MineSweeper Outline / Planner

Flow of the program

* Run the program
* A file opens prompting the user for game input
  + Difficulty
  + Guaranteed first move (?)
  + Whatever else is required of the user
* Instead of a file it could be anything based on the GUI that is incorporated into the system. Files would be for textFileGUI, whereas some other GUI might use a different method of prompting the user for information.
* Perhaps best to start off simple, then move into an external file, which will then be changed to use everything.
* Info is put into an object (ADT) and is passed into a minesweeper instance
  + The instance will return an object of the same type, with different settings
* Update score file (if applicable)
* Either close or repeat

Things needed:

MineSweeperRunner

* Controls the flow of the game
* Prompts the user for info
  + Might want to do one question at a time? Or all in one file. Which would be better? Might want to test it out later (as in having a file with one question pop up, be answered, then closed, allows for more modularity when asking for prompts and is less intimidating for the user – also might skip redundancies)
* Requires an object to hold all the data that will be used for the game
* Updates a score file (perhaps scores will be shown in the prompt file?)

MSData

* ADT that will be used by MineSweeperRunner and Instance
* Will hold lots of variables I haven’t thought of yet
* Things MSData needs to have
  + Easy / Medium / Hard / Custom
  + Length
  + Height
  + Bombs
  + Playing Again (defaults to true)
* Could include something in reference to how the **graphical user interface** will be seen. The interface could be included in the initial run process, so the file which is to be included could be present there… Something like a plug in. So say, for example, that the program was run with a parameter set as “FileGUI.java” which would use that as its interface. All other interfaces would have to be called under \*GUI.java and would have to be an extension of an interface so they all have the same structure. I have no idea how this would work – so it will require a lot of testing and research. Try to get it working though, as it would be a pain in the ass to change it.
  + To start it, I need to create an interface with the proper methods
  + I would need to look into how to import something based on the parameters given in the initial call of the program, and how all that would work… that could be complicated.
  + As a simple way to work around this, still implement the GUI interface and create multiple GUI files, but instead of having them be interchangeable based on input during initial running, the names of the files can just be changed. KEEP EVERYTHING AS INTERNAL TO THE FILES AS POSSIBLE!!!

MineSweeperInstance

* Runs MineSweeper
* Takes in a MSData class
* Creates an instance of the Map ADT
* Creates an instance of the MSData class, intended to be returned.
* Has an array of CellHold, which it passes to the GUI currently being used.
* Takes the point that the GUI returned and updates the native array.
* Based on the results of the update, it either calls the GUI again or it ends.
* If it ends make a call to the GUI, which will have some end screen default, with a prompt for play again (maybe even have some way to change the GUI? Probably not that’s real hard).

MSMap

* Constructor will take in 3 integers: x, y, and bombs.
* Might take in some other shit.
* Is held through-out the game, and is called to check what exists where and is to show the values to be sent to the file – as in it is used to hold data.
* Uses a 2 d array, which will be of CellHold type.
* ADT that will create a map for minesweeper – meaning its bomb and number locations. It stores them in itself as object type CellHold – which will have either Bomb, Number, Blank set to true – or have -1 be a bomb, 0 be a blank and 1-8 be the number
* Maybe cell hold can take in a parameter that affects how the info will be stored.
* To do this, we will create an array that is x+1 by y+1 large
  + After that we will randomly generate the bombs, demanding that they not be repeats of previously filled spaces
  + These spots will be saved in an array of points
  + Once we do that, we will go through each point and increment all 8 spots around it by 1
  + Copy the x y space, so excluding the outside ring, into an array that is part of the ADT
  + Alternatively, as each bomb is placed, the space around it is incremented, might be buggy – would have to make it so it ignores other bombs efficiently.
  + Alternatively, every time we update a valid square (kind of tricky) we update the native ADT array also.
* Might be other ways to do this, will write any other ideas that come to mind.
* Make sure to include a few test files to pass, try to include bomb layouts that are:
  + Bombs being parallel
  + Bombs being perpendicular
  + Bombs being spread out
  + Having all numbers from 1-8
* Include a test function that will read in a file and check to see if they are the same.
* This is a skeleton, so ALL interfaces will have to go through this – make sure it is concrete.

CellHold

* Object used in MSMap, will have a variable called “CellContents” being -1 if it’s a bomb, 0, if it’s empty, and 1-8 if it’s a number.
* Constructors include either initializing everything to 0 and is true, or it will copy an existing CellContents, and be set to false.
* CellContents itself might have a variable in it to make it so that if it’s false (as in unvisited) it shows empty, if it is true it will show whatever value it is storing. Default for map is true, default for everything else should be false
* Includes a function called convertToChar which will convert these numbers into their corresponding character values (“b” or “ “ or “1-9”).
* For now it just needs to act as an int. must include getter and setter functions, since im not sure how its innards will change.

GUI interface

* The general standard for any gui to be added to the program
* Will have to be used to:
  + Show the user the game information they are to interact with
  + Record the actions the user inputs to the game interface
  + Verify that nothing has been violated
  + Return the actions to the game engine
* All gui will need to take in an array of CellHold, which will be used to generate the interface shown to the user
* All GUI will need to return an array of CellHold, which will be used to update the game, or just the coordinate of the update point.
* The GUI will only be the face of the game, it is used to encapsulate the rest of the process.
* All the GUI is responsible for is collecting the update point from the user and verifying that it is properly updated.
* Documentation for each GUI will be included in separate word files, which will be inside their own file.
* The GUI should include instructions in the initial setup phase