**CPSC ASSIGNMENT 1 REPORT**

Refactoring 1

This refactoring changed the code of the protected “isPathClearMethod” of the abstract Piece class. This method was used by all its subclasses except for the King and Knight class.

The method was not working properly as detected by: testBishopCanMoveDiagonalMoveUpRightPathNotClear() and testBishopCanMoveDiagonalMoveDownLeftPathNotClear() methods in the BishopTest JUnit test case as well as the equivalent methods in the QueenTest JUnit test case. The method also exhibited “Duplicate Code” and “Long Method” code smells.

I applied the “Substitute Algorithm” refactoring by ensuring that the prepared alternative algorithm compiled successfully, followed by running the already prepared “…PathNotClear()” JUnit test cases for all Piece subclasses that are able to move multiple squares and cannot skip over pieces. The newly improved algorithm passed all tests successfully.

The result of the refactoring is the “isPathClearMethod()” in the abstract Piece class.

Because the “isPathClearMethod()” is protected, it was tested by running JUnit test cases on the canMove() method of the Piece subclasses that can move multiple pieces in a direction, without being able to skip over other pieces. For each direction one of these pieces can move, a Board object would be initialized with a different piece placed in the way of movement. The canMove() method of these pieces will then call isPathClearMethod() and return false if detecting the piece along the path of movement.

Aside from now successfully being able to tell when a path is clear for diagonal movement, the new algorithm is also much shorter, due to less code duplication, and easier to understand, due to previous nested conditionals with while loops.