# MegaBuyte Java Programming Test

## Introduction

This exercise is intended to test the candidate's design and programming ability in Java. As far as is possible, it intends to replicate the normal working environment. As such, it is not designed to be a test of memory. Manuals etc can - and should - be referred to. Recruitment is a two-way process: this exercise is also intended to allow candidates to sample the working environment at Megabuyte.

The tasks consists of a number of steps. Each step should be completed before the next one is attempted. It is *not* expected that in the time allocated that all steps will be completed.

There are no rigid assessment criteria. We will be looking at a number of factors including overall design, code readability, maintainability, amount of work completed etc.

The person administrating your test will be at hand to answer any questions. Please do not hesitate to contact this person if you get stuck at any stage or if any part of these instructions are unclear.

## Problem Definition

You have been asked to write a computer program to allow two human players to play chess[[1]](#footnote-1). If you are not familiar with the rules of chess please ask your test administrator!

An implementation of the following interface which gives the next move of each player will be provided for you :-

*public interface UserInput*

*{*

*/\*\**

*This method obtains the next move from the next player and populates the from[] and to[] arrays accordingly. These arrays should be of length at least 2.*

*On exit, from[0] and from[1] store the x and y co-ordinate respectively of the piece to be moved.*

*Similarly, to[0] and to[1] store the x and y co-ordinate respectively of the destination location.*

*The x, y co-ordinates are in the range 0,0 (bottom left) to 7,7 (top right). The same co-ordinate system is used for both players.*

*The first time this function is called it returns the move for player 1 (whose pieces occupy the positions 0,0 to 7,1). It then returns moves for alternating players.*

*This function returns false when there are no more moves. Otherwise, it returns true.*

*\*/*

*public boolean nextMove(int[]from, int[]to) throws UserInputException;*

*}*

An implementation of this interface, called UserInputFile, which reads moves from a text file whose name is passed in the constructor is provided for you. Some sample text files will be provided. Your test administrator will advice you where these are on the test machine.

## Step 1.

Create a program which creates an instance of UserInputFile and repeatedly calls the nextMove() method and simply outputs the desired moves onto standard out. Sample output could be :-

Player 1: move from 4,1 to 4,3

Player 2: move from 4,6 to 4,4

## Step 2.

Model and create a board object containing pieces. Initialise the board with the default Chess starting position.

## Step 3.

Update the state of the board according to the moves returned by nextMove().

## Step 4.

Modify the program to test the validity of each move (at this stage do not worry about the special case of a player being in "check").

## Step 5.

Modify the program to output (in simple ASCII form) the board after each move. It is suggested that player 1 is represented by upper-case characters and player 2 by lower-case characters.

## Step 6.

Modify the program to test for and take appropriate action on "check" and "checkmate".

1. There are a number of minor simplifications. In this version players are not allowed to castle, promote pawns or take pieces via "en passant". [↑](#footnote-ref-1)