TextFileGUI documentation

The first implementation of a GUI for MineSweeper. It will follow the same sequence of steps as the original MineSweeper did, just more specific and air tight.

Initial setup:

Open a text file asking the user for some input values for the game to start. Might be a staggered approach, as to verify that each input is correct, and be clearer in the case of an error. Prompts the user for easy, medium, hard or custom. If custom it prompts the user for length, width and bomb count.

Values are returned as an MSData object.

Opens a text file for the user, it will initially be created for the first time. Input from the user will be taken in, in the form of an “X” for click here, a “!” for bomb marking, or a “?” to undo a bomb marking.

If an “X” is selected, that is the location of the update. If a “!” is selected, then that is going to be updated as a bomb spot, and the bomb count will decrease. If a “?” is placed, it must be after a “!”, and that spot will go back to a blank spot after updating.

The file is constantly checked for updates (the user has to save it to make it update), and when an update is found, it closes it and runs it through the user error tester

First it checks length of every line. If any are greater than 1 + the previous arrays length, then it resends it back.

If more the one line are 1 + the previous arrays length, then it resends it back as well

To do error handling AND to get hold of the location of the X spot we will use regular expressions.

The regular expression to use is:

(\[[ ?!1-8][Xx?!]?\])+

This will be run on every line in the file, it will use REVIEW MATCH SYNTAX FOR JAVA.

Intention is to check that length is equal to the line being checked. If they are different then it is an error.

There will be a second group that will contain the [.X], where . can be ? ! or a white space.

The location of the X group, subtracted by 1 and divided by 3 will give the x coordinate of the action location, and the line number will give the y spacing (test this to be sure).

Go through each line, checking to see if an x is found. Once an x is found either

-> Continue looking for errors / repeating x’s

-> Return the x,y coordinates of the located action spot

Methods required:

promptGameData

takes in nothing

returns an MSData object

2 different ways of working

Way 1 (to be implemented first):

Open a file with all the request data input

Easy? Medium? Hard? Custom?

If custom – length width bomb count

Maybe have some high score stuff?

To collect data just check specific lines,

When the file changes sizes, that’s when the file will be taken in (as in it was saved)

To collect data, either use regex expression (to be developed) or collect it from specific lines

Regex is probably more reliable

Way 2:

Have prompts open, asking individual commands. If the user makes an error, open the file again with an error message on it.

Where the data is depends on the file, regex could be used to check it.

Error message will be at the top of the file, before it is opened.

Could ALSO have a score board (just data from another file, which would be something to pass in… or maybe just a generic file name? probably a generic file name is best.)

promptUserInput

takes in an array of CellHold

returns a point object (the x, y coordinates)

the pui will receive an array of CellHold that is the current state of the game, it will then put this data into a file, with brackets and whatnot, and prompt the user to either write an X a ? or a !