MINESWEEPER JAVA:

In this code, I have predefined the dimension of the grid with integer values. The

number and location of the mines are randomly chosen. There could be multiple classes to create

the game. Each class will handle various methods. There will be 2d arrays for the game. Loops

will be used to create the grid and to place mines. The code will need to be checked for compile

time and timeout error.

“.” and “\*” are used to determine the safe zone and the bombs respectively. The players will

need to determine their safe zone with the hint numbers which will be provided on the grid.

These clues are calculated with method in the program.

I have a mentioned a few test cases, much more is needed, for the codes. I recommend using the

red, green, refactor process for the testing- in that you fail, you pass and you try to refactor your

codes for a more effective code.

In this game, my thought process was to create a program in which, I will place safe zones first.

Then, I will place the bombs, making sure that a change of dimensions will not throw the bombs

out of scope. I would like to have the number of bombs calculated depending on the dimension

of the grid- that is, not having 5 bombs on a 2x2 grid. Then, I would make sure the bombs are

replacing the safe zones and not increasing the boundaries of the grid- in other words with an

increase in the number of bombs the grid should not grow. In fact, there will be an error if the

code tries to accommodate all the bombs. Therefore, I would like to have a formula for the

number of bombs in a grid with specific dimensions.

The code should be able to calculate the values of the adjacent cells which may or may not have

any bombs. With the help of the hint values, the player should be able to plan his next move.

MINESWEEPER GAME TESTING:

The goal of the game is to locate mines. I have created a **M X N**, two-dimensional grid, that has

random number of mines. The hint values in the grid are used to assist the player in locating the

mines.

I have mentioned 5 test cases to assess all possible adjacent cells which could potentially hold a

mine. Not all tests mentioned below are created. The purpose of this test is to check and inspect

as many cases as possible. Test cases test from counting the number of bombs to testing the code

itself with methods, loops and classes. Multiple classes must have individual tests.

1. Test 1- should say 0 bombs - the grid is (0x0) – which means there is no grid and hence no bomb.
2. Test 2 - test if a grid prints with the parameters given. Ex: ensure a 2x2 grid prints out 2 rows and 2 columns with no bombs.
3. Test 3 - should say 1 bomb with the test parameters (1x1)- that is just 1 cell and will hold 1 bomb.
4. Test 4 - should say 1 bomb when it’s a 1x2 grid. If both cells have bombs the game will be over and the player will not be able to win.
5. Test 5 - test that in a (2x2) grid, the location of the bombs will always be adjacent.

There will be many more tests needed if the code was completed. To complete the code, more

methods need to be created and implemented. The methods should contain Boolean values to

check for bombs. The test for that will be to increment the count variable if the Boolean value

returns true.

There will also be test cases which should pick the center cell of the grid and test the possible 8

adjacent cells which are top, top right diagonal, right, bottom right diagonal, bottom, bottom left

diagonal, left, and top left diagonal.