Assignment 1: PathOfDestruction

Dr. Sharma Thankachan

COP 3203 – Fall 2018

Due September 1st, 2018

1. Deliverable:

A source file named *PathOfDestruction.java* written in Java.

2. Objective:

The objective of this assignment is to get you warmed up for CS2 and for a semester of Java. This assignment requires only basic Java knowledge and an understanding of CS1 concepts, and is not meant to take a long time. It is not particularly lengthy, but will hopefully make you think and plan a little bit.

3. Description:

The game of American checkers has two types of pieces: regular checkers and king checkers. Regular checkers placed on a checkerboard must follow one restriction: all checkers must be on the same color as all other checkers placed. Figures 1 and 2 below are examples of valid checkerboards (where a capital O symbolizes a checker). For this assignment the color of the checker does not matter. Figure 3 is invalid because not all of the checkers occupy the same color.

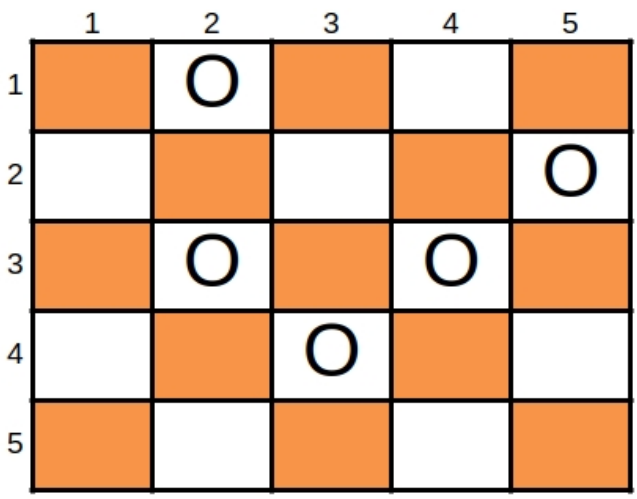

Figure 1

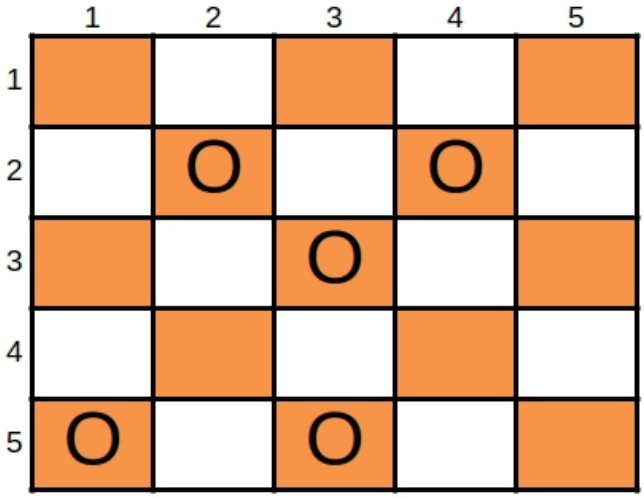

Figure 2

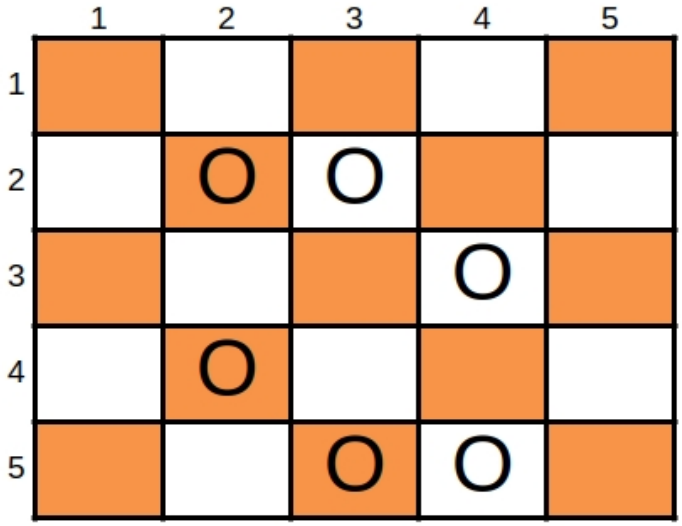

Figure 3

King checkers can move more freely than normal checkers on a checkerboard. They can move diagonally by one space in any direction (Figure 4), or they can hop over other checkers in any diagonal direction (Figure 5). These hops can only occur if the hopped checker is directly, diagonally adjacent. Whenever a checker is hopped, it is removed from the board (and thus cannot be hopped again). Kings are represented with a capital K.

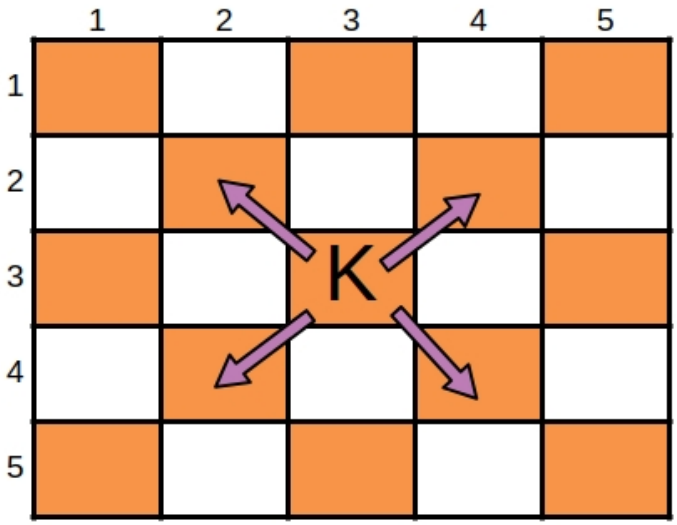

Figure 4

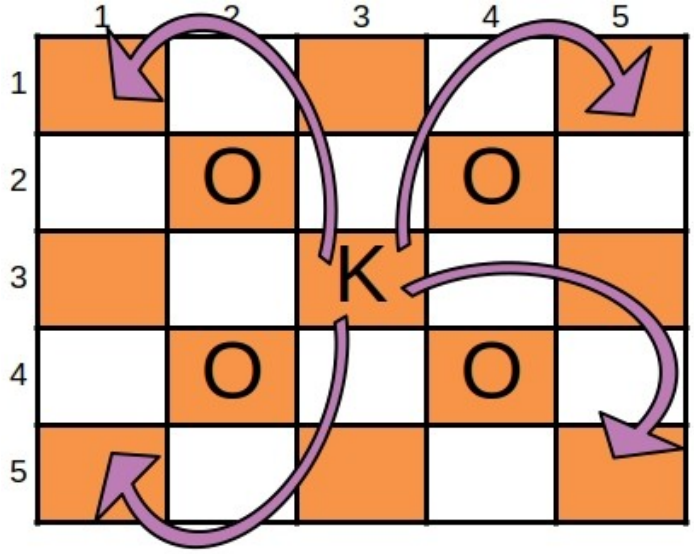

Figure 5

When a king hops a checker, it can then attempt to make another hop if one is available from its new position. This results in a "chain" of hops. Any chain of hops which jumps over every checker on the board can be considered a "path of destruction" and this path would remove all checkers (except the king) from the board. The next page gives an example of a possible path of destruction on an arbitrary checkerboard, (Figure 6) and an example of a checkerboard where no path of destruction is possible, (Figure 7). The primary goal of this assignment is to check if a path of destruction exists for a variable-sized checkerboard with an arbitrary layout of checkers.

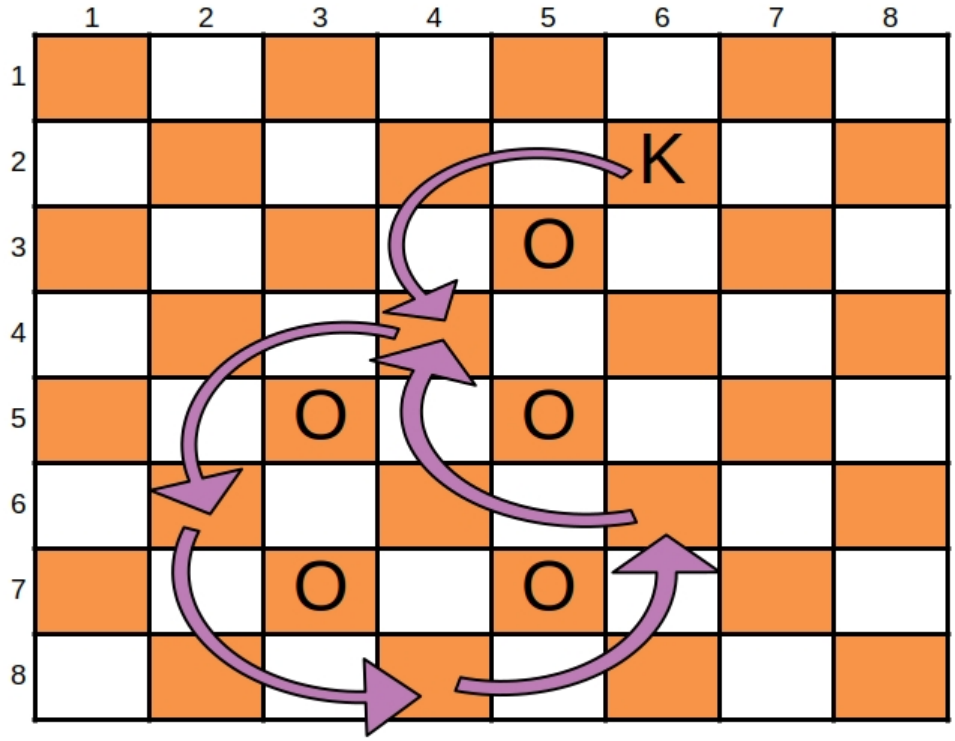

Figure 6

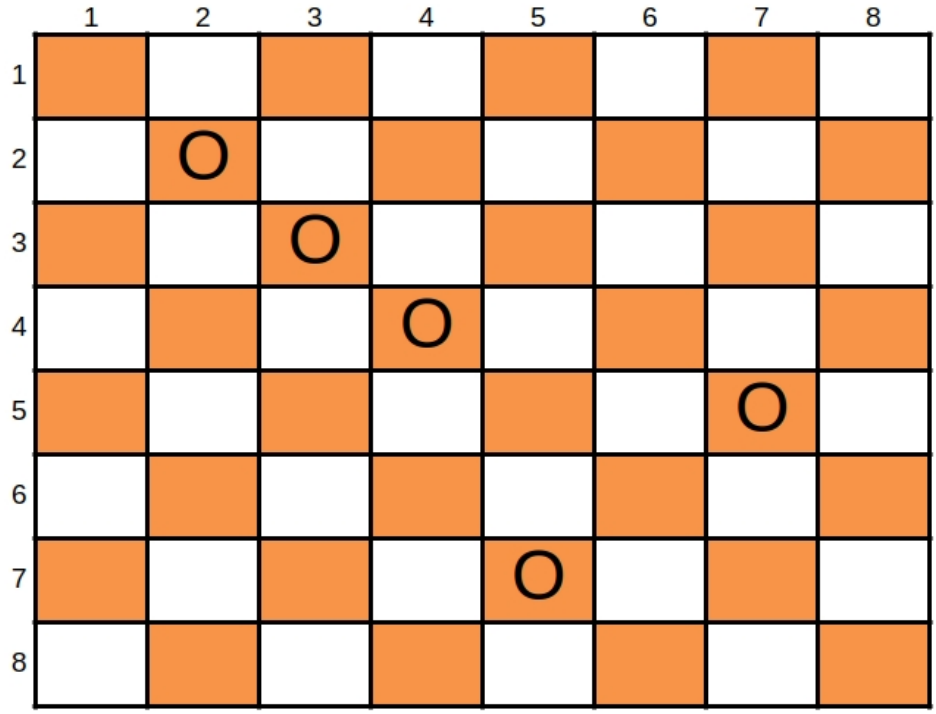

Figure 7

4. Requirements:

This assignment be named *PathOfDestruction.java* and contain a public class named *PathOfDestruction*. Within this public class you must write these 4 methods:

```
public static boolean isValidBoard(String boardFile) throws IOException
```

This method will return true if the checkerboard described in the file *boardFile* has a valid layout of checkers. Recall that all checkers on a checkerboard must be on the same "color" as all other checkers on that board. This can be determined mathematically, so to write this function properly you'll need to discover that relationship. Refer to the input specifications for more detail about the potential contents of *boardFile.* Return false if the checkerboard is invalid.

NOTE: It is highly recommended to draw out some checkerboards to try and deduce the relationship on paper before proceeding.

```
public static boolean hasPathOfDestruction(String boardFile) throws IOException
```

This method will return true if the checkerboard described in the file *boardFile* contains a path of destruction, otherwise return false.

NOTE: Boards containing no checkers have paths of destruction, since a king placed on such a board can follow a path that results in all of the checkers on the board being removed (by just staying put). This is a type of vacuous truth.

NOTE: Boards that contain an invalid layout of checkers cannot possibly have a path of destruction.

`public static double hoursSpent()`

This method returns a double, greater than 0.0, representing your best estimate of the total hours spent working on this assignment.

`public static double difficultyRating()`

This method returns a double, greater than or equal to 1.0 and less than or equal to 5.0, representing how difficult you found this assignment, where a rating of 1.0 indicates that this assignment was incredibly easy, and a rating of 5.0 indicates that this assignment was grotesquely difficult.

5. Input Specification

This is a sample input file that represents the checkerboard in Figure 6:

```
8 8 5
5 3
5 5
3 5
3 7
5 7
```

The first line of every checkboard file will contain 3 integers.

- The first is the maximum x coordinate that a board may have. All checkers must have x coordinates inside of the range [1, maximum x] on a valid board. The maximum x coordinate will always be greater than 0.
- The second is the maximum y coordinate. All checkers must have y coordinates inside of the range [1, maximum y] on a valid board. The maximum y coordinate will always be greater than 0.
- The third is the number of checkers on this board. This number will always be accurate and will be greater than or equal to 0.

The second line, and every line thereafter contain the x and y integer coordinates of a single checker, respectively. Checkers are guaranteed to be unique, but their coordinates are not guaranteed to be valid coordinates. For example, the valid checker (1, 13) could be found inside in a file with a maximum x of 20 and a maximum y of 30, but the invalid checkers (4000, 13) and (0, -22) may also be found within this file. It is your responsibility to check the coordinates of each checker as you parse the file to ensure they are within bounds for that given board.

6. Grading Criteria:

The grading criteria for this assignment is as follows:

50%	*hasPathOfDestruction()* unit tests
40%	*isValidBoard()* unit tests
5%	*hoursSpent()* unit test
5%	*difficultyRating()* unit test

Comments will not be graded generally, however I reserve the right to penalize assignments that include no comments, or whose comments are horrible. In the real world, comments are extremely useful and expected, so get used to writing them. I also reserve the right to penalize assignments with atrocious/inconsistent style (or lack thereof). This does not mean that you will lose points for using one particular style over another, as long as you apply your style consistently.

If you'd like to earn a 0, do any of the following:

- Submit the assignment with no name in a comment at the top.
- Submit the assignment with the wrong file name.
- Submit the assignment in a different programming language.
- Submit the assignment after the deadline.
- Cheat on the assignment/share code.
- Submit an assignment that doesn't compile.
- Submit an assignment that doesn't compile on Eustis.