Liam Walsh Computer Science Final Project

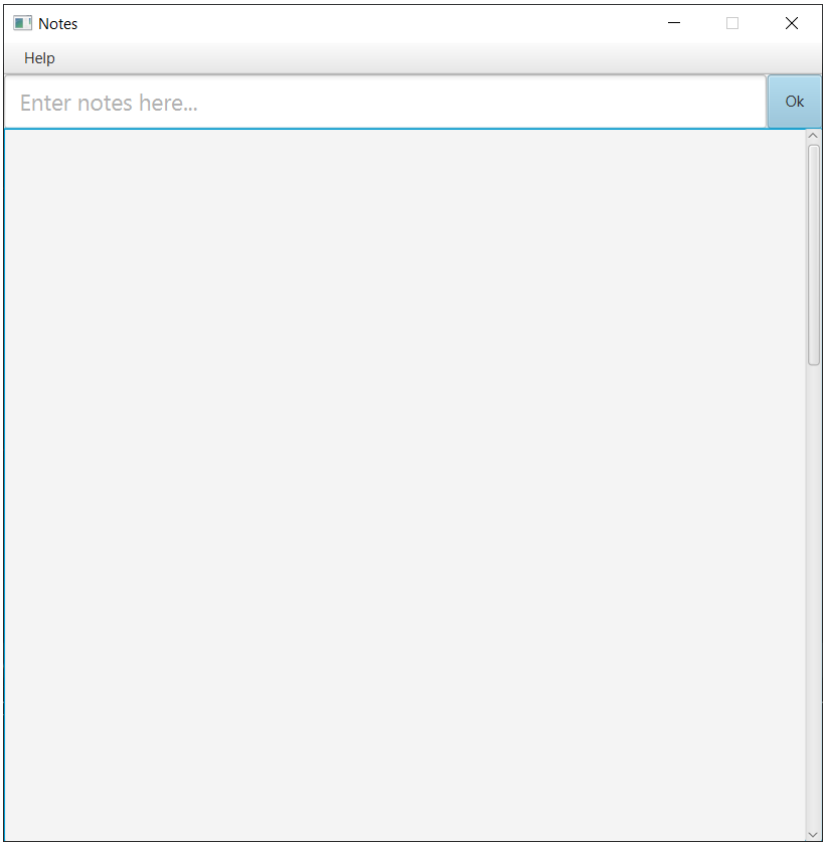
For my final project I created a sticky note application. It allows the user to quickly type out a note and save it for viewing later. The motivation behind this project was a website by google called Google Keep. Google Keep allows you to keep virtual sticky notes and save them to your google account. I used this service a lot in high school to keep track of homework assignments and such, and I wanted to make a standalone version that would run on my computer instead of a google server.

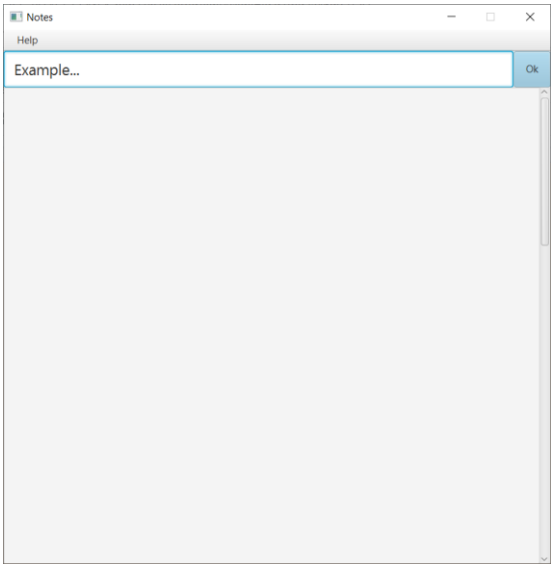
Upon the program starting up, it loads any previously saved notes from a text document. From there the user can then create more notes by typing their thoughts into the text box and hitting enter, or the "OK" button. The user can also delete any of their notes by clicking on them and hitting delete. From here the program will reorganize the notes to get rid of the gap left by the deleted note. At any point if the app is closed, all the contents get saved into a text document with three semicolons after each note. For example the documents could say something like "Note 1;;;Note2;;;Note3;;;". The semicolons are there for loading the notes. The whole string gets split up into an array of strings separated by the three semicolons.

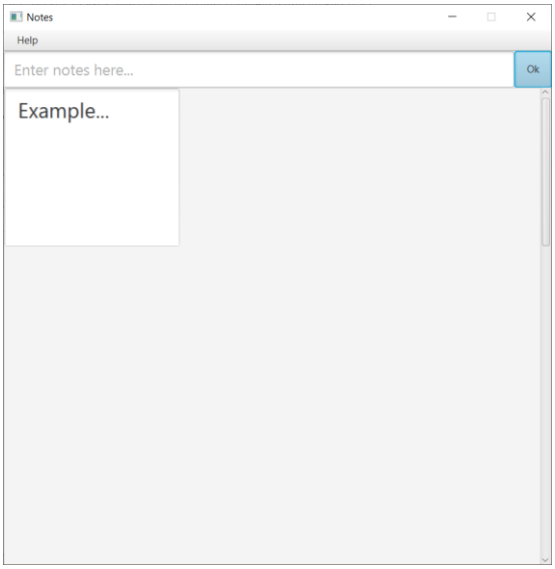
There are four classes in this project. The Main class starts up the program and loads up the UI, as well as overrides the default javafx stop() method so that I could make the program automatically save before fully closing. The Note class is for making note objects, as well as making a TextArea object. Each Note object contains variables for size, color, and text. The size and color functions do not do anything though because they were meant for functionality that I later decided was not going to be implemented. Then there is the SavingNotes class. This class contains a method for saving notes as well as for loading them via a text document. Lastly there is the Controller class. This handles user input and mostly calls upon other classes' methods when the user presses a button for example.

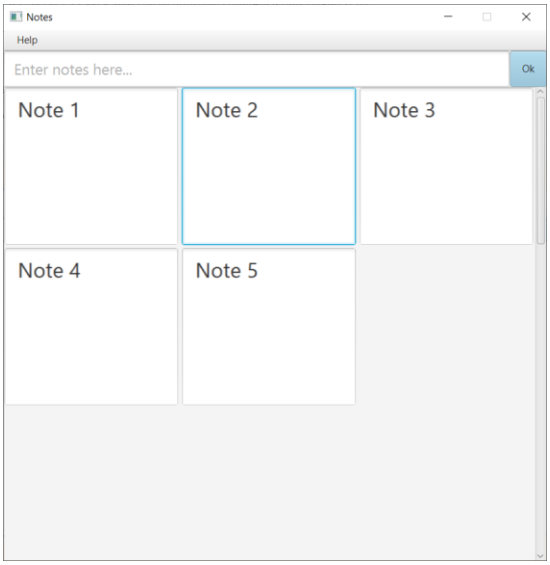
During the creation of the UI there were problems with setting up the pane that holds notes. I ended up going with a grid pane nested within a scroll pane. This way the grid can be very long so that it could potentially hold a lot of notes, and the user can scroll through the entire grid. There is still an issue that if the user inputs too many notes the grid glitches and the notes clutter the screen and obscure each other. If I were to go back and fix anything it would be this issue. I would like the grid to be infinitely expandable but in its current state I simply made it contain a lot of rows so most users would not come across the issue anyway.

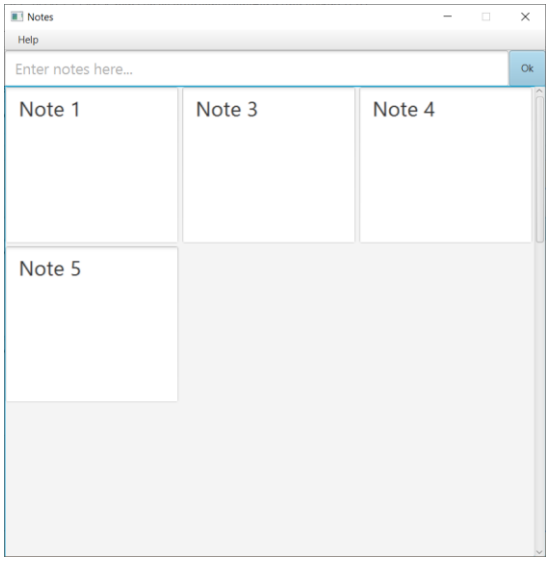
The topics covered in this project were class definition, inheritance, and generics/collections. Class definitions was covered because of the custom Note and SavingNotes classes that I made. Inheritance was covered because multiple classes in this project extend other classes in order to function and share information between them. Generics and collections were covered because Java.util classes like ArrayList and ObservableList both used generics to functions and were types of collections.


Notes
Help
Enter notes here...
Ok


Notes
Help
Example...
Ok


Notes
Help
Enter notes here...
Ok
Example...


Notes
Help
Enter notes here...
Ok
Note 1
Note 2
Note 3
Note 4
Note 5


Notes
Help
Enter notes here...
Ok
Note 1
Note 3
Note 4
Note 5