Given an 2D board, count how many different battleships are in it. The battleships are represented with 'X's, empty slots are represented with '.'s. You may assume the following rules:

* You receive a valid board, made of only battleships or empty slots.
* Battleships can only be placed horizontally or vertically. In other words, they can only be made of the shape 1xN (1 row, N columns) or Nx1 (N rows, 1 column), where N can be of any size.
* At least one horizontal or vertical cell separates between two battleships - there are no adjacent battleships.

Key:

The key to the solution for this question is to utilize the idea of dynamic programming, making decision on each grid of the board basing on the grid above and grid to the left, if doing iteration from top left corner to bottom right corner. Exceptions are the first row and first column

This translates into:

First row: an ‘X’ is a new battleship if and only if the grid to the left is not an ‘X’

First column: an ‘X’ is a new battleship if and only if the grid to the top is not an ‘X’

Other: An ‘X’ is a new battleship if and only if the grid to its left is not an ‘X’ and the grid to its top is not an ‘X’

Counting increase one in one of the cases above.

Solution:

public class Solution {

public int countBattleships(char[][] board) {

int colNum = board[0].length;

int rowNum = board.length;

if(colNum==0 || rowNum ==0 ) return 0;

int number=0;

if(board[0][0]=='X') number++;

for(int col=1; col<colNum; col++){

if(board[0][col]=='X' && board[0][col-1]!='X') number++;

}

for(int row=1; row<rowNum; row++){

if(board[row][0]=='X' && board[row-1][0]!='X') number++;

}

for(int row=1; row< rowNum; row++){

for(int col=1; col< colNum; col++){

if(board[row][col]=='X' && board[row][col-1]!='X' && board[row-1][col]!='X') number++;

}

}

return number;

}

}