**Unit 5 Programming Project**

Battleship

Battleship is a guessing game for two players. It is known worldwide as a pencil and paper game which dates from World War I. It was published by various companies as a pad-and-pencil game in the 1930s, and was released as a plastic board game by Milton Bradley in 1967. The purpose of the game is to destroy the opposing player's battleships, and ends with a fight to the death.

The game is played on four grids, two for each player. The grids are typically square and the individual squares in the grid are identified by letter and number. On one grid the player arranges ships and records the shots by the opponent. On the other grid the player records their own shots.

Before play begins, each player secretly arranges their ships on their primary grid. Each ship occupies a number of consecutive squares on the grid, arranged either horizontally or vertically. The number of squares for each ship is determined by the type of the ship. The ships cannot overlap (i.e., only one ship can occupy any given square in the grid). The types and numbers of ships allowed are the same for each player. These may vary depending on the rules.

After the ships have been positioned, the game proceeds in a series of rounds. In each round, each player takes a turn to announce a target square in the opponent's grid which is to be shot at. The opponent announces whether or not the square is occupied by a ship, and if it is a "miss", the player marks their primary grid with a white peg; if a "hit" they mark this on their own primary grid with a red peg. The attacking player notes the hit or miss on their own "tracking" grid with the appropriate color peg (red for "hit", white for "miss"), in order to build up a picture of the opponent's fleet. After the player gets a hit the player continues shooting until the player gets a miss.

When all of the squares of a ship have been hit, the ship is sunk, and the ship's owner announces this (e.g. "You sank my battleship!"). If all of a player's ships have been sunk, the game is over and their opponent wins.

For the unit 5 programming project, we will work on implementing a simple version of the battleship game. If you have never played battleship before, please try playing one of the online versions before starting the project as understanding the logic behind the game is critical for implementing the project code. An online version : <https://battleship-game.org/en/>

# CHECKPOINTS

This project is divided into 5 milestones to ensure that your battleship is on track for completion, all of which must be completed to obtain at least a MEETS for the assignment. To obtain at least a BASELINE for the assignment, checkpoints 1-3 must be completed.

## Checkpoint 1 - BOARD.java Set Up

For checkpoint 1, you must flesh out the basic functionality of the BOARD class that has been provided by implementing a constructor and three methods.

* Create a default constructor that initializes all class variables. Printing your board (using the provided toString() method should look like the board below at the start of the game.

|  |
| --- |
| **1 2 3 4 5 6 7 8**  **a · · · · · · · ·**  **b · · · · · · · ·**  **c · · · · · · · ·**  **d · · · · · · · ·**  **e · · · · · · · ·**  **f · · · · · · · ·**  **g · · · · · · · ·**  **h · · · · · · · ·** |

* Create a method to called markBoard that changes the value of the given letter and number position. This method takes in three parameters, the letter and number positions (both type char), and the SYMBOL index(int) that the board needs to be marked with and returns if the change has been successful.
* Create a method called containsLetter that takes in a char and returns a boolean that represents if the Board class constant array ALPHA contains that letter.
* Create another method called containsNumber that takes in a char and returns a boolean that represents if the Board class constant array NUMBER contains that number.

## Checkpoint 2 - Board.java Getter Methods

For checkpoint 2, you will need to implement a few methods to get information about the board however none of these methods should change the board’s state. This checkpoint will require 3 new methods to the Board class.

* Create a method called getValue that takes two parameters of char type which represent the location of a value on the board (1 letter, 1 number) and returns the value(char) of the position in the board. Remember to call the validChoice method before this one to prevent an Out of Bounds error.
* Create a method called checkShip that takes two parameters of char type which represent the location of a value on the board (1 letter, 1 number) and returns if there is a ship in that position. If the location does not exist, return false. Remember to call the validChoice method before this one to prevent an Out of Bounds error.
* Create a method called calculateHitOrMiss that takes three parameters. Two of the parmeters will be of char type which represent the location of a value on the board (1 letter, 1 number) and the third parameters will be of Board type which holds ships. Returns if a shot at that location would result in hit (1) or a miss (2). The SYMBOL class constant contains what the ship symbol is.

## Checkpoint 3 - Board.java Mutilator Methods

This will be the last checkpoint for the Board class. In this checkpoint, you will be updating your board based on input to the board. This checkpoint only has 1 new method to write.

* Create a method called placeShip that takes in 4 parameters (length of ship, letter/number starting position of ship and direction to place the ship) and returns if the ship was successfully place or not(boolean). Remember ships must fit on the board and also not overlap other ships. Do not start placing a ship unless you are sure of this. Don’t forget to take advantage of the methods you have created so far and provided to you.

## Checkpoint 4 - Battleship.java Game Play

In checkpoint 4, you will start programming the gameplay of Battleship in the Battleship.java class. There will be 2 methods to write in this checkpoint.

* Create a method called setGameUp that sets up the game. This method should take in a board per player as its parameters.
  + For each player, ask them to place their 5 ships, Carrier(length 5), battleship(length 4), Cruiser(length 3), submarine(length 3), destroyer(length 2).
  + Make sure to print out the state of the board after each ship has been placed so the player knows where their ship is.
* Create a method called playGame that allows the players to play the game. This board should take in both the player’s attack board and ship board as parameters and should allow the players to alternate attacking each other.

## Checkpoint 5 - Battleship.java End Game

Hooray! You are now on checkpoint 5. In this checkpoint, you will be

* Create a method called gameOver returns if the game is over(boolean). This method should take in each player’s attack boards..
* Update your playGame method to incorporate your gameOver method and have it announce a winner.
* Play the game a few times with friends :)