# Testrapport

Testene som er utført tar for seg de mest kompliserte metodene i spillet. De metodene som er testet her er de vi har hatt problemer med under utviklingen. Det er også noen metoder vi har sett det mest hensiktmessig å praktisk teste, istedenfor å skrive JUnit tester, som for eksempel isChecked metoden i klassen Board. Metoder som har vært problemfrie, og metoder som er hjelpemetoder har vi heller ikke laget JUnit tester av. Testene følger for øvrig med på vedlagt CD.

## JUnit tester:

### Test 1:

Her testes getLegalMoves() metoden i GameRules klassen. Denne metoden bruker alle de andre metodene i GameRules, som hjelpemetoder, se javadoc.

package chess.Logic;

import chess.Logic.Board;

import chess.Models.Pawn;

import chess.Models.Piece;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

import static org.junit.Assert.\*;

public class GameRulesTest {

public GameRulesTest() { }

@Test

public void testGetLegalMoves() {

System.out.println("getLegalMoves");

Board b = new Board();

Piece p = b.getPieceAt(7, 1); // Dette er en hest

GameRules instance = new GameRules();

if (instance.getLegalMoves(p, b)[5][2] == 1) {

if (instance.getLegalMoves(p, b)[5][0] == 1) {

System.out.println("Success! Hesten på (1,7) har indeed låv å flytte til (2,5) og (0,5).");

}

}

if (instance.getLegalMoves(p, b)[4][4] != 1) {

System.out.println("Success! Hesten på (1,7) skal ikke få lov til å flytte til (4,4)");

}

Piece p2 = b.getPieceAt(6, 2); // Dette er en bonde.

if(instance.getLegalMoves(p2, b)[4][2] == 1){

if(instance.getLegalMoves(p2, b)[5][2] == 1){

System.out.println("Success! Bonden på (2,6) har låv til å flytte til (2,4) og (2,5)!");

}

}

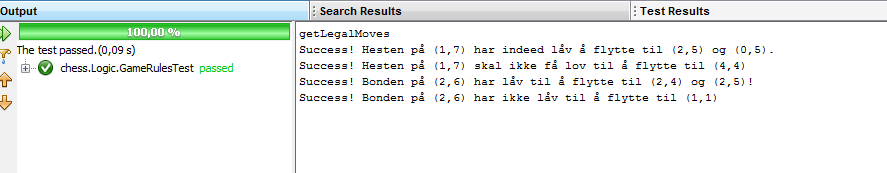
if(instance.getLegalMoves(p2, b)[1][1] !=1){

System.out.println("Success! Bonden på (2,6) har ikke låv til å flytte til (1,1)");

}

}

}



Figur - Test 1

Test 2: **Her testes metodene i Board klassen, som vi har sett hensiktsmessig å teste med JUnit.**

**import chess.Models.Piece;**

**import java.util.ArrayList;**

**import org.junit.After;**

**import org.junit.AfterClass;**

**import org.junit.Before;**

**import org.junit.BeforeClass;**

**import org.junit.Test;**

**import static org.junit.Assert.\*;**

**public class BoardTest {**

**public BoardTest() {**

**}**

**/\*\***

**\* Test of movePieceToPosition method, of class Board. Shows that a pawn at (7,1) can move to (7,2).**

**\* We also test the method getPieceAt() here.**

**\*/**

**@Test**

**public void testMovePieceToPosition() {**

**System.out.println("-----------\nmovePieceToPosition");**

**int x = 7;**

**int y = 2;**

**Board b = new Board();**

**Piece p = b.getPieceAt(1, 7); // p er nå en sort bonde.**

**System.out.println(p + " x:" +p.getX() + " y:" + p.getY());**

**b.movePieceToPosition(x, y, p, b); //No check if legal, just force move.**

**System.out.println(p + " x:" +p.getX() + " y:" + p.getY());**

**if(b.getPieceAt(y, x) == null){**

**fail("Did not move piece");**

**}**

**}**

**/\*\***

**\* Test of killPiece method, of class Board.**

**\*/**

**@Test**

**public void testKillPiece() {**

**System.out.println("killPiece");**

**Board b = new Board();**

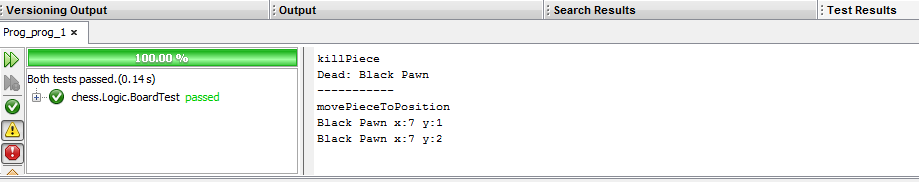
**Piece p = b.getPieceAt(1, 7); // p er nå en sort bonde.**

**b.killPiece(p);**

**System.out.println("Dead: " + b.getDeadBlack().get(0));**

**}**

**}**

****

Figur - Test 2

## Praktisk testing:

Vi har hele veien testet alt grundig praktisk. Men metoder som har skilt seg ut er disse:

* isChecked() – metode i klassen Board. Denne metoden ble skrevet om flere ganger, og var litt vrien. Men etter litt fram å tilbake med testingen, fikk vi reversert trekket som skulle reverseres om kongen fremdeles var i sjakk, og samtidig la samme spiller fortsette med et nytt trekk.
* Board.java sin movePieceToPosition er testet i praksis og fungerer.
* Bonder blir gjort om til dronning hvis de når motsatt side.
* Rokkering er testet i praksis og det fungerer.
* GameRules.java sin getLegalMoves() er testet i praksis og fungerer.
* Board.java sin UndoMove() fungerer for det meste. Alle trekk kan angres, bortsett fra spesialtekk, som omforming av bonde til dronning og rokkering.
* Lagring og lasting av spill er testet i praksis og fungerer utmerket.

## Konklusjon:

Spillet har nådd et nivå hvor det er fullt spillbart. Den styggeste buggen intreffer kun hvis man prøver å angre et spesialtrekk(rokkering, transformasjon). De øvrige funksjonene ser ut til å fungere som tenkt.