Mine Sweeper

Write a Java program that simulates the famous game Mine Sweeper. If you do not know what a mine sweeper is, search it on google (I am pretty sure you have seen it before, at least).

The rules are simple. A field of certain number of mines is generated by the computer and it reveals an empty board to the user. The user has to choose the spot he/she wants to reveal. If the chosen spot contains a mine, the game is over and the user loses the game. If the chosen spot does not contain a mine, then it will display the number of mines surrounding it (up to eight). Once the user has revealed everything except the spots that contain mines, the user wins the game.

Your job is to implement this game by implementing the methods that are left empty. I have filled some of the methods, don't change them. If you are confused with the program, ask me.

Create a class called MineSweeper in Eclipse, and copy the code that is in the txt(attached with the mail) into the file you just created.

Now, I will explain each method you need to implement and the methods I have already implemented.

MineSweeper.java

This file will contains three instance variables and eight instance methods. I have completed two of methods. You need to use the three variables to complete the remaining six methods.

REMEMBER TO CHECK THE SAMPLE RUN AT THE END.

The three instance variables I have written are:

size

This is going to be the size of the board. In the method `initialize()`, You will ask the user to enter an integer and store it in here.

numOfMines

This is going to be the number of mines on the board. In the method `initialize()`, you will ask the user to enter an integer and store it in here.

kb

This is the scanner that you will use to prompt the user input. If you forgot how to use it, check the previous project. If you have further questions about his, ask me. Note that I have already initialized it, so you will just use it directly.

Here are the ones you need to implement:

public static void initialize()

This method will asks for the user input to store the size of the game board(for convenience, the game board will just be a square) and the number of mines that will be on the board. If the number of mines is bigger than the size of the board(size*size), we set the number of mines to be the square of size.

public static boolean inBound(int r, int c)

This method checks if the row number and column number is valid. If they are negative or if they exceed the bound of the board, it returns false. If they are within the board, it should return true.

public static void setMines(int[] [] board)

This method will set up the mines on the board. You will need to use `Math.random()` to generate two random decimal numbers within **[0, 1) (including 0, excluding 1)** to be the row index and column index of the spot that will have the mine. You should change that spot to **-1** to represent the mine. If the spot already has a mine, it should generate new indexes. Note that the number you

generated is type **double** and you need to cast it to **int** to have it work. If you forgot the techniques of using `Math.random()`, check the notes you had last year. If you cannot find it, ask me. The program should keep generating the row index and column index until the number of mines reaches the amount specified by `numOfMines`.

public static void setValues(int[] [] board)

This method will set the values for each board. The value should represent the number of mines surrounding it.

Here is one example (X represents mines, note that in the program, mines are represented by the number **-1**, when displaying, -1 will be converted to X for the convenience of display. You don't need to worry about converting it since I have already done the job in `displayBoard()`)

```
+---+---+---+---+---+
| X | 5 | X | X | X |
+---+---+---+---+---+
| X | X | X | X | X |
+---+---+---+---+---+
| 4 | 7 | X | X | X |
+---+---+---+---+---+
| X | X | X | X | 5 |
+---+---+---+---+---+
| X | X | 5 | X | X |
+---+---+---+---+---+
```

As you can see, the value 7 is surrounded by 7 mines and the value 4 is surrounded by only four mines.

`public static boolean getUserMove(int[][] board, boolean[][] revealed)`

In this method, you will need to ask the user for two consecutive inputs. The first integer will be stored as the row index and the second integer will be stored as the column index. You will need to check if the spot is inbound and whether the spot has been revealed or not, if so, you will ask the user to enter the numbers again until the spot is in the boundary and the spot has not been revealed yet, you record that the spot now has been revealed. Then you need to check whether or not the spot contains a mine, if the spot has the mine, you return true to represent you have found a mine. Otherwise return false to represent no mine has been stepped on.

`public static void revealEverything(boolean[][] revealed)`

This method is simple. You just need to mark that everything has been revealed. This is for the purpose of final display of the board to show everything on the board to the user.

Here are the ones I have already implemented. Do not modify them!

public static void main(String args[])

This is the main loop of the game. It calls the methods that you wrote to run the game.

public static void displayBoard(int[][] board, int[][] revealed)

This method will display the board in a good format(in my opinion xD). If the spot has not been revealed, the space will be displayed at that spot. The mines(represented by number -1) will be displayed as X.

Sample Run

(Yours may not look exactly the same since the mines are at different positions each run. Again, user inputs are underlined)

```
Please enter the board size: 5

Please enter the number of mines: 6

     1   2   3   4   5
   +---+---+---+---+---+
 1 |   |   |   |   |   |
   +---+---+---+---+---+
 2 |   |   |   |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+
 5 |   |   |   |   |   |
   +---+---+---+---+---+
Which square do you want to check: 1 1

     1   2   3   4   5
   +---+---+---+---+---+
 1 | 0 |   |   |   |   |
   +---+---+---+---+---+
 2 |   |   |   |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+
 5 |   |   |   |   |   |
   +---+---+---+---+---+
Which square do you want to check: 1 4

     1   2   3   4   5
   +---+---+---+---+---+
 1 | 0 |   |   | 2 |   |
   +---+---+---+---+---+
```

```
 2 |   |   |   |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+
 5 |   |   |   |   |   |
   +---+---+---+---+---+
Which square do you want to check: 5 5

     1   2   3   4   5
   +---+---+---+---+---+
 1 | 0 |   |   | 2 |   |
   +---+---+---+---+---+
 2 |   |   |   |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+
 5 |   |   |   |   | 0 |
   +---+---+---+---+---+
Which square do you want to check: 6 6
Invalid square. Try again!

Which square do you want to check: -4 4
Invalid square. Try again!

Which square do you want to check: 0 0
Invalid square. Try again!

Which square do you want to check: 1 1
Invalid square. Try again!

Which square do you want to check: 2 3

     1   2   3   4   5
   +---+---+---+---+---+
 1 | 0 |   |   | 2 |   |
   +---+---+---+---+---+
 2 |   |   | 4 |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
```

```
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+
 5 |   |   |   |   | 0 |
   +---+---+---+---+---+
Which square do you want to check: 2 2

     1   2   3   4   5
   +---+---+---+---+---+
 1 | 0 |   |   | 2 |   |
   +---+---+---+---+---+
 2 |   | 2 | 4 |   |   |
   +---+---+---+---+---+
 3 |   |   |   |   |   |
   +---+---+---+---+---+
 4 |   |   |   |   |   |
   +---+---+---+---+---+
 5 |   |   |   |   | 0 |
   +---+---+---+---+---+
Which square do you want to check: 1 4
Invalid square. Try again!

Which square do you want to check: 1 3

     1   2   3   4   5
   +---+---+---+---+---+
 1 | 0 | 1 | X | 2 | 1 |
   +---+---+---+---+---+
 2 | 1 | 2 | 4 | X | 2 |
   +---+---+---+---+---+
 3 | 2 | X | 4 | X | 2 |
   +---+---+---+---+---+
 4 | 2 | X | 4 | 2 | 1 |
   +---+---+---+---+---+
 5 | 1 | 2 | X | 1 | 0 |
   +---+---+---+---+---+
*****************************************
* BOOM! You stepped on a mine! Game Over! *
*****************************************
```