardB), "permet moviment invalid");

4:

*assertFalse*(gg\_24.isValidMove(3, 3, taulell\_aux\_w.MockBoardW), "permet moviment invalid");

*assertTrue*(gg\_24.isValidMove(2, 3, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

*assertTrue*(gg\_24B.isValidMove(1, 3, taulell\_aux\_b.MockBoardB), "no permet moviment correcte");

*assertFalse*(gg\_24B.isValidMove(1, 5, taulell\_aux\_b.MockBoardB), "permet moviment invalid");

5:

*assertTrue*(gg\_54.isValidMove(6, 5, taulell\_aux\_w.MockBoardW), "no permet menjar");  
*assertFalse*(gg\_54.isValidMove(5, 5, taulell\_aux\_w.MockBoardW), "permet menjar blanques");

*assertTrue*(gg\_24.isValidMove(1, 4, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

*assertTrue*(gg\_24B.isValidMove(2, 5, taulell\_aux\_b.MockBoardB), "no permet moviment cap abaix");

*assertTrue*(gg\_54B.isValidMove(6, 3, taulell\_aux\_b.MockBoardB), "no permet menjar");  
*assertFalse*(gg\_54B.isValidMove(5, 5, taulell\_aux\_b.MockBoardB), "permet menjar blanques");

**Funcionalitat**: Constructor de la subclasse knight.

**Localització**: Model/pieces/Knight.java/knight()

**Test**: ModelTests/pieces/KnightTests.java/testConstructorKnight().

**Funcionalitat**: isValidMove() de la subclasse knight.

**Localització**: Model/pieces/Knight.java/isValidMove()

**Test**: ModelTests/pieces/KnightTests.java/testisValidMoveKnight(). Demostració path, statement, decision i condition coverage

Captura de pantalla de computadora

Descripción generada automáticamente

1:

*assertFalse*(knight\_00.isValidMove(1, -1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");  
*assertFalse*(knight\_00.isValidMove(-1, 1, taulell\_aux\_w.MockBoardW), "es mou fora taulell");  
*assertTrue*(knight\_00.isValidMove(1, 2, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

2:

*assertTrue*(knight\_24.isValidMove(3, 6, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

*assertFalse*(knight\_24.isValidMove(1, 2, taulell\_aux\_w.MockBoardW), "permet moures enrere");

*assertFalse*(knight\_24.isValidMove(4, 6, taulell\_aux\_w.MockBoardW), "permet moviments incorrectes");

3:

*assertTrue*(knight\_24.isValidMove(3, 6, taulell\_aux\_w.MockBoardW), "no permet moviment correcte");

*assertTrue*(knight\_54B.isValidMove(4, 2, taulell\_aux\_b.MockBoardB), "no permet menjar");  
*assertFalse*(knight\_54B.isValidMove(6, 2, taulell\_aux\_b.MockBoardB), "permet menjar negres");

**Funcionalitat**: Constructor de la subclasse bishop.

**Localització**: Model/pieces/Bishop.java/bishop()

**Test**: ModelTests/pieces/BishopTests.java/testConstructorBishop().

**Funcionalitat**: isValidMove() de la subclasse Bishop.

**Localització**: Model/pieces/Bishop.java/isValidMove()

**Test**: ModelTests/pieces/BishopTests.java/testisValidMoveBishop(). Demostració Loop testing

Texto

Descripción generada automáticamente

Casos de prova: dx[1,2,3,5,7,8]

*assertTrue*(bishop\_88B.isValidMove(7, 7, taulell\_aux.MockBoard), "no permet mov valid");  
*assertTrue*(bishop\_88B.isValidMove(6, 6, taulell\_aux.MockBoard), "no permet mov valid");  
*assertTrue*(bishop\_88B.isValidMove(3, 3, taulell\_aux.MockBoard), "no permet mov valid");  
*assertTrue*(bishop\_88B.isValidMove(1, 1, taulell\_aux.MockBoard), "no permet mov valid");  
*assertTrue*(bishop\_88B.isValidMove(0, 0, taulell\_aux.MockBoard), "no permet mov valid");

**Funcionalitat**: Constructor de la subclasse lance.

**Localització**: Model/pieces/Lance.java/lance()

**Test**: ModelTests/pieces/LanceTests.java/testConstructorLance().

**Funcionalitat**: isValidMove() de la subclasse Lance.

**Localització**: Model/pieces/Lance.java/isValidMove()

**Test**: ModelTests/pieces/LanceTests.java/testisValidMoveLance(). Demostració Loop testing

Texto

Descripción generada automáticamente

Casos de prova: newY[1,2,3,5,7,8 i 7,6,5,3,1,0]

*assertTrue*(lance\_00.isValidMove(0, 1, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_00.isValidMove(0, 2, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_00.isValidMove(0, 3, taulell\_aux.MockBoard), "no permet moviment correcte");*assertTrue*(lance\_00.isValidMove(0, 5, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_00.isValidMove(0, 7, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_00.isValidMove(0, 8, taulell\_aux.MockBoard), "no permet moviment correcte");  
  
Piece lance\_88B = taulell\_aux.getPiece(8,8);  
*assertTrue*(lance\_88B.isValidMove(8, 7, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_88B.isValidMove(8, 6, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_88B.isValidMove(8, 5, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_88B.isValidMove(8, 3, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_88B.isValidMove(8, 1, taulell\_aux.MockBoard), "no permet moviment correcte");  
*assertTrue*(lance\_88B.isValidMove(8, 0, taulell\_aux.MockBoard), "no permet moviment correcte");

**Funcionalitat**: Constructor de la subclasse rook.

**Localització**: Model/pieces/Rook.java/Rook()

**Test**: ModelTests/pieces/RookTests.java/testConstructorRook().

**Funcionalitat**: isValidMove() de la subclasse Rook.

**Localització**: Model/pieces/Rook.java/isValidMove()

**Test**: ModelTests/pieces/RookTests.java/testisValidMoveRook().

**Funcionalitat**: Constructor de la classe Board. Bàsicament el que fa és col·locar les peces a la posició inicial.

**Localització**: Model/Board.java/Board()

**Test**: ModelTests/BoardTests.java/testconstructorBoard().

**Funcionalitat**: InitializeBoard() de la classe Board. Es el mètode que crida el constructor per col·locar les peces a la posició inicial.

**Localització**: Model/Board.java/Board()

**Test**: ModelTests/BoardTests.java/testconstructorBoard().

**Funcionalitat**:movePiece() de la classe Board. Métode que mou les peces de lloc en el tauler i actualitza les peces.

**Localització**: Model/Board.java/movePiece()

**Test**: ModelTests/BoardTests.java/testmovePiece().

**Funcionalitat**: displayBoard() de la classe Board. Aquest mètode serveix per imprimir per pantalla la situació actual del tauler.

**Localització**: Model/pieces/Board.java/displayBoard()

**Test**: ModelTests /BoardTests.java/testdisplayBoard ().

**Funcionalitat**: findKing() de la classe Board. Aquest mètode detecta si tenim els 2 reis en el tauler.

**Localització**: Model/Board.java/Board()

**Test**: ModelTests/BoardTests.java/testfindKing().

**Funcionalitat**:iskingincheck() de la classe Board. Mètode que detecta si un rei es troba en hake.

**Localització**: Model/Board.java/iskingincheck()

**Test**: ModelTests/BoardTests.java/testiskingincheck).

**Funcionalitat**:iskingincheckmate() de la classe Board. Mètode que detecta si el rei es troba en hake mate.

**Localització**: Model/Board.java/iskingincheckmate()

**Test**: ModelTests/BoardTests.java/testiskingincheckmate().

**Funcionalitat**:startGame() de la classe GameCOntroller. Aquest mètode es el que fa córrer la partida. Inicialitza el tauler, va alternant el torn dels jugadors i també va comprovant que els moviments introduïts per teclat dels jugadors son moviments vàlids. Si els reis es troben en hake o en hake mate i si el joc ha acabat finalitza la partida. Per comprovar aquest mètode hem utilitzat mockito per simular diferents estats de la partida.

**Localització**: Controller/GameController.java/startGame())

**Test**:

GameControllerTests/GameCOntrollerTests.java/testpieceisnull().

GameControllerTests/GameCOntrollerTests.java/testinvalidmove ().

GameControllerTests/GameCOntrollerTests.java/testkingincheck().

GameControllerTests/GameCOntrollerTests.java/ testkingincheckmate().

GameControllerTests/GameCOntrollerTests.java/ testvalidmove().

**Funcionalitat**:startGame() de la classe GameCOntroller. Aquest mètode es el que fa córrer la partida. Inicialitza el tauler, va alternant el torn dels jugadors i també va comprovant que els moviments introduïts per teclat dels jugadors son moviments vàlids. Si els reis es troben en hake o en hake mate i si el joc ha acabat finalitza la partida. Per comprovar aquest mètode hem utilitzat mockito per simular diferents estats de la partida.

**Localització**: Controller/GameController.java/startGame())

**Test**:

GameControllerTests/GameCOntrollerTests.java/testpieceisnull().

GameControllerTests/GameCOntrollerTests.java/testinvalidmove ().

GameControllerTests/GameCOntrollerTests.java/testkingincheck().

GameControllerTests/GameCOntrollerTests.java/ testkingincheckmate().

GameControllerTests/GameCOntrollerTests.java/ testvalidmove().

**Funcionalitat**:main() de la classe view. Aquest mètode crida a gamecontroller.startgame(). Com la vista és molt simple ja que només imprimeix els moviments per pantalla i interacciona amb els jugadors a partir de lo que introdueixen per teclat només te aquest mètode. Els tests que he realitzat també utilitzaven mockito per simular moments de la partida.

**Localització**: View/Main.java/main(string artgs[])

**Test**:

ViewTests/MainTests.java/testpieceisnull().

ViewTests/MainTests.java/testinvalidinputhandling().

ViewTests/MainTests.java/testinteractionwithcontroller().