CS 251 Intermediate Programming
Lab 9: Hexagonal Minesweeper – Part 1: Mine Manager

Brooke Chenoweth

Fall 2018

In this lab, you will begin work on a minesweeper[1] game played on a hexagonal grid. You will be working on internal logic and bookkeeping independent of the game GUI which you will write in the next assignment.

Problem Specification

You are going to create a `HexMineManager` class to handle your game bookkeeping. You will demonstrate its functionality in a `TestHexMines` class.[2] The HexMineManager should be independent of any future GUI code.

`HexMineManager`

Although the actual game will likely have a fixed board size and number of mines (at least for each difficulty level), your class should be able to handle any positive integer values (within reason. Obviously, you can't have more mines than you have cells on the board).

You will need to identify locations in your game board, but it is up to you what coordinate system you use. I recommend reading `https://www.redblobgames.com/grids/hexagons/` before you try to reinvent the wheel.

Your class must provide the following functionality.

- Construct a new manager of some specified size.

  You may either create a roughly rectangular grid of with specified number of rows and columns or create a hexagon with a specified width or radius.[3]

- Place some specified number of mines on the board.

  This might be done as part of the constructor, or you might choose to have a separate method to place the mines.

[1] `https://en.wikipedia.org/wiki/Minesweeper_(video_game)`

[2] You may want some additional classes/interfaces to accomplish this, but I leave that design up to you.

[3] Or create multiple constructors and support both if you are feeling ambitious.

  - The final game will use random mine placement, but for this test program, you should have reproducible results each time it is run.
  - One simple way to have "random" behaviour that can be repeated is to use a `Random` object that has been initialized with a fixed seed.

- Override `toString` to provide a string representation of your board. Cells are packed in a hex grid rather than a square one, so your string representation should take that into account and indicate the offset rows.

- Be able to toggle a flag on a specified covered cell so the user will be able to mark possible mines locations.

- Uncover a specified covered (and unflagged) cell.
  - If cell has a mine, uncover all unmined cells.
  - If cell doesn't have a mine, reveal number of mined neighbors.

    If cell didn't have any mined neighbors, uncover all the neighbors and keep uncovering any of them that also have no adjacent mines.

`TestHexMines`

The purpose of this testing class is to demonstrate to us that your mine manager works correctly, so you need to put it through its paces.

- You should create at least two `HexMineManager` objects with separate sizes and number of mines. Demonstrate that they operate independently of each other. (I want to make sure you aren't locked into just one configuration and that you aren't making things static inappropriately.)

- Test manipulating cells on each manager.
  - Print out the string representation of the manager before you begin and after each change to the manager.
    - Demonstrate uncovering a cell, revealing number of mined neighbors. Make sure uncovering continues if there were no adjacent mines.
    - Demonstrate toggling a flag on a covered cell.
    - Repeat a few more tests with other locations.
  - After demonstrating each manager, repeat the tests with the first manager again to prove that it is independent of the second.

- Don't just copy and paste the same test code for each manager. Structure your code sensibly with methods rather than writing one giant `main` method.

- Your testing output should come from `TestHexMines`, not `HexMineManager`. Don't put console output in the middle of your mine manager logic.

I am providing you with example output from my testing code to give you an idea what I'm expecting this to look like.

Turning in your assignment

Submit your `HexMineManager.java`, `TestHexMines.java`, and any other source files you used for the project to UNM Learnable.