White Hat Gaming Platform Programming Test

Introduction

This exercise is intended to test your design and programming ability in Java or any other JVM-based language that can call Java libraries.

There are no rigid assessment criteria. We will be looking at a number of factors including overall design, code readability, maintainability, issues found.

Once you have completed the test please package up your solution (zip or similar) and email to the person who sent you the test. At some point soon after completion we will do a walkthrough call with you where you explain your solution and we ask additional questions.

Problem Definition

You have been asked to write a computer program to allow two human players to play chess.

A good reference for the rules and general set up for chess can be found on Wikipedia:
http://en.wikipedia.org/wiki/Rules_of_chess

The program should simply read in moves and validate them, tracking and showing the board state. It should determine if a move leaves the player is in check. It does not need to handle checkmate.

Provided is a Java interface *com.whitehatgaming.UserInput* via which each attempted move can be obtained. You do not need to implement this interface. A single implementation *com.whitehatgaming.UserInputFile* is also provided which will read the moves from a text file. Three sample move files are provided in the data directory. Feel free to add additional files and/or write automated tests as required.

Please see the Javadoc for *UserInput* to see how the coordinate system for the chess board is represented in the input data.

Requirements

1. The board should start in the standard chess starting state with all the 16 pieces lined up for each player.
2. Play starts with player 1 (white) and on each valid move alternates to the other player. On an invalid move the existing player stays in control.
3. The moves must be read in using the supplied *UserInputFile* class until there are no more moves
4. All moves must have a piece on the starting square and either an opponent piece or nothing on the destination square Anything else is invalid.
5. Validate the move according to the moves allowed by the piece on the starting square:
    - The king can move only 1 square but in any direction
    - The bishop can move any number of squares but only diagonally
    - The rook can move any number of squares but only horizontally or vertically
    - The queen can move any number of squares horizontally, vertically or diagonally.
    - The knight can move in an L shape with sides of 2 and 1 squares respectively. That is 8 different possible moves. Unlike other pieces it jumps over other pieces.
    - The pawn can move one or two squares forward on its first move (when not taking an opponent piece)
    - The pawn can move one square forward on subsequent moves (when not taking an opponent piece)
    - The pawn can move one square forward diagonally if taking an opponent piece
6. After each successful move render the board in simple ASCII form. It is suggested that player 1 is represented by upper-case characters and player 2 by lower-case characters. The conventional characters to use here are: Rook, kNight, Bishop, King, Queen, Pawn.
7. If the destination square contains an opponent piece then that piece is removed from the board. Unless that piece is a King where rules around check apply (see later)
8. For pieces other than the knight disallow the move if there are any other pieces in the way between the start and end square.
9. If a move ends with a player's king under attack that is "check"
10. A player cannot end their own move in check
11. If a player starts their move in check this should be displayed as "in check"