Christopher Newport University
Department of Physics, Computer Science & Engineering
CPSC 280: Introduction to Software Engineering

* Assignment 1: Chess Check *

Instructor: Dr. Roberto A. Flores

Goal

Practice Personal Software Process Level 0 (PSP0).

Description

Write a program that receives a chessboard as input and indicates whether a king is checked or not– where a king is checked if it is in a square that can be taken by an opponent's piece.

A chessboard is an 8-by-8 board, where empty spaces are represented by dots, white pieces by uppercase letters, and black pieces by lowercase letters. The figure below shows a chessboard with all pieces at the beginning of a game. White pieces move upward and black pieces move downward.

```
rnbqkbnr
pppppppp
........
........
........
........
PPPPPPPP
RNBQKBNR
```

Each chess piece can move as follows:

- Pawn (p or P): It moves straight ahead, one square at a time, and takes pieces diagonally.
- Knight (n or N): Its move resembles an “L”, and it is the only piece that can jump over other pieces.
- Bishop (b or B): It moves any number of squares diagonally.
- Rook (r or R): It moves any number of squares vertically or horizontally.
- Queen (q or Q): It moves any number of squares in any direction (diagonally, horizontally or vertically).
- King (k or K): It moves one square at a time in any direction (diagonally, horizontally or vertically).

The figures below show the movements of these pieces (where ‘*’ indicates the relative location where another piece can be taken):

Bishop	King	Knight	Pawn	Queen	Rook

`.......*`
`*.....*.`
`.*...*..`
`..*.*...`
`...b....`
`..*.*...`
`.*...*..`
`*.....*.`
	`........`
	`........`
	`........`
	`..***...`
	`..*k*...`
	`..***...`
	`........`
	`........`
		`........`
		`........`
		`..*.*...`
		`.*...*..`
		`...n....`
		`.*...*..`
		`..*.*...`
		`........`
			`........`
			`........`
			`........`
			`........`
			`...p....`
			`..*.*...`
			`........`
			`........`
				`...*...*`
				`*..*..*.`
				`.*.*.*..`
				`..***...`
				`***q****`
				`..***...`
				`.*.*.*..`
				`*..*..*.`
					`...*....`
					`...*....`
					`...*....`
					`...*....`
					`***r****`
					`...*....`
					`...*....`
					`...*....`

Pawn movements will depend on its side. Black pawns can only take a piece one square diagonally down. White pawns can only take a piece one square diagonally up. The figure above shows that a black pawn (lowercase ‘p’) could take another piece from the immediate square diagonally and down.

Implementation

Write a class “Chess” with the method “getCheck” below (you can have additional methods if needed).

public static char getCheck(char[][] board)

This method receives a 2-dimensional array representing a chessboard, which will always be an 8-by-8 array of characters. Pieces will be arranged in arbitrary configurations. There will be no configuration where both kings are in check or where more than one piece checks a king.

The method should return the piece checking the king (e.g., ‘q’), or ‘-’ (dash) if no king is checked.

For example, given the board below, the method returns ‘B’ (i.e., white bishop is checking black king).

```
{ '.','.','k','.','.','.','.','.' }
{ 'p','p','p','.','p','p','p','p' }
{ '.','.','.','.','.','.','.','.' }
{ '.','R','.','.','.','B','.','.' }
{ '.','.','.','.','.','.','.','.' }
{ '.','.','.','.','.','.','.','.' }
{ 'P','P','P','P','P','P','P','P' }
{ 'K','.','.','.','.','.','.','.' }
```

Initial Files

You are given a JUnit class “ChessTest” to test your implementation.

In addition, you’re given the Word files “PSP0 forms” and “PSP0 Checklist”. “PSP0 Forms” has empty time, defect and summary to record your PSP0 data. “PSP0 Checklist” lists the expectations for grading filled out forms.

If needed, refer to the class slides and Humphrey’s book for PSP0 instructions and scripts.

Submission & Grading

Submit your “*.java” files (your program) plus your filled out PSP forms (with your PSP0 data) to Scholar.

Grades for this assignment will be distributed as follows: 50% for complete PSP0 forms (see “PSP0 Checklist” for expectations), and 50% for a program that executes according to this description and the provided test cases.

Test cases must not be hardcoded.

If fields are defined they must be private. If they were static they must be final.

Be reminded that no code should be disclosed to nor taken from others (including online sources).

•••