```java
package Project3;

import org.junit.Test;

import static org.junit.Assert.*;

/**********************************************************************
 * CIS 163 Section 01
 * Project 3: Chess Game
 * ChessTesting Class
 *
 * This class contains tests for all methods in ChessModel.
 *
 * @author George Fayette
 * @version 1/14/2019
 *********************************************************************/
public class ChessTesting {

    @Test
    public void testBlackPawnMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(1, 0, new Pawn(Player.BLACK));
        game.tryMove(new Move(1, 0, 3, 0));
        assertTrue(game.pieceAt(3, 0).type().equals("Pawn"));
        assertTrue(game.pieceAt(3, 0).player() == Player.BLACK);
    }

    @Test
    public void testWhitePawnMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(6, 0, new Pawn(Player.WHITE));
        game.tryMove(new Move(6, 0, 4, 0));
        assertTrue(game.pieceAt(4, 0).type().equals("Pawn"));
        assertTrue(game.pieceAt(4, 0).player() == Player.WHITE);
    }

    @Test
    public void testBlackKingMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 4, new King(Player.BLACK));
        game.tryMove(new Move(0, 4, 0, 3));
        assertTrue(game.pieceAt(0, 3).type().equals("King"));
        assertTrue(game.pieceAt(0, 3).player() == Player.BLACK);
    }

    @Test
    public void testWhiteKingMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(7, 4, new King(Player.WHITE));
        game.tryMove(new Move(7, 4, 7, 5));
        assertTrue(game.pieceAt(7, 5).type().equals("King"));
        assertTrue(game.pieceAt(7, 5).player() == Player.WHITE);
    }

    @Test
    public void testBlackQueenMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 3, new Queen(Player.BLACK));
        game.tryMove(new Move(0, 3, 3, 0));
        assertTrue(game.pieceAt(3, 0).type().equals("Queen"));
        assertTrue(game.pieceAt(3, 0).player() == Player.BLACK);
    }

    @Test
    public void testWhiteQueenMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 3, new Queen(Player.WHITE));
        game.tryMove(new Move(0, 3, 3, 0));
        assertTrue(game.pieceAt(3, 0).type().equals("Queen"));
        assertTrue(game.pieceAt(3, 0).player() == Player.WHITE);
    }

    @Test
    public void testBlackBishopMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 2, new Bishop(Player.BLACK));
        game.tryMove(new Move(0, 2, 2, 0));
        assertTrue(game.pieceAt(2, 0).type().equals("Bishop"));
        assertTrue(game.pieceAt(2, 0).player() == Player.BLACK);
    }

    @Test
    public void testWhiteBishopMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 2, new Bishop(Player.WHITE));
        game.tryMove(new Move(0, 2, 2, 0));
        assertTrue(game.pieceAt(2, 0).type().equals("Bishop"));
        assertTrue(game.pieceAt(2, 0).player() == Player.WHITE);
    }


    @Test
    public void testBlackKnightMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 1, new Knight(Player.BLACK));
        game.tryMove(new Move(0, 1, 2, 0));
        assertTrue(game.pieceAt(2, 0).type().equals("Knight"));
        assertTrue(game.pieceAt(2, 0).player() == Player.BLACK);
    }

    @Test
    public void testWhiteKnightMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 1, new Knight(Player.WHITE));
        game.tryMove(new Move(0, 1, 2, 0));
        assertTrue(game.pieceAt(2, 0).type().equals("Knight"));
        assertTrue(game.pieceAt(2, 0).player() == Player.WHITE);
    }

    @Test
    public void testBlackRookMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 0, new Rook(Player.BLACK));
        game.tryMove(new Move(0, 0, 0, 7));
        assertTrue(game.pieceAt(0, 7).type().equals("Rook"));
        assertTrue(game.pieceAt(0, 7).player() == Player.BLACK);
    }

    @Test
    public void testWhiteRookMove() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 0, new Rook(Player.WHITE));
        game.tryMove(new Move(0, 0, 0, 7));
        assertTrue(game.pieceAt(0, 7).type().equals("Rook"));
        assertTrue(game.pieceAt(0, 7).player() == Player.WHITE);
    }


    @Test
    public void testCastling() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(0, 0, new Rook(Player.BLACK));
        game.setPiece(0, 4, new King(Player.BLACK));
        game.tryMove(new Move(0, 4, 0, 2));
        assertTrue(game.pieceAt(0, 3).type().equals("Rook"));
        assertTrue(game.pieceAt(0, 3).player() == Player.BLACK);
        assertTrue(game.pieceAt(0, 2).type().equals("King"));
        assertTrue(game.pieceAt(0, 2).player() == Player.BLACK);
    }

    @Test
    public void testEnPassant() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(1, 0, new Pawn(Player.BLACK));
        game.setPiece(4, 1, new Pawn(Player.WHITE));
        game.tryMove(new Move(4, 1, 3, 1));
        game.tryMove(new Move(1, 0, 3, 0));
        game.tryMove(new Move(3, 1, 2, 0));
        assertTrue(game.pieceAt(2, 0).type().equals("Pawn"));
        assertTrue(game.pieceAt(2, 0).player() == Player.WHITE);
        assertTrue(game.pieceAt(3, 0) == null);
    }


    @Test
    public void testUndo() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(1, 0, new Pawn(Player.BLACK));
        game.tryMove(new Move(1, 0, 3, 0));
        game.undo();
        assertTrue(game.pieceAt(1, 0).type().equals("Pawn"));
        assertTrue(game.pieceAt(1, 0).player() == Player.BLACK);

    }


    @Test
    public void testUpgrade() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(5, 0, new Pawn(Player.WHITE));
        game.tryMove(new Move(5, 0, 3, 0));
        game.upgradePawn("Queen");
        assertTrue(game.pieceAt(3, 0).type().equals("Queen"));
        assertTrue(game.pieceAt(3, 0).player() == Player.WHITE);

    }

    @Test
    public void testUpdateStatus() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(5, 0, new Pawn(Player.WHITE));
        game.setPiece(1, 0, new Pawn(Player.BLACK));
        game.tryMove(new Move(5, 0, 3, 0));
        game.updateStatus();
        assertTrue(game.GUIcode() == GUIcodes.NO_MESSAGE);

    }

    @Test
    public void testAI() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(7, 0, new Rook(Player.WHITE));
        game.setPiece(1, 0, new Pawn(Player.BLACK));
        game.tryMove(new Move(7, 0, 6, 0));
        game.AI();
        assertTrue(game.pieceAt(2, 0).type().equals("Pawn"));
        assertTrue(game.pieceAt(2, 0).player() == Player.BLACK);
    }

    @Test
    public void testCheckMate() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(7, 0, new Rook(Player.WHITE));
        game.setPiece(6, 1, new Rook(Player.WHITE));
        game.setPiece(0, 0, new King(Player.BLACK));
        game.tryMove(new Move(7, 0, 6, 0));
        assertTrue(game.isCheckmate());
    }

    @Test
    public void testDraw() {
        ChessModel game = new ChessModel();
        game.clearBoard();
        game.setPiece(1, 7, new Rook(Player.WHITE));
        game.setPiece(7, 1, new Rook(Player.WHITE));
        game.setPiece(0, 0, new King(Player.BLACK));
        game.tryMove(new Move(7, 1, 6, 1));
        assertTrue(game.isDraw());
    }

    @Test
    public void testMove() {
        Move move = new Move();
        move = new Move(7, 1, 6, 1);
        assertTrue(move.toString()
                .equals("Move [fromRow=7, fromColumn=1, toRow=6, toColumn=1]"))
    }


}

```