TCSS 143
Programming Assignment 2

Due: **See Canvas** *(by 11:55pm submitted electronically*).

NOTE: Be sure to adhere to the **University's Policy on Academic Integrity** as discussed in class. Programming assignments are to be written individually and submitted programs must be the result of your **own** efforts. Any suspicion of academic integrity violation will be dealt with accordingly

Purpose: This assignment is designed to demonstrate your understanding of basic Java Concepts

Program 2 is worth 100 points. The points will be assigned according to the following scheme:

Program compiles	20 Points
Documentation	
Java Docs	5 Points
Indentation, Whitespace	5 Points
Hierarchy Diagram	**5 Points**
Process	
Objects	35 Points
- Fields	
- Constructors	
- Methods	
Two-Dimensional Array	10 Points
Random	5 Points
Output	
Correctly printed output (Console o/p)	15 Points
Total	**100 Points**

Assignment:

The purpose of this programming project is to demonstrate understanding of 2 dimensional arrays and the creation, implementation, and use of a simple class.

You are to design a class "`SweeperGame.java`" which represents a piece of buried treasure in the ground that must be found on a grid representing the ground. During the instantiation (or creation) of an object of this class, each space in the grid will be initialized as empty (you will actually use a space ' ' character for this purpose) and a random spot on the grid will be chosen for the location of the buried treasure. For this purpose, the character "T" will be placed in this random location. Further details on the constructor are listed below.

An object of this class is used as a game for the player to take a turn to guess where the treasure is buried until the treasure has been found. The instantiation of this object and the search will be performed by a "driver" program which has been provided to you allowing you to only have to concentrate on developing the class (the driver program is in the file "`SimpleSweeper.java`").

DETAILS

The treasure is placed in a random location in the ground, so you will need to import the "Random" class in the `SweeperGame` class.

Name your class "`SweeperGame.java`".

The fields of you object should be private. This is to make sure you are encapsulating your data, by making sure fields are private you will ensure that the only access to those fields is through your getters or setters and are able to limit what data can be accessed.

*Fields (**MUST BE PRIVATE**):*

- A character (char) array named `gameBoard`. This represents the grid in which the treasure is hiding. The bottom and leftmost square of this grid will be (0,0).
- Two integers that will hold the X and Y position of the buried treasure in the grid, named `treasureX` and `treasureY`.
- A integer called `totalMoves` that keep tracks of how many times the uses has searched for the treasure.
- A boolean called `found` that keep tracks of whether the treasure has been found.

Constructor:

- Receives two integer parameters as the user's input for the number of height and width of the grid where the treasure is buried.
- The constructor instantiates the 2D array "`gameBoard`" as a character array with the first parameter for the height (`height`) and the second parameter as the width (`width`).
- Initialize `gameBoard`'s cells, each to contain a space ' ' (done with single quotes)
- Set `treasureX` (the x coordinate of the treasure) randomly based using the first parameter
- Set `treasureY` (the y coordinate of the treasure) randomly based using the second parameter
- Set `gameBoard[treasureX][treasureY]` to the char 'T'
- Set `totalMoves to` 0
- Set `found to` false

Methods:

- `beenSwept` receives the x coordinate and y coordinate to be searched and returns true if the space has already been searched, false otherwise.
- `treasureFound` receives the x coordinate and y coordinate to be searched and returns true if the treasure is found there, false otherwise.
- `checkOutOfBounds` receives the x coordinate and y coordinate to be searched and returns true if the values are within the array indices range, false otherwise (used to check that these indices will not cause an error when applied to the array).
- `getBoardHeight` returns the height of the board.
- `getBoardWidth` returns the width of the board.
- `getTotalMoves` returns the total amount of moves taken.
- `digSand` receives the x coordinate and y coordinate to be searched returns true if the

treasure is found, false otherwise and sets the `found` field to true. If the treasure is NOT found `digSand` also sets the `gameBoard` array at the received x coordinate and y coordinate location to the "Manhattan distance" to the treasure. Increment the number of moves taken if the treasure has not been found and the space has not been previously searched.

- `toString` displays the current `gameBoard` array and it's contents EXCEPT the location of the treasure which remains hidden until he/she is found at which point toString will be called (by the driver) and the 'T' will be displayed.

  NOW, and perhaps the awkward portion of the toString output. Normally, when displaying a 2D array, the [0][0] cell is displayed in the upper left corner as with matrices. However, we will be displaying the array like a regular mathematical graph so the bottom left coordinates will be (0,0) and increase as we go up or to the right. Your output should look the same as what is seen on the next page in the sample run.

Manhattan Distance

Manhattan distance is described as from *wikitionary*:

Manhattan distance (*plural* **Manhattan distances**)

1. The distance between two points in a grid based on a strictly horizontal and/or vertical path (that is, along the grid lines), as opposed to the diagonal or "as the crow flies" distance. The Manhattan distance is the simple sum of the horizontal and vertical components, whereas the diagonal distance might be computed by applying the Pythagorean theorem.

Here is a sample output to better understand what the Manhattan distance is. It is simply calculated by finding the vertical and horizontal distance to the treasure.

```
,~,~,~,~,
;5:4;3:2;
`~`~`~`~`~`
;4:3;2:1;
`~`~`~`~`~`
;3:2;1:T;
`~`~`~`~`~`
;4:3;2:1;
`~`~`~`~`~`
```

Sample Output:

```
Sweep the sand and find the treasure
Please enter the number of height and width
of the game board
Height of board: 5
Width of board: 5
Enter treasure coordinates separated by a space: 0 0
,~,~,~,~,~,
;.:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;5:.;.:.;.:
`~`~`~`~`~`

Enter treasure coordinates separated by a space: 4 0
,~,~,~,~,~,
;7:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;5:.;.:.;.:
`~`~`~`~`~`

Enter treasure coordinates separated by a space: 2 2
,~,~,~,~,~,
;7:.;.:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;.:.;3:.;.:
`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`
;5:.;.:.;.:
`~`~`~`~`~`
```

```
Enter treasure coordinates separated by a space: 3 3
,~,~,~,~,~,
;7:.;.:.;.:
`~`~`~`~`~`~`
;.:.;.:3;.:
`~`~`~`~`~`~`
;.:.;3:.;.:
`~`~`~`~`~`~`
;.:.;.:.;.:
`~`~`~`~`~`~`
;5:.;.:.;.:
`~`~`~`~`~`~`

Enter treasure coordinates separated by a space: 1 4
,~,~,~,~,~,
;7:.;.:.;.:
`~`~`~`~`~`~`
;.:.;.:3;.:
`~`~`~`~`~`~`
;.:.;3:.;.:
`~`~`~`~`~`~`
;.:.;.:.;T:
`~`~`~`~`~`~`
;5:.;.:.;.:
`~`~`~`~`~`~`

You found the treasure in 5 moves!
Out of 25 possible moves.
Would you like to find more treasure [Y/N]? N
```

How to submit your assignment:

1. Zip and upload your Java files on Canvas.
    - Please make to also include the hierarchy diagram, this can be generated by jGrasp
2. There will be a submission link for Assignment 2.