Álex Velasco Cañete de Cárdenas (1457468)
Sergio Manchón Romero (1424310)

ESCACS

En aquest informe s'explicaran les funcionalitats implementades i els test que s'han passat.

No s'inclouen les funcionalitats simples com setters ni getters o altres mètodes que no poden ser testejats excepte el set i get de les caselles que si permet fer caixa negra.

Tampoc afegirem al informe aquells mètodes que donin informació del joc pintada (imprimir el tauler o les instruccions).

Repositori:

https://github.com/131alexx/chessGameTQS

Sintesi.

Decision Coverage: getWinner (Class: Game), getPossibleMoves (Class: King)

Condition Coverage: getWinner (Class: Game), getPossibleMoves (Class: King)

Path Coverage: getPossibleMoves (Class: King)

Loop testing: getPossibleMoves (Class: Rook), getPossibleMoves (Class: Bishop)

Mock objects: MockBoard (Class: Board), MockGame (Class: Game)

Per a tots els mètodes: Statement Coverage

Per a mètodes de baix nivell: Caixa Negra (Particions equivalents + Valors límits/frontera)

1. Code Coverage

Abans d'anar funcionalitat per funcionalitat, farem un statement coverage per veure que s'han executat totes les línies de codi que poden ser testejades.

chessGameTQS	99,4 %	10.449	68	10.517
src	98,2 %	2.858	52	2.910
chessGameTQS	98,2 %	2.858	52	2.910
Main.java	0,0 %	0	34	34
Game.java	95,1 %	350	18	368
Game	95,1 %	350	18	368
Bishop.java	100,0 %	223	0	223
Board.java	100,0 %	677	0	677
King.java	100,0 %	302	0	302
Knight.java	100,0 %	327	0	327
Pawn.java	100,0 %	259	0	259
Piece.java	100,0 %	26	0	26
Queen.java	100,0 %	385	0	385
Rook.java	100,0 %	201	0	201
Square.java	100,0 %	108	0	108
test	99,8 %	7.591	16	7.607

La classe Main no ha sigut testejada, ja que tots el mètodes pertanyen a la classe Game i ja han sigut testejats.

De la classe game tenim un 95%, el que no s'executa aquí es la funció getInput que s'encarrega de demanar una entrada a l'usuari i per tant no pot ser testejada:

Álex Velasco Cañete de Cárdenas (1457468)
Sergio Manchón Romero (1424310)

```
public char getInput() {
    Scanner scan = new Scanner(System.in);
    String key = scan.next();
    if (key.length() > 1)
        return 'e';
    return key.charAt(0);
}
```

Per testejar això s'ha creat un MockObject on la funció getInput és substituïda per una cadena de caràcter que simula un torn del joc.

1. Funcionalitats i testeig

Les següents funcionalitats corresponen a la classe BOARD

1.1 Canvi de torn de Jugador.

L'objectiu és que al final de cada torn la variable que defineix el torn del jugador s'actualitzi.

Localització:

Mètode: swapTurn Directori: src/Board.Java **Classe: Board**

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Torn del jugador 1/ jugador 2

1.2 Inicialitzar taulell

Inicialització de les peces dels dos Jugadors. Es col·loquen les peces en les peses inicials quan s'inicia la partida

Localització:

Mètode: resetBoard Directori: src/Board.Java Classe: Board

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

1.3 Retornar la casella

Retorna la casella segons les posicions indicades. Aquestes posicions s'han de trobar entre 0 i 7

Localització:

Mètode: getSquare Directori: src/Board.Java Classe: Board

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Dins/fora del taulell., frontera del taulell.

1.4 Moure la peça

Aquesta funcionalitat recull les dues caselles involucrades, la casella d'origen i la casella destí. Si permet moure la peça retornarà un true a més de fer el moviment. Si no la pot moure, retornarà un false.

Localització:

Mètode: movePiece Directori: src/Board.Java Classe: Board

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Moviment possible/impossible.

Álex Velasco Cañete de Cárdenas (1457468)
Sergio Manchón Romero (1424310)

1.5 Actualitzar la casella

Actualitza la casella amb la peça indicada i la posició on es trobarà. Aquestes posicions s'han de trobar entre 0 i 7

Localització:

Mètode: setSquare Directori: src/Board.Java Classe: Board

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Actualitzar a una ubicació possible/impossible

1.6 Moure el cursor

Aquestes funcionalitats permetran moure el cursor del tauler per tal de que l'usuari pugui veure on es troba el cursor.

Localització:

Mètodes: `cursorDown, cursorUp, cursorLeft, cursorRight`

Directori: src/Board.Java Classe: Board

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Possible/Impossible moviment del cursor. (Dins/fora taulell)

La següent funcionalitat correspon a la classe SQUARE

1.7 Comparació de casella

Compara que dues caselles tinguin la mateixa peça i es trobin a la mateixa posició

Localització:

Mètode: equals

Directori: src/Square.Java Classe: Square

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Mateixa/diferent/sense peça i ubicacions iguals/diferents

Les següents funcionalitats corresponen a la classe GAME

1.8 Jaque

Funcionalitat que comprova si en el tauler hi ha un JAQUE

Localització:

Mètode: isJaque

Directori: src/Game.Java Classe: Game

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Jaque del jugador 1, del jugador 2 i sense jaque.

MOCK OBJECT

Per testejar aquest mètode s'ha utilitzat un mock object de la clase Board que prepara un taulell on ja hi ha un jaque.

1.9 Joc Terminat

Funcionalitat que comprova si el joc ha terminat. Mira que els dos reis es trobin sobre el taulell.

Localització:

Mètode: isFinished

Directori: src/Game.Java Classe: Game

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Joc finalitzat/sense finalitzar.

1.10 Aconseguir guanyador

Un cop el mètode Joc Terminat es true, s'executa aquesta funcionalitat per veure qui és el guanyador(qui tingui el rei al taulell)

Localització:

Mètode: getWinner

Directori: src/Game.Java Classe: Game

Testing:

TEST CAIXA BLANCA:

- Statement Coverage
- Condition Coverage
- Decision Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Guanya jugador 1 i guanya jugador 2.

1.11 Jugar un torn

Aquesta funcionalitat permet als jugador fer un torn, no termina fins que el jugador mou una peça.

Localització:

Mètode: playTurn

Directori: src/Board.java Classe: Board

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Canviar en torn 1 / 2, jugades possibles / impossibles.

1.12 Recollir entrada d'usuari

Aquesta funcionalitat permet recollir l'entrada del usuari, i si es una entrada vàlida, efectuar el moviment del cursor.

Localització:

Mètode: getInput

Directori: src/Game.Java Classe: Game

Testing:

MOCK OBJECT:

Per fer el testing s'ha utilitzat un MockObject de la classe Game modificant el mètode getInput per simular l'entrada d'usuari

Directori MockObject: test/MockGame

El mètode que s'ha modificat respecte a la classe GAME es el getInput de manera que quan es cridi en el test s'executi el mètode implementat en el MockObject.

El que fa aquest mètode es retornar una cadena fixa de chars que permetrà al test moure el cursor segons la cadena rebuda.

Les següents funcionalitats corresponen a la classe PIECE i a les seves classes heretades (Totes les peces diferents)

1.13 Comparar peça

Compara que dues peces siguin iguals (mateix jugador i tipus peça)

Localització:

Mètode: equals

Directori: src/Piece.Java Classe: Piece

Testing:

TEST CAIXA BLANCA:

- Statement Coverage

TEST CAIXA NEGRA:

- Particions equivalents: Mateixa/diferent peça i mateix/diferent jugador.

1.14 Aconseguir els moviments possibles que pot fer una peça

Aquesta funcionalitat retorna una llista de moviments possibles que pot fer la peça segons la seva posició.

Localització:

Mètode: getPossibleMoves()

Directori: src/Piece.Java Classes: Piece (Mètode abstracte)

src/Bishop.Java Classes: Bishop

src/Pawn.Java Classes: Pawn

src/Rook.Java Classes: Rook

src/Queen.Java Classes: Queen

src/Knight.Java Classes: Knight

src/King.Java Classes: King

Testing:

TEST CAIXA NEGRA:

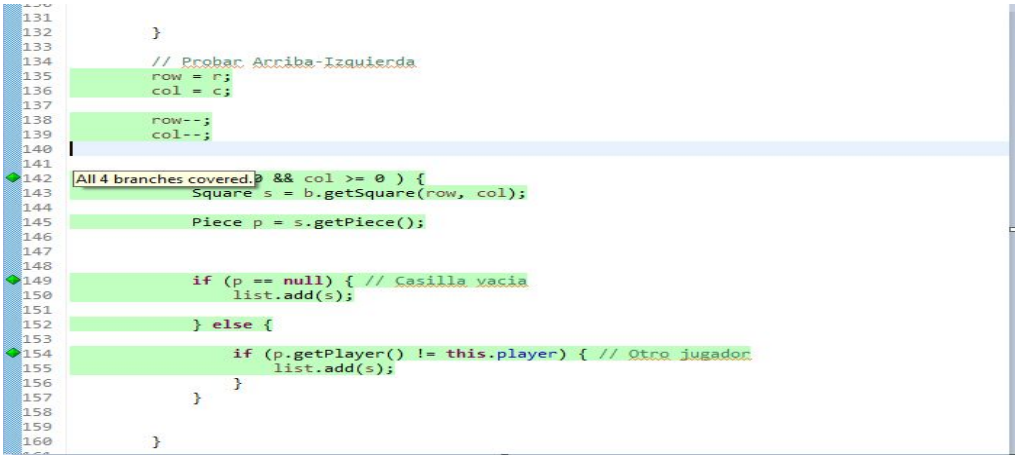
- Particions equivalents: obtenir les posicions quan la peça es troba als límits del taulell, tant dins com fora.

TEST CAIXA BLANCA:

- *Statement Coverage: Totes les classes*
- *Decision Coverage, Condition Coverage:* ***CLASS KING***

  També s'ha realitzat el decision coverage per al mètode getPossiblesMoves de la peça **Rei**. Adjuntem una imatge fent el code Coverage, que a més de veure que les línies de codi han sigut executades, per cada condiciones et diu si totes les condicions han sigut provades o no:

Álex Velasco Cañete de Cárdenas (1457468)
Sergio Manchón Romero (1424310)



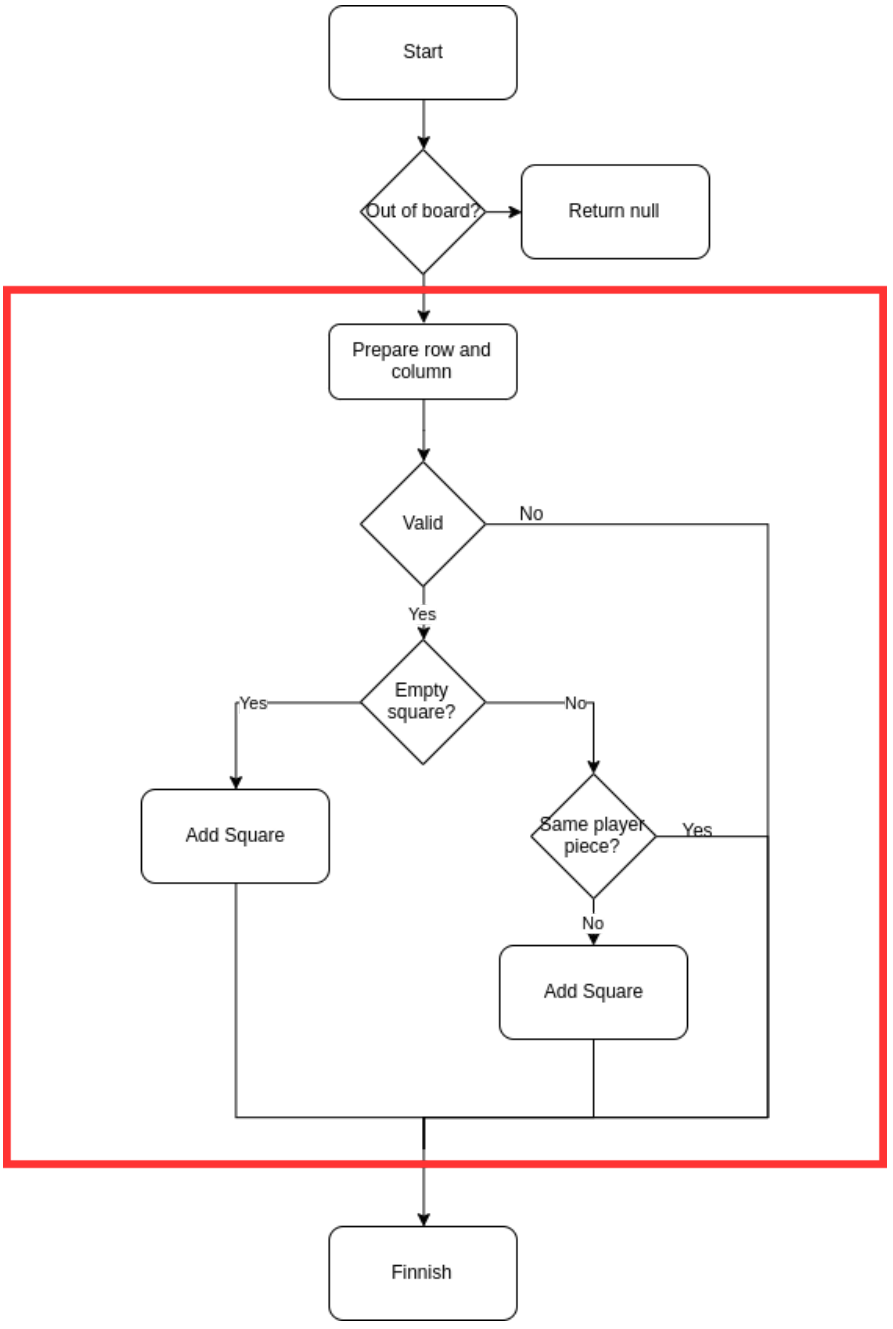

```
131
132         }
133
134         // Probar Arriba-Izquierda
135         row = r;
136         col = c;
137
138         row--;
139         col--;
140
141
142         while (row >= 0 && col >= 0 ) {
143             Square s = b.getSquare(row, col);
144
145             Piece p = s.getPiece();
146
147
148
149             if (p == null) { // Casilla vacia
150                 list.add(s);
151
152             } else {
153
154                 if (p.getPlayer() != this.player) { // Otro jugador
155                     list.add(s);
156                 }
157             }
158
159
160         }
```

No afegim tot el codi ja que hi ha molts condicionals però com podem veure totes les línies surten en verd. Si no s'hagués executat el condition Coverage les condicions sortirien en color groc. Si ens fixem en l'etiqueta de la línia 142, Et diu que per aquell IF les quatre condicions han sigut cobertes

- *Path Coverage:* ***CLASS KING***

  S'ha fet Path Coverage del mètode getPossibleMovements del Rei. A la següent imatge inserim el diagrama de flux:

Álex Velasco Cañete de Cárdenas (1457468)
Sergio Manchón Romero (1424310)



Cal tenir en compte que aquest diagrama de flux només correspòn a una direcció del Rey. Si volguessim fer les 8 direccions hauriem de conectar de manera seqüencial 8 vegades el que está en el quadrat roig (No ho fem per qüestions d'espai).

El nombre total de Paths que trobem al mètode es de 33 camins diferents.

- *Loop testing:* ***CLASS PIECE***

  S'ha realitzat loop testing de la clase Piece (S'han fet set servir La torre i el Alfil que tenen bucles dins del seu getPossiblesMoves.