ACM Summer Internship 2014

ASSIGNMENT 3

Due Date: 30th June 2014, 11pm.

Instructions:

- Everyone is supposed to do this assignment individually. Try to use Java Docs as much as you can to help you out.
- Those of you who don't submit the assignment will get zero points.

NOTE:
THERE IS A STRICT POLICY AGAINST PLAGIARISM. ANYONE FOUND COPYING CODE FROM SOMEONE ELSE OR THE INTERNET WILL BE DISQUALIFIED FROM THE INTERNSHIP.

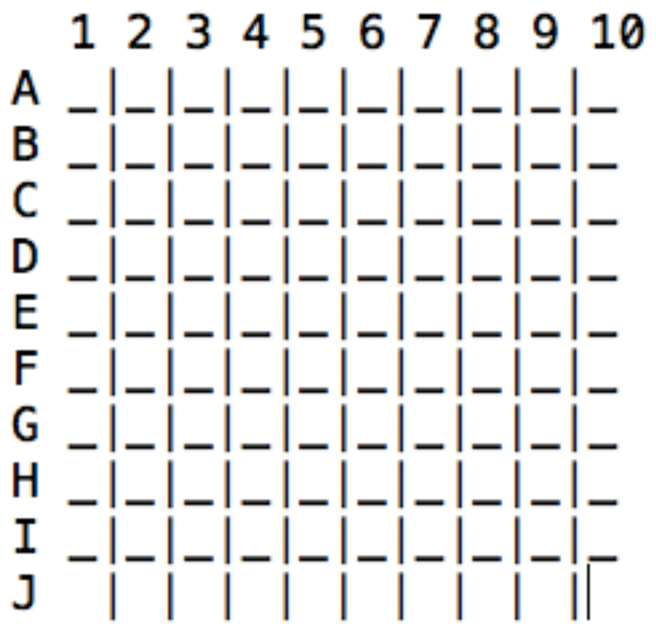
BattleShip

Part 1:

Single-Player Version

You are to implement a command line interface version of the classic board game BattleShip!
To get a rough idea of the game, refer to the following link.
http://en.wikipedia.org/wiki/Battleship_(game)

You will create a 10 x 10 board which will initially be empty.

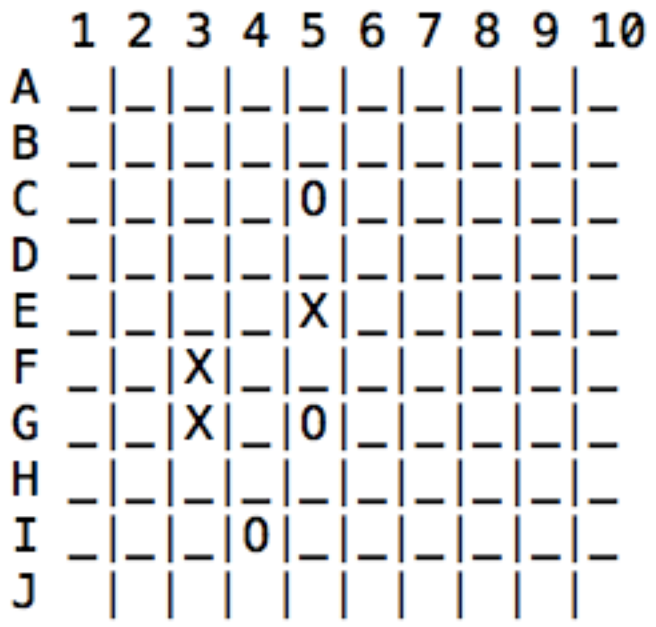


```
   1 2 3 4 5 6 7 8 9 10
A _|_|_|_|_|_|_|_|_|_
B _|_|_|_|_|_|_|_|_|_
C _|_|_|_|_|_|_|_|_|_
D _|_|_|_|_|_|_|_|_|_
E _|_|_|_|_|_|_|_|_|_
F _|_|_|_|_|_|_|_|_|_
G _|_|_|_|_|_|_|_|_|_
H _|_|_|_|_|_|_|_|_|_
I _|_|_|_|_|_|_|_|_|_
J  | | | | | | | | ||
```

In the single player version of the game, the player will enter the locations of the ships he wishes to place on the board. There are 5 ships to be placed, 1 large ship which takes over 5 boxes of the board, 2 medium ships that each take up 3 boxes of the board, and 2 small ships which also each take up 2 boxes of the board.
Once this is done, a board is shown (with the ships hidden) and the player has to make moves, specifying the coordinates of the board that he wishes to shoot at. When he shoots, that part of the

board gets uncovered. If the box was empty, a 'X' is shown. If a part of ship was in the box a 'O' is shown. Remember: The complete ship will not be uncovered, only the box shot will be uncovered. The user enters inputs in the following format —> e.g. A5, D2, C0 etc.
(The letter denotes the row number and the number denotes the column number. The top left box will be A0 and the bottom right box will be J7)

Your board should be displayed in the following format during the game.



```
  1 2 3 4 5 6 7 8 9 10
A _|_|_|_|_|_|_|_|_|_
B _|_|_|_|_|_|_|_|_|_
C _|_|_|_|O|_|_|_|_|_
D _|_|_|_|_|_|_|_|_|_
E _|_|_|_|X|_|_|_|_|_
F _|_|X|_|_|_|_|_|_|_
G _|_|X|_|O|_|_|_|_|_
H _|_|_|_|_|_|_|_|_|_
I _|_|_|O|_|_|_|_|_|_
J  | | | | | | | | |
```

Classes:

BattleBoard
BattleShip - You do not need to make changes to the BattleShip class.
SmallShip extends BattleShip // this type of ship covers 1 box (length = 2)
MediumShip extends BattleShip // this type of ship covers 2 boxes (length = 3)
LargeShip extends BattleShip // this type of ship covers 3 boxes (length = 5)

```
BattleShip {
        int length;
        boolean orientation; // false means horizontal, true means vertical
        BattleShip(boolean orientation) {
                // this.orientation refers to the member variable, orientation
                // orientation refers to the argument of this constructor, orientation.
                this.orientation = orientation;
        }
}
```

```
BattleBoard{
        int[][] board; // make 10x10 board
        // 1 signifies a empty slot that has been shot.
        // 2 signifies a ship.
        // 3 signifies a ship that has been shot down.


        String toString() {
                // this will print the board as much as the player can see,
                // it will only show the boxes which have been shot by the player
                // but will not reveal the hidden ships
        }

        String printActualBoard() {
                // this will print the actual board, containing all the ships
                // as were placed by the player initially
        }

        boolean addShip(BattleShip x, int x_cor, int y_cor) {
                // takes a ship.
                // You may need to cast your, SmallShip, MediumShip or LargeShip to BattleShip
before passing it as an argument.
                // x_cor, y_cor signify a starting coordinate.
                // should return true if ship has successfully been deployed.
                // should return false if the ship cannot be deployed if it overlaps with another ship.
        }

        int shoot(int x_cor, int y_cor) {
                // shoot on particular coordinates.
                // should return 0 if the particular coordinate has already been shot.
                // should return 1 if  shot at a empty place.
                // should return 2 if shot at a ship coordinate.

        }

        boolean isLost() { // to check if all the ships have been destroyed or not.

        }
}
```

Part 2:

Two-Player Version

You are to implement a two player version of the game which works over TCP/IP.
One of the users will start the game as a Server and the other user will join as the Client, by using the server's IP and Port.
Both players will get to plant 5 ships (1 large, 2 medium, 2 small) on their own board and when they are done, the game will send a message to the other game saying that the player is ready.
Once both players are ready, the game begins.
When each player has made his own board, the board must NOT be sent to the other player. The other player will only see a blank board. After every move, the move will be sent to the other

player's game, it will be evaluated whether it hit a ship or not, and a corresponding reply would be sent. This would be simultaneously happening for both players.
The player who is able to destroy the other person's ships in lesser number of moves wins.
When either one of the players wins the game, a message should be sent to the other player and the game should be over at that point.