Battleship Instructions

In this assignment, you will create a simple class for playing the game battleship. Battleship is played on a square grid of 10 rows and 10 columns, which you will represent using a 2-d array in a class named Board. A separate class, Battleship, will allow you to try playing your game and see whether the methods work as planned.

To play battleship, 2 players secretly add ships of various lengths to their individual boards. A ship may be placed horizontally or vertically on the board. In your program, you will represent blank squares on the board using the character ‘-’, and squares where ships have been added with the character ‘b’. For example, the board below contains 3 boats, 2 of length 5 and one of length 3:

- - - - - - - m - -

- x x x - - m x - -

- - - - - - - x - -

- - - m - - - b - -

- - - - - - - b - -

- m - - - - - b - -

- - - - m - - - - -

- - b b b b b - m -

- - - - - - - - - -

- - - - - m - - - -

Players then take turns to try and shoot the ships on each other’s boards by choosing a row and column reference for each attempted shot. When a player chooses a square containing a ship, that square is marked as a “hit”: in your program, this will be done using the character ‘x’. When the player “misses” by choosing a blank square this will be marked with the character ‘m’ in your program. The game is over when one player has shot every square containing part of a ship on their opponent's board. The board below is from a game in progress. One ship has already been completely destroyed and 2 squares of one of the other ships have been shot.

- - - - - - - m - -

- x x x - - m x - -

- - - - - - - x - -

- - - m - - - b - -

- - - - - - - b - -

- m - - - - - b - -

- - - - m - - - - -

- - b b b b b - m -

- - - - - - - - - -

- - - - - m - - - -

The specification for the Board class is as follows:

Variables

char[][] squares - An array which represents the board on which a player places their battleship and records shots.

Constructors

Board() - This constructor initializes the squares array with every value set to ‘-’, which is the character used to represent a blank square.

Methods

* **public String toString()** - Returns a multi-line representation of the board by printing each character in squares with spaces after each character and a new line for each row.
* **public boolean addShip(int row, int col, int len, boolean horizontal)** - Attempts to add a ship of length len to the grid, starting at the row and column specified and proceeding either rightwards (if horizontal is true), or downwards (if vertical is true). If the ship can be placed in the place specified, each square making up the ship should be set to ‘b’, and the method should return true. A ship may not be placed if it would go off the grid, or would intersect another ship on the grid. If the ship cannot be placed, no values in squares should be changed and the method should return false.
* **public int shoot(int row, int col)** - If row and col specify a square which is out of bounds, the method should return -1. If the square at the specified row and column contains ‘-’ (i.e. is blank), the square should be changed to ‘m’ to signify a miss, and the method should return 0. If the square contains ‘b’ (i.e. a battleship which hasn’t been hit yet) this square should be changed to an ‘x’ to signify a hit, and the method should return 1. If the square contains either ‘x’ or ‘m’ the method should return 2 (these are squares which have already been “shot”).
* **public boolean gameOver()** - If the character ‘b’ does not appear at any location in squares, then there are no unsunk ships remaining on the board, so return true to indicate that the game is over. Otherwise return false.
* **public boolean foundShip(int len)** - Search the board for any possible ships of length len. If there are exactly len consecutive squares (either horizontal or vertical) containing a ‘b’ character somewhere in the grid, then return true, otherwise, return false.

When you have created your Board class, download the Battleship.java class to the same folder. When you run the main method in this class, you will be prompted to add ships to the board, then to attempt to shoot these. See the sample run below for an example.

Sample Run

Welcome to Battleship!

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

Add a new ship? (y/n)

n

You need a ship of length 3 to play!

Add a new ship? (y/n)

y

Starting in which row?

2

Starting in which column?

3

How long?

3

Horizontal (h) or vertical (v)?

h

New ship added!

- - - - - - - - - -

- - - - - - - - - -

- - - b b b - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

Add a new ship? (y/n)

n

Ok, let's play!

Input row

5

Input column

6

Miss!

- - - - - - - - - -

- - - - - - - - - -

- - - b b b - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - m - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

Input row

2

Input column

3

Hit!

- - - - - - - - - -

- - - - - - - - - -

- - - x b b - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - m - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

Input row

2

Input column

4

Hit!

- - - - - - - - - -

- - - - - - - - - -

- - - x x b - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - m - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

Input row

2

Input column

5

Hit!

- - - - - - - - - -

- - - - - - - - - -

- - - x x x - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - m - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

- - - - - - - - - -

Game over!

You should run the game several times and try different inputs - this uses the methods from your Board class, so these will all need to produce the correct results and outputs if the program is to work as desired. You can also add, edit, or isolate code to help you with testing specific features.

Milestones

As you work on this assignment, you can use the milestones below to inform your development process:

**Milestone 1**: Write the Board class header, declare the squares variable and write the single constructor required.

**Milestone 2**: Write and test the toString and gameOver methods.

**Milestone 3**: Write and test the addShip method.

**Milestone 4**: Write and test the shoot method. Run the full Battleship program to test all the features.