Assignment 1

Please read this document very carefully. Follow instructions exactly. If you have any questions please post them to MS Teams or ask during office hours.

This assignment is due May 21th, by 11:59pm

I have created an Assignment 1 channel in Teams. If you have questions about the assignment, please post them there. Thank you.

Overview

In this assignment you will implement several methods which would be useful for creating a Battleship game. More details regarding these specific methods are given later in this document. If you are not familiar with the game of Battleship, you can read this article:

`https://en.wikipedia.org/wiki/Battleship_(game)`

You are not responsible for making the game playable/complete. You are only responsible for implementing the methods in this document. For the purposes of this assignment we will make several assumptions.

1. The game will always be played on a 10x10 grid. This grid will be represented as a 2D array of Strings.
2. The board will contain four ships. A 2-ship, 3-ship, 4-ship and a 5-ship. Each of these ships takes up 2, 3, 4, and 5 spaces on the grad respectively. Ships are place vertically or horizontally on the grid.
3. Each cell on the game grid will contain a String which denotes the "state" of that cell. A cell can have one of the following values: `_, 2, 3, 4, 5, 2H, 3H, 4H, 5H`. Please find an interpretation for these values below.
   - `_`: There is no ship at this cell (it's water).
   - `2, 3, 4, 5`: Part of a 2, 3, 4, or 5, ship is at this cell. This spot has **not** been "hit".
   - `2H, 3H, 4H, 5H`: Part of a 2, 3, 4, or 5, ship is at this cell. This spot **has** been "hit".
4. A valid board is a 10x10 2D array of Strings where each cell is one of: `_, 2, 3, 4, 5, 2H, 3H, 4H, 5H`. Furthermore, the following criteria are met.
   - The total number of `2` and `2H` cells equals 2
   - The total number of `3` and `3H` cells equals 3
   - The total number of `4` and `4H` cells equals 4
   - The total number of `5` and `5H` cells equals 5
   - All ships are contiguous in their placement
   - All ships are horizontal or vertical in their placement

   As an example see these valid boards below (quotations are dropped for readability):

```
[[_, _, _, _, _, _, _, _, _, _],
 [_, 2, 2, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, 3, _, _, _, _, _, _],
 [_, _, _, 3, _, _, _, _, _, _],
 [_, _, _, 3, _, _, _, _, _, 5],
 [_, _, _, _, 4, 4, 4, 4, _, 5],
 [_, _, _, _, _, _, _, _, _, 5],
 [_, _, _, _, _, _, _, _, _, 5],
 [_, _, _, _, _, _, _, _, _, 5]]

[[2, 2, 5, 5H, 5, 5, 5H, _, _, _],
 [3H, 3H, 3H, 4, 4, 4H, 4, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _]]
```

As an example, below are some invalid boards:

```
[[2, _, _, _, _, _, _, _, _, _],
 [2, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, 3, 3H, 3, 3H, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, 4, 4, 4, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, 5],
 [_, _, _, _, _, _, _, _, _, 5],
 [_, _, _, _, _, _, _, _, _, 5],
 [_, _, _, _, _, _, _, _, _, 5H],
 [_, _, _, _, 2, 2H, _, _, _, 5]]

[[_, 3, 3, 3, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, 5, _, _, 2, 2, _, _, _],
 [_, _, 5, _, _, _, _, _, _, _],
 [_, _, 5, _, _, _, _, _, _, _],
 [_, _, 5, _, _, _, _, _, _, _],
 [_, _, 5, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _],
 [_, _, _, _, _, _, _, _, _, _]]
```

Methods

You are required to implement all of the methods below. Use the `.java` I posted as a starting point. Pay attention to parameters of each method, for example, if it is said an input will be a valid board, you can trust your method will never be tested on input which isn't a valid board. **You may also create and use helper methods if you wish. In fact, I recommend you do so.**

1. `is_valid_starting_board(String[][]) -> boolean`
   A valid starting board is a valid board with the additional constraint that all cells must be one of: `_, 2, 3, 4, 5`. That is, no cells have be "hit" yet. This method returns true if and only if the parameter is a valid starting board.

2. `shoot(String[][], int, int) -> String`
   The first parameter is a valid board, *b*. The second and third parameters are the row and column which is being "shot". You may assume these two parameters are between 0 and 9 inclusive. The method follows the below functionality.

   - If the value of `b[row][column]` is `"_"` the method returns: `"Miss"`. Nothing on the board is mutated.
   - If the value of `b[row][column]` is a number `k`, where `k` is one of 2, 3, 4, or 5, it mutates that cell on the board to now be `"kH"`. Then, if there still exists a cell equal to `k` it returns: `"Hit"`. Otherwise, it returns: `"Sunk"`.
   - If the value of `b[row][column]` is a number `kH`, where `k` is one of 2, 3, 4, or 5, it returns: `"Hit"`. Nothing on the board is mutated.

3. `can_place_ship_at(String[][], char, int, int, int) -> boolean`
   This method checks to see if a player could potentially add a ship to the board at a particular location. The first parameter is a valid board. The second parameter will be one of `'v'` or `'h'`. This will indicate if the player wants to place the ship vertically or horizontally respectively. The third and fourth parameter correspond to the start and end values of where the ship is intended to be placed. If the ship is being place horizontally then start and end will refer to columns; if placed vertically the start and end will refer to rows. Furthermore, the fifth and final parameter is the location (row or column) which the ship is being placed. If the ship is being placed horizontally, location will indicate the row the ship is being placed; if the ship is being placed vertically, location will indicate the column the ship is being placed. You may assume that start, end, and location are between 0 and 9 inclusive and start is less than or equal to end. The method will return true if and only if a ship can be placed where the parameters indicate. That is, there are no other ships in the way.

   ***Note: I have updated this method to include a location parameter.**

Submitting and Grading

This assignment will be submitted electronically via Avenue. You should submit one file to Avenue named `BattleShip.java` This assignment is worth 14% of your final grade. Your methods are grading 100% on correctness. No weight is put on efficiency, elegance of solution, etc. Grading is done automatically. That is, a program calls your method, passes it certain arguments, and checks to see if it returns the expected output. The assignment is broken down as follows:

- `is_valid_starting_board()`: 40%
- `shoot()`: 30%
- `can_place_ship_at()`: 30%

Good luck!

Academic Dishonesty Disclaimer

All of the work you submit must be done by you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code.

Please don't copy. The TAs and I want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

Never look at another assignment solution, whether it is on paper or on the computer screen. Never show another student your assignment solution. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in involve students who have never met, and who just happened to find the same solution online. If you find a solution, someone else will too.