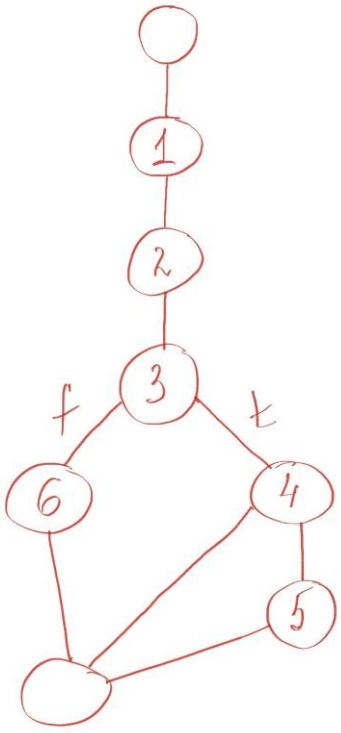
Practica 7

El grafo de posibles caminos del método canMove de la clase Game:

```
168         * @return
169         */
170        public boolean canMove(int startrow, int startcol, int endrow, int endcol) {
171            boolean vuelta = false;
172            Piece p = this.board.getCell( row: startrow,  col: startcol).getPiece();
173
174            if (p != null && p.getType() == this.turn) {
175                vuelta = p.canmove( board: this.board,  row: endrow,  col: endcol);
176            }
177            return vuelta;
178        }
```



Caja blanca:

El método canMove:

Entrada	Salida
startrow = 6 startcol = 0 endrow = 5 Endcol = 0 this.board = inicial this.turn = White	true
startrow = 7 startcol = 0 endrow = 6 Endcol = 0 this.board = inicial this.turn = White	false
startrow = 2 startcol = 0 endrow = 3 Endcol = 0 this.board = inicial this.turn = White	false
startrow = 1 startcol = 0 endrow = 2 Endcol = 0 this.board = inicial this.turn = White	false
startrow = 6 startcol = 0 endrow = 3 Endcol = 0 this.board = inicial this.turn = White	false

Caja negra:

Método getCell de la clase Board:

Entrada, clases de equivalencia:

Codigo	Row
e1a	Valor menores que 0
e1b	Valor mayores que 7
e1c	Valor entre 0 y 7

Codigo	Col
e2a	Valor menores que 0
e2b	Valor mayores que 7
e2c	Valor entre 0 y 7

Salida, clases de equivalencia:

Codigo	Salida
sa	Cell
sb	Null

Valores frontera o límite:

Codigo	Row
l1a	-1
l1b	0
l1c	7
l1d	8

Codigo	Col
l2a	-1
l2b	0
l2c	7
l2d	8

Casos de prueba: clases de equivalencia

Entrada	Salida	Clases de equivalencia	Descripción
row=-3, col=0	Null	e1a, sb	Valor menor que 0
row=10, col=0	Null	e1b, sb	Valor mayor que 7
row=1, col=0	Cell	e1c, sa	Valores entre 0 y 7
row=7, col=2	Cell	e2c, sb	Valores entre 0 y 7
row=0, col=-3	Null	e2a, sb	Valor menor que 0
row=0, col=10	Null	e2b, sb	Valor mayor que 7

Casos de prueba: valores límite

Entrada	salida	Valor límite	Descripción
row=-1, col=0	Null	l1a	Valor límite inválido a la izquierda
row=0, col=0	Cell	l1b	Valor límite válido a la izquierda
row=7, col=0	Cell	l1c	Valor límite válido a la derecha
row=8, col=0	Null	l1d	Valor límite inválido a la derecha

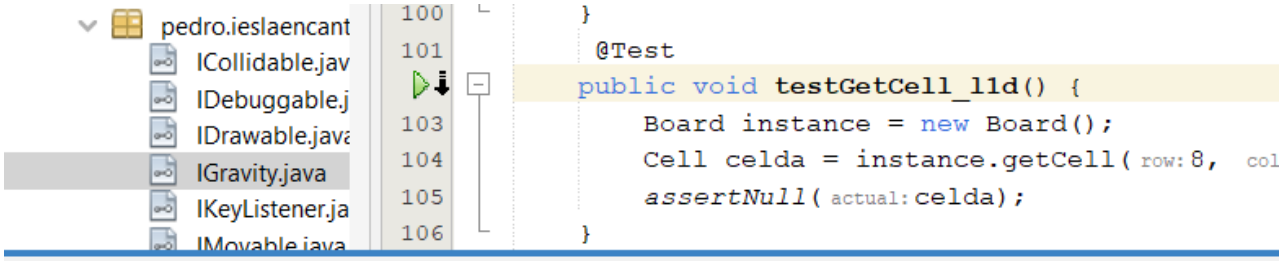
row=0, col=-1	Null	I2a	Valor límite inválido a la izquierda
row=2, col=0	Cell	I2b	Valor límite válido a la izquierda
row=0, col=7	Cell	I2c	Valor límite válido a la derecha
row=0, col=10	Null	I2d	Valor límite inválido a la derecha

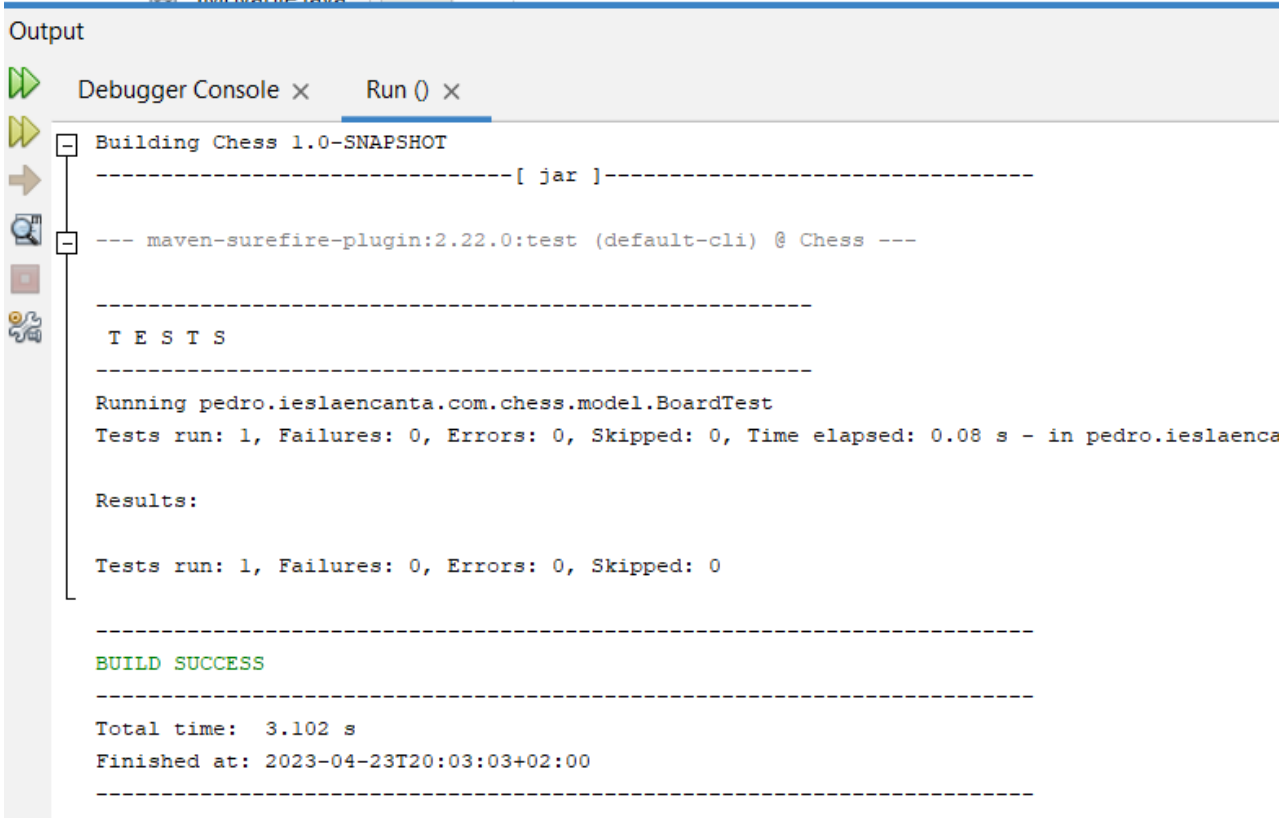
Resumen: En metodo GetCell (assertNull(celda) no podemos hacer test porque tenemos error "ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 8", que significa que este metodo no vuelve null en caso cuando tenemos row o col menor que 0 y mayor que 7.

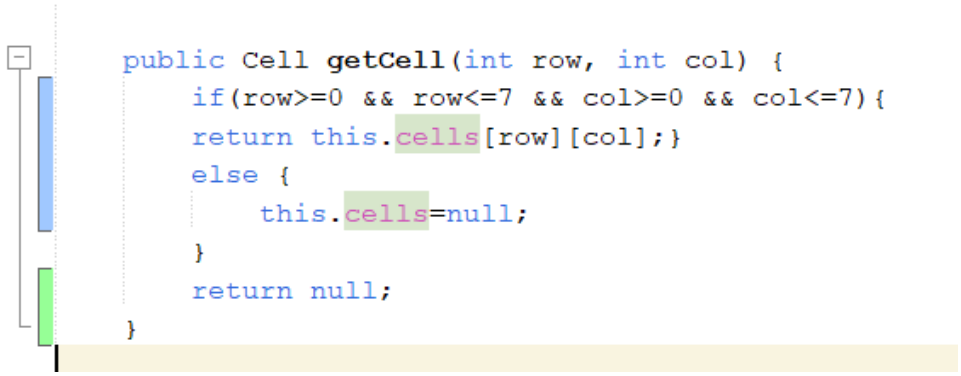
Es método caja negra y no podemos cambiar codigo, pero si cambiamos – todo funciona bien.

Si añadimos este comprueba en metodo GetCell y vuelve row y col =nul en caso que menor que 0 y mayor que 7=>

```
public Cell getCell(int row, int col) {
    if(row>=0 && row<=7 && col>=0 && col<=7){
    return this.cells[row][col];}
    else {
        this.cells=null;
    }
    return null;
}
```







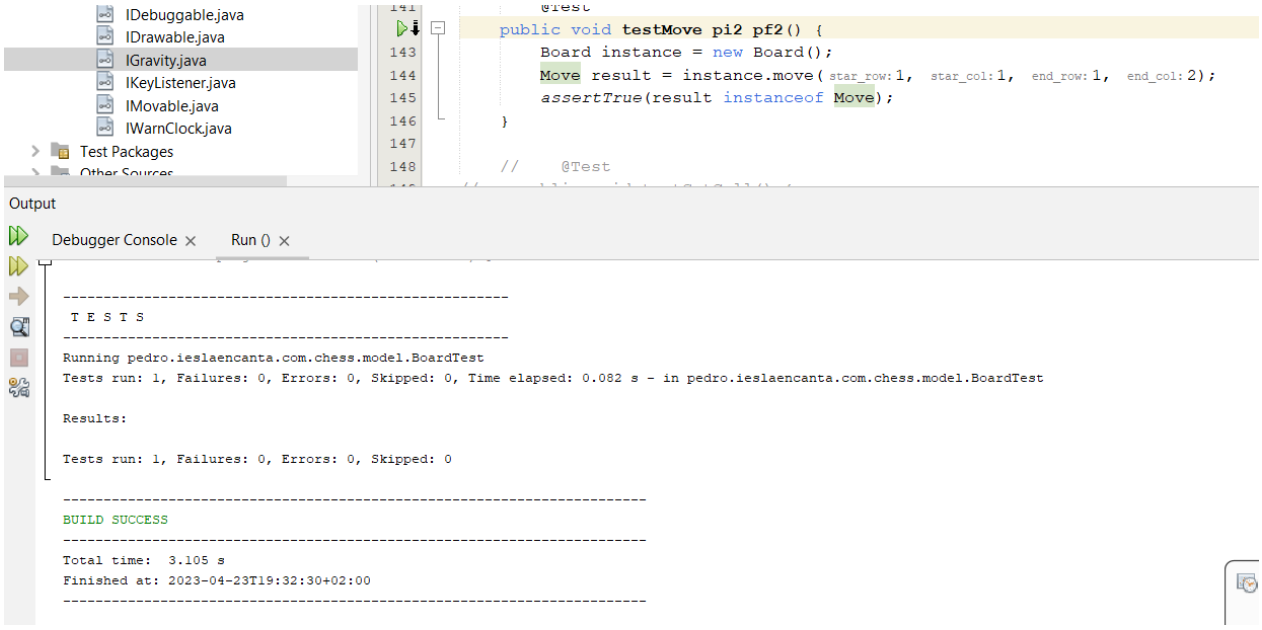
En caso de assertTrue(celda instanceof Cell); todo funciona bien:

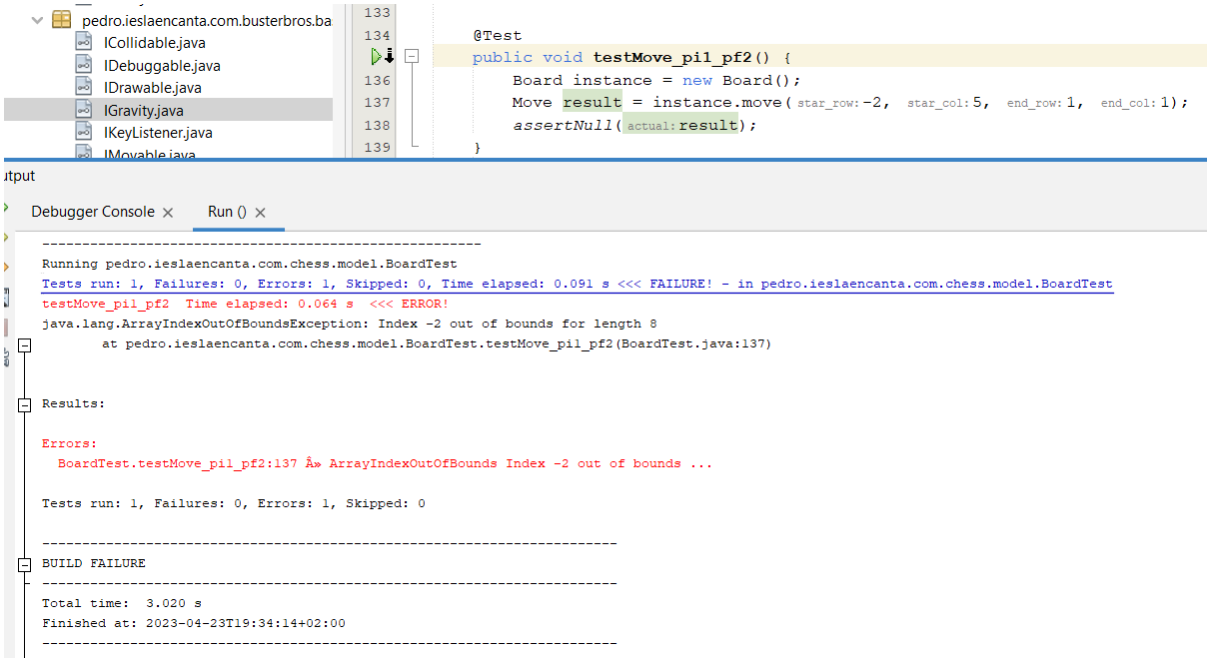
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running pedro.ieslaencanta.com.chess.model.BoardTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.089 s - in pedro.ieslaencant

Results:

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.229 s
Finished at: 2023-04-23T19:15:36+02:00
------------------------------------------------------------------------
```

Método move de la clase Board:

En este método como en el método GetCell:

Todo funciona donde salida-Move, y no funciona donde salida Null.





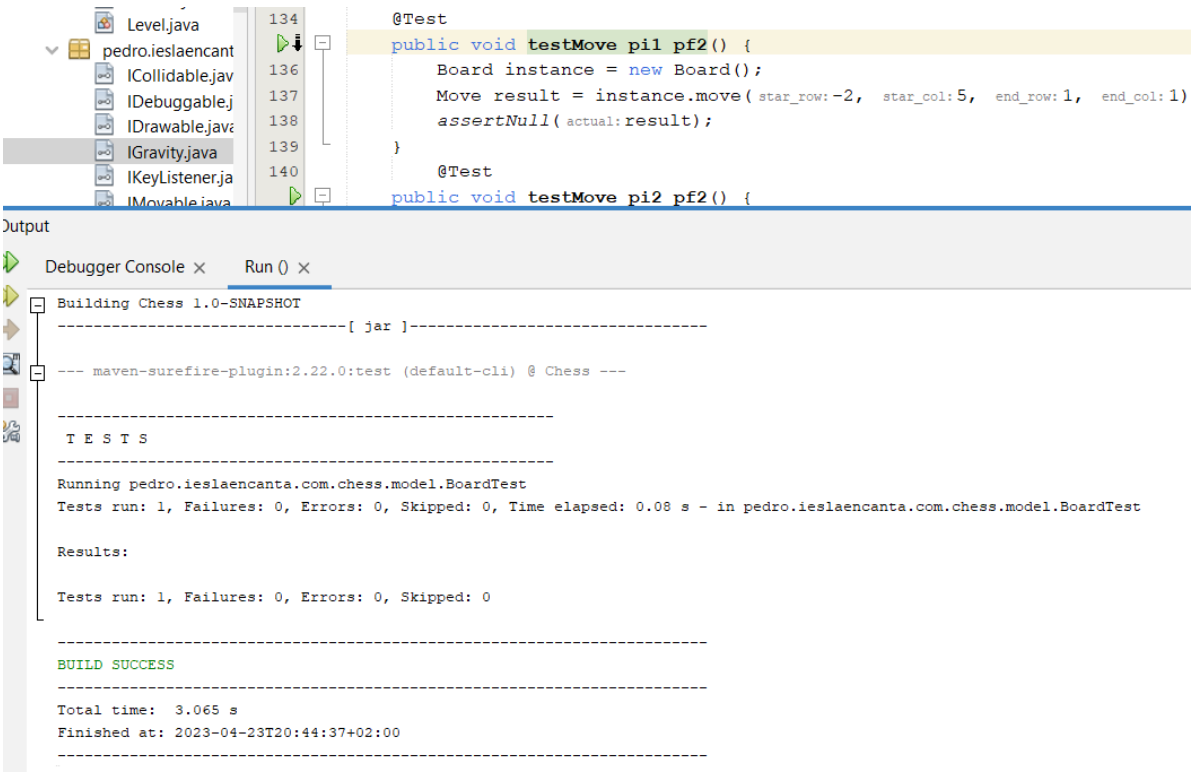
Es método caja negra y no podemos cambiar codigo, pero si cambiamos – todo funciona bien.

Si cambiamos método move- todo funciona bien.

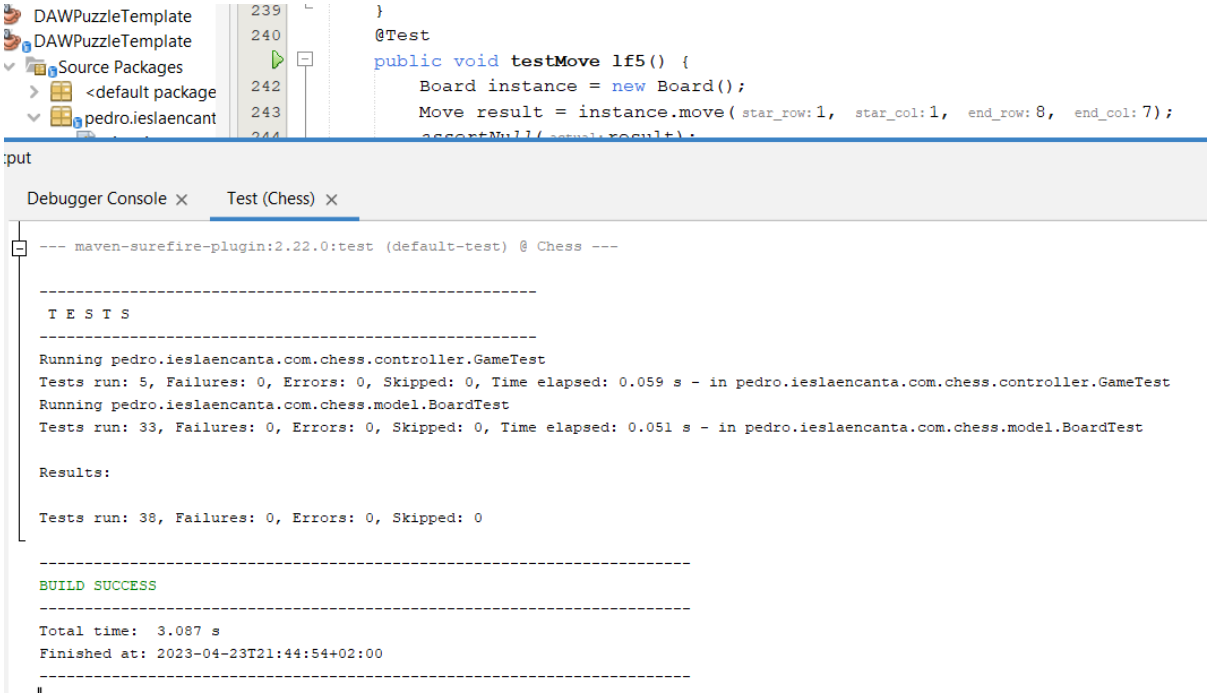
```
public Move move(int star_row, int star_col, int end_row, int end_col) {
    if(star_row>=0 && star_row<=7 && star_col>=0 && star_col<=7 && end_row>=0 && end_row<=7 && end_col>=0 && end_col<=7){
        Piece p = this.cells[star_row][star_col].getPiece();
        Move m = null;
        if (p != null) {
            m = p.move(this, end_row, end_col);
            this.cells[star_row][star_col].setPiece(null);
            this.cells[end_row][end_col].setPiece(p);
        }
        return m;
    }
```

```
    else {
        this.cells=null;
    }
    return null;
  }
```

```
public Move move(int star_row, int star_col, int end_row, int end_col) {
    if(star_row>=0 && star_row<=7 && star_col>=0 && star_col<=7 && end_row>=0 && end_row<=7 && end_col>=0 && end_col<=7){

    Piece p = this.cells[star_row][star_col].getPiece();

    Move m = null;
    if (p != null) {
        m = p.move( board: this,  row: end_row,  col: end_col);
        this.cells[star_row][star_col].setPiece( piece: null);
        this.cells[end_row][end_col].setPiece( piece: p);
    }
    return m;
     }
    else {
        this.cells=null;
    }
    return null;
}
```



Todos tests funciona bien.



Método reset de la clase Board:

Entrada, clases de equivalencia:

Codigo	Cell/piecetype
e1a	Figura de color negro (black)
e1b	Figura de color blanco (white)
e1c	Null
e1d	Posición entre 0 y 7

Salida, clases de equivalencia:

Codigo	Salida
sa	Rook
sb	Knight
sc	Bishop
sd	King
se	Queen
sf	Pawn

sg	Null

Codigo	Salida
sh	Black
si	White

Casos de prueba: clases de equivalencia

Entrada	Salida	Clases de equivalencia	Descripción
Rook Black [0,0]	Rook Black[0,0]	e1a, e1d, sa, sh	Rook Black [0,0]
Rook Black [0,7]	Rook Black [0,7]	e1a, e1d, sa, sh	Rook Black [0,7]
Rook white [7,0]	Rook white [7,0]	e1b, e1d, sa, si	Rook white [7,0]
Rook white [7,7]	Rook white [7,7]	e1b, e1d, sa, si	Rook white [7,7]
Knight Black [0,1]	Knight Black [0,1]	e1a, e1d, sb, sh	Knight Black [0,1]
Knight Black [0,6]	Knight Black [0,6]	e1a, e1d, sb, sh	Knight Black [0,6]
Knight white [7,1]	Knight white [7,1]	e1b, e1d, sb, si	Knight white [7,1]
Knight white [7,6]	Knight white [7,6]	e1b, e1d, sb, si	Knight white [7,6]

Bishop Black [0,2]	Bishop Black [0,2]	e1a, e1d, sc, sh	Bishop Black [0,2]
Bishop Black [0,5]	Bishop Black [0,5]	e1a, e1d, sc, sh	Bishop Black [0,5]
Bishop white [7,2]	Bishop white [7,2]	e1b, e1d, sc, si	Bishop white [7,2]
Bishop white [7,5]	Bishop white [7,5]	e1b, e1d, sc, si	Bishop white [7,5]
King Black [0,3]	King Black [0,3]	e1a, e1d, sd, sh	King Black [0,3]
King white [7,3]	King white [7,3]	e1b, e1d, sd, si	King white [7,3]
Queen Black [0,4]	Queen Black [0,4]	e1a, e1d, sd, sh	Queen Black [0,4]
Queen white [7,4]	Queen white [7,4]	e1b, e1d, sd, si	Queen white [7,4]
Pawn Black [1,7]	Pawn Black [1,7]	e1a, e1d, se, sh	Pawn Black [1,7]
Pawn Black [1,5]	Pawn Black [1,5]	e1a, e1d, se, sh	Pawn Black [1,5]
Pawn white [6,6]	Pawn white [6,6]	e1b, e1d, se, si	Pawn white [6,6]
Pawn white [6,2]	Pawn white [6,2]	e1b, e1d, se, si	Pawn white [6,2]
Null [2,0]	Null	e1c, e1d, sg	null

Hay errores en este codigo:

```
Running pedro.ieslaencanta.com.chess.model.BoardTest
Tests run: 21, Failures: 8, Errors: 0, Skipped: 0, Time elapsed: 0.05 s <<< FAILURE! - in pedro.ieslaencanta.com.c
testReset_Rook_black_00  Time elapsed: 0.005 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Rook_black_00(BoardTest.java:266)

testReset_Rook_black_07  Time elapsed: 0.002 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Rook_black_07(BoardTest.java:274)

testReset_Knight_white_71  Time elapsed: 0.002 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Knight_white_71(BoardTest.java:314)

testReset_Knight_white_76  Time elapsed: 0.005 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Knight_white_76(BoardTest.java:322)

testReset_Rook_white_70  Time elapsed: 0 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Rook_white_70(BoardTest.java:282)

testReset_Rook_white_77  Time elapsed: 0.003 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Rook_white_77(BoardTest.java:290)

testReset_Knight_black_01  Time elapsed: 0.001 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Knight_black_01(BoardTest.java:298)

testReset_Knight_black_06  Time elapsed: 0.001 s  <<< FAILURE!
| org.opentest4j.AssertionFailedError: expected: <true> but was: <false>
        at pedro.ieslaencanta.com.chess.model.BoardTest.testReset_Knight_black_06(BoardTest.java:306)
```

En Cell [0,0] y [0,6] para Black, [7,0] y [7,6] para White esta Rook, Cell [0,1] y [0,7] Black, [7,1] y [7,7] para White esta Knight.

```
    }
    this.cells[0][0].setPiece(new Knight(0, 1, PieceType.Black, true,"1"));
    this.cells[0][1].setPiece(new Rook(0, 0, PieceType.Black, true,"1"));
    this.cells[0][2].setPiece(new Bishop(0, 2, PieceType.Black, true,"1"));
    this.cells[0][3].setPiece(new Queen(0, 3, PieceType.Black, true,"1"));
    this.kingBlack=new King(0, 4, PieceType.Black, true,"1");
    this.cells[0][4].setPiece(this.kingBlack);
    this.cells[0][5].setPiece(new Bishop(0, 5, PieceType.Black, true,"2"));
    this.cells[0][6].setPiece(new Rook(0, 7, PieceType.Black, true,"2"));
    this.cells[0][7].setPiece(new Knight(0, 6, PieceType.Black, true,"2"));

    this.cells[7][0].setPiece(new Knight(7, 1, PieceType.White, true,"1"));
    this.cells[7][1].setPiece(new Rook(7, 0, PieceType.White, true,"1"));
    this.cells[7][2].setPiece(new Bishop(7, 2, PieceType.White, true,"1"));
    this.cells[7][3].setPiece(new Queen(7, 3, PieceType.White, true,"1"));
    this.kingWhite= new King(7, 4, PieceType.White, true,"1");
    this.cells[7][4].setPiece(this.getKingWhite());
    this.cells[7][5].setPiece(new Bishop(7, 5, PieceType.White, true,"2"));
    this.cells[7][6].setPiece(new Rook(7, 7, PieceType.White, true,"2"));
    this.cells[7][7].setPiece(new Knight(70, 6, PieceType.White, true,"2"));
    for (int i = 0; i < this.cells[1].length ; i++) {
        this.cells[1][i].setPiece(new Pawn(1, i, PieceType.Black, true,String.valueOf(i)));
        this.cells[6][i].setPiece(new Pawn(6, i, PieceType.White, true,String.valueOf(i)));
    }
}
```

Necesito cambiar código:

```
this.cells[0][1].setPiece(new Knight( row: 0,  col: 1,  type: PieceType.Black,  alive: true, id: "1"));
this.cells[0][0].setPiece(new Rook( row: 0,  col: 0,  type: PieceType.Black,  alive: true, id: "1"));
this.cells[0][2].setPiece(new Bishop( row: 0,  col: 2,  type: PieceType.Black,  alive: true, id: "1"));
this.cells[0][3].setPiece(new Queen( row: 0,  col: 3,  type: PieceType.Black,  alive: true, id: "1"));
this.kingBlack=new King( row: 0,  col: 4,  type: PieceType.Black,  alive: true, id: "1");
this.cells[0][4].setPiece( piece: this.kingBlack);
this.cells[0][5].setPiece(new Bishop( row: 0,  col: 5,  type: PieceType.Black,  alive: true, id: "2"));
this.cells[0][7].setPiece(new Rook( row: 0,  col: 7,  type: PieceType.Black,  alive: true, id: "2"));
this.cells[0][6].setPiece(new Knight( row: 0,  col: 6,  type: PieceType.Black,  alive: true, id: "2"));

this.cells[7][1].setPiece(new Knight( row: 7,  col: 1,  type: PieceType.White,  alive: true, id: "1"));
this.cells[7][0].setPiece(new Rook( row: 7,  col: 0,  type: PieceType.White,  alive: true, id: "1"));
this.cells[7][2].setPiece(new Bishop( row: 7,  col: 2,  type: PieceType.White,  alive: true, id: "1"));
this.cells[7][3].setPiece(new Queen( row: 7,  col: 3,  type: PieceType.White,  alive: true, id: "1"));
this.kingWhite= new King( row: 7,  col: 4,  type: PieceType.White,  alive: true, id: "1");
this.cells[7][4].setPiece( piece: this.getKingWhite());
this.cells[7][5].setPiece(new Bishop( row: 7,  col: 5,  type: PieceType.White,  alive: true, id: "2"));
this.cells[7][7].setPiece(new Rook( row: 7,  col: 7,  type: PieceType.White,  alive: true, id: "2"));
this.cells[7][6].setPiece(new Knight( row: 7,  col: 6,  type: PieceType.White,  alive: true, id: "2"));
for (int i = 0; i < this.cells[1].length ; i++) {
    this.cells[1][i].setPiece(new Pawn( row: 1,  col: i,  type: PieceType.Black,  alive: true, id: String.valueOf(i)));
    this.cells[6][i].setPiece(new Pawn( row: 6,  col: i,  type: PieceType.White,  alive: true, id: String.valueOf(i)));
}
```

Todo tests esta bien:

```
Nothing to compile - all classes are up to date

--- maven-surefire-plugin:2.22.0:test (default-test) @ Chess ---

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running pedro.ieslaencanta.com.chess.controller.GameTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.083 s - in pedro.ieslaencanta.
Running pedro.ieslaencanta.com.chess.model.BoardTest
Tests run: 21, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.023 s - in pedro.ieslaencanta

Results:

Tests run: 26, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  3.816 s
Finished at: 2023-04-28T01:43:15+02:00
------------------------------------------------------------------------
```