CSC 260.002 Programming Assignment #9
Due date: Tuesday, November 29

In this assignment, you will implement a class called ChessPiece to represent a chess piece on a chess board. This will be a simple class (see the UML below) with four instance data, a constructor, accessors for all four instance data, mutators for row and column, and a toString which should return a String containing the ChessPiece's color and type, row and column value all on one line, as in `White Rook 0 0`. You may implement a 0-arg constructor if you wish but this is not necessary.

ChessPiece:

```
type, color: String
row, column: int

+ChessPiece(type: String, color: String, row: int,
     column: int)
+ getType(): String
+ getColor(): String
+ getRow(): int
+ getColumn(): int
+ setRow(int row): void
+ setColumn(int column): void
+ toString(): String
```

The setRow and setColumn mutators must ensure that the passed parameter is legal (between 0 and 7). If the passed parameter is incorrect, do not change the row/column.

Once the ChessPiece class is implemented, create a ChessGame class which will contain a main method as well as an inputFromDisk method, an inputMove method, a move method, an output chessboard method and a saveToDisk method. These are described later. The ChessGame will implement a chess board as an 8x8 array of ChessPiece objects. Each element of the chess board will either be a ChessPiece or null (to indicate an empty square).

The ChessGame program will operate as follows.

```
Declare the chessboard and initialize all 64 elements to null
Create a Scanner for user input
Pass the chessboard to the inputFromDisk method (described later)
Call the input method to get the starting row from the user
While (row>=1) {                          // if 0, exit the loop and end the game
    Call the input method to get the starting column from the user
    Call the input method to get the ending row from the user
    Call the input method to get the ending column from the user
    Move the ChessPiece from the starting row/col to ending row/col
    Call the output chessboard method
    Call the input method to get the starting row from the user
}
Call the saveToDisk method by passing the chessboard
```

For the inputFromDisk method, hardcode the name of the file you are inputting from (chessboard.txt, which is on the website) and input each line of input as two String values and two int values (e.g., "White", "Rook", 0, 0). Create a ChessPiece whose name, color, row and column are these four inputs, and assign chessboard[row][col] to equal the new piece. Close the file and return the chessboard object (the 2-D array). You should look at the chessboard.txt file to get an idea of how the input looks.

For the input method, pass it two Strings, "starting" or "ending" and "row" or "column". Input the int value from the user and data verify that the value is between 1 and 8 or, if it is starting row, that it is between 0 and 8 (as 0 for starting row ends the loop). Return the int value to main as in

```
startCol = getInput(in, "starting", "column");
```

where in is our keyboard Scanner.

The move method will receive the startRow, startCol, endRow and endCol. Subtract 1 from each of these so that they are in the range of 0-7. Now, test to see if there is a ChessPiece at startRow, startCol and if there is not a ChessPiece at endRow, endCol. If these are true, move the piece (we are not implementing capturing pieces so the destination location must be empty, and we will not be testing to see if the move is a legal chess move given the piece's type). If either the starting location is empty or the ending location is occupied, do not move the piece and instead output an error message. The move method will look something like the following.

```
if(board[startRow][startCol]!= null&&
   board[endRow][endCol]==null) {
     board[endRow][endCol] = board[startRow][startCol];
     board[endRow][endCol].setRow(endRow);
     board[endRow][endCol].setCol(endCol);
     board[startRow][startCol] = null;
}
else … indicate an illegal move (no piece at the given location)
```

The saveToDisk method will open the same file as your inputFromDisk method (chessboard.txt) so you will be replacing the old board set up with the new one. You will only call this method after the while loop in main indicating that the user has input 0 for the startRow. Iterate through the board, row-by-row, and if a ChessPiece is present at the current [i][j] location, output its color, type, row and column to the disk file. You can accomplish this by writing each value individually through the proper access method, or by using the toString method. Do not output any chessboard[i][j] if that location currently stores null. When done, close the output file. Remember both the inputFromDisk and saveToDisk (as well as main) must include `throws IOException` in their header. The inputFromDisk method uses a Scanner and the saveToDisk uses a PrintWriter.

The output chessboard method receives the chessboard and iterates through each row, outputting either the initials of the piece (first letter of color, first letter of type) or -- if the location in the array stores null (indicating an empty square). NOTE: In the input file, I am using "Horse" instead of "Knight" so that we can differentiate in the output the type of piece (using a 'K' for Knight would confuse the piece with a King). You do not need to worry about this, just continue to use "Horse" for the type instead of "Knight".

Below is an excerpt of running the program for two moves before exiting.

```
    WR  WH  WB  WQ  WK  WB  --  WR
    WP  --  WP  --  --  WP  WP  WP
    --  WP  --  WP  --  --  --  WH
    --  --  --  --  WP  --  --  --
    --  --  BP  --  BP  BB  --  --
    --  --  --  BP  --  --  --  --
    BP  BP  --  --  --  BP  BP  BP
    BR  BH  --  BQ  BK  BH  BB  BR

Starting row  2
Starting col  1
Ending row  4
Ending col  1

    WR  WH  WB  WQ  WK  WB  --  WR
    --  --  WP  --  --  WP  WP  WP
    --  WP  --  WP  --  --  --  WH
    WP  --  --  --  WP  --  --  --
    --  --  BP  --  BP  BB  --  --
    --  --  --  BP  --  --  --  --
    BP  BP  --  --  --  BP  BP  BP
    BR  BH  --  BQ  BK  BH  BB  BR

Starting row  8
Starting col  4
Ending row  5
Ending col  1

    WR  WH  WB  WQ  WK  WB  --  WR
    --  --  WP  --  --  WP  WP  WP
    --  WP  --  WP  --  --  --  WH
    WP  --  --  --  WP  --  --  --
    BQ  --  BP  --  BP  BB  --  --
    --  --  --  BP  --  --  --  --
    BP  BP  --  --  --  BP  BP  BP
    BR  BH  --  --  BK  BH  BB  BR

Starting row  0
```

Run your program on a few (at least 5) legal moves of your choice and then exit. This will change the stored file. Submit your source code for both classes, the output of the last board configuration (for instance, the last output shown above) and the stored file. Copy the output and the stored file into the source code of your ChessGame program for convenience and hand in both classes. Or, hand in both classes, and a printout of both the last output and the stored file.