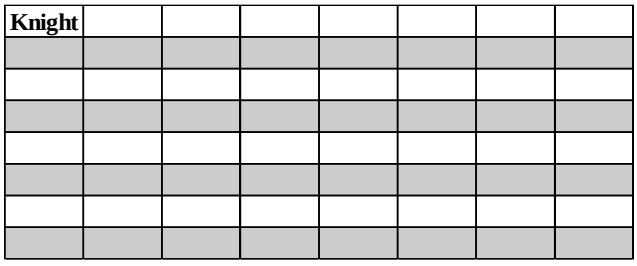
COP3503 Computer Science 2
Dr. Andrew Steinberg
Fall 2023

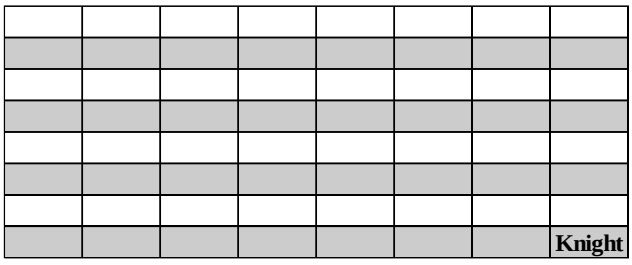
Programming Assignment 1
A Game That Requires Strategy
Due: 9/10/2023 at 11:59pm

Objective: Students will apply concepts of clever problem solving in this assignment and warmup with basic Java skills.

Assignment Description: We are going to play a fun strategy game that only requires two players! In this game, we have an 8 x 8 board and a Knight chess piece that starts on the top left of the board. Each player gets to move the Knight piece one square over either down, diagonal, or to the right of its current position (a player cannot move the piece two or more squares). The Knight piece can keep moving until it reaches the bottom right corner of the board. The respective player that moves the Knight to the bottom right corner of the board wins the game! In this assignment you are going to implement the winning strategy for both players.



When the game begins, the knight is on the top left square of the board.



When the game ends, the knight is on the bottom right square of the board.

For this assignment, you must follow these **requirements**.

1. You will create a public class called `Game`.
2. The `Game` Class will have the following attributes:
    a. 2D integer array that symbolizes the board.
    b. 1D primitive char array (not a String object) that will store computerize moves. There is no need to use the Random Class. Hint: This should be used in one of the hypothetical strategies for one of the players to always win.
3. The `Game` Class has a constructor that takes two parameters
    a. A primitive integer that represents the size of board (assume the board is always a square)
    b. A String object reference that contains the name of a file that keeps the moves the player 2 will use in the game.
4. The `Game` Class has a public non static method called `readMoves`. `readMoves` will scan the respective text files of moves and store them in the character array attribute. The characters ‘r’, ‘b’, and ‘d’ represent the moves.

a. 'r' means move to the right one square
b. 'b' means move down one square
c. 'd' means move diagonal one square

The method has one String reference parameter that represents the name of the text file. This method should be invoked when the object of the class is instantiated. The moves in the text files are only associated with player 2's moves.

5. The `Game` class has a public non static method called `play`. `play` will simulate a round of the game. The method has one parameter that represents the player you want to win the game. If a one is passed, then player 1 must win. If a two is passed, then player 2 must win. The method returns an integer type value that represents the winner of the game (1 being player 1 and 2 being player 2**). You can assume that player 1 will always go first in the game**, however based on the value past, you need to determine the appropriate strategy for always winning the game for the appropriate player. ***This also includes what the other player must do in their respective move. That means for this scenario there is something that needs to always happen for player 2 to win!!!*** **NOTE: If a student just simply writes the statement that returns the player number without any simulation to the game will receive an automatic score of 0 on the assignment with NO partial credit from the other categories in the rubric.**
6. You are allowed to create helper methods as long as they are not called directly from the driver file. The helper methods must be called from your solution file.

A driver file (`GameDriver.java`) has been provided for you to show you how the methods are called along with 10 test cases to see if you get the same scenario result.

What to submit: Submit a file called `Game.java` to webcourses. You are not required to submit the driver file as that will be provided for the graders to test your code. Please make sure the driver file provided works for your code. Any name changes may cause your program not to work when graded, which will result in a lower score on the assignment and would not be changed. You do not need to submit the text files of moves.

Comment Header: Please make sure to provide the appropriate comment header (like in the Eustis assignment) as the first thing on the top of your file. Otherwise, if your comment header is done incorrectly including not being placed literally at the top of the file will result point deductions in accordance with the rubric.

Hardcoding Output: For this assignment it is expected that you implement an actual solution simulation to the game. If a student just places in their code to return the respective value without any sort of simulation of the game, the student will automatically receive a score of 0 with NO partial credit.

Text Files: All text files are placed in the directory as your driver file and solution file. DO NOT CREATE SUBFOLDERS! If you create subfolders, then this will cause our batch grader to not run your code properly. This will result in points being deducted that cannot be disputed. You

can assume that all proper moves are associated and that there are no invalid characters in the text files.

Important Note for running Eustis and Packages: Many of you are probably using IDEs like Netbeans and Eclipse to build your Java Solutions. Please note that some of these IDEs will automatically place your Java file in some sort of package. Please make sure your Java file is not defined in some package as this can result package private errors. Any such error that occurs during the grading will not be fixed and points will be deducted as such in accordance with the respective categories in the rubric. Also, DO NOT create a main method in your solution file!! This will result in your code not running properly with the driver file which will result in points being deducted from the respective categories.