**Final Proposal**

By: Travis Dowd

Date: July 14, 2020

This project is a text editor with extra features and programs built in, similar in concept to GNU/Emacs, but this is no emacs and is not meant to be a replacement for one. Simply I have always wanted to create a real text editor, and this is my contribution to the other thousands that are out there. I have already created a simple line-based editor (influenced by the classic UNIX ed), which I had written in POSIX sh. You can find the repository for that here: [ted](https://github.com/TeaSkittle/ted). So, this project is my take on writing a GUI editor with more features and that can be actually used from day-to-day. Inspired by the naming convention used in vim (and due to the fact that this is my second text editor), I call this editor Tedi, which is an acronym for: **T**ravis' **ED**itor **I**mproved.

The target audience for this project is power users searching for a simple editor with extensibility, without the steep learning curve of more notable and traditional UNIX editors. It is designed to strike a balance between applications such as: Microsoft’s notepad, and vim. Albeit this does make the project rather niche, but it is fully functional and capable. To capture the focus of the target audience, I designed the editor to be fully extensible. I have written an extensive readme file that documents how the code editor works. The code is: simple, concise, and commented to make it very simple to edit, add and/or take away features. As long as the user understands Java and CSS, they can very easily change the editor to their liking.

There isn’t much need for another text editor out there, as there are several out there in the computing world. Although this editor does fill the rather empty space between overly simple editors and extremely complex ones. I haven’t seen an editor that does things exactly the way this one does. For example, one of the main features implemented in this project is the ability to enter shell like commands into a Java based interpreter on the bottom of the text editor’s window. Due to the time restriction of this project, I was only able to have a few commands.

To start there are two very basic commands the user can enter: print, and quit. Quit does exactly as you would expect, it successfully exits the program. The print command works like the UNIX echo command, it simply prints all following text inputted after the command. These are both simple but it is very easy to add more commands and make the editor more sophisticated. The last command implement is: calc. Calc is a reverse polish notation (also known as postfix notation) calculator that is based on a stack. With this the user can perform any calculation within the scope of Java, the syntax is just slightly altered. For example, a simple addition calculation looks as such: 6 4 + = 10, the operator comes after the numbers. This will be explained more in detail within the readme file.

There were several difficulties in facing this project. Most of them were complications with file I/O, and underestimating the scope of the project. If I avoided trying to add the commands the project would have been far simpler, but that is what makes it unique to other ones out there. Here is a list of ones that either caused bugs, errors, or headaches:

1. Opening a file into a JavaFX TextArea
2. Saving text inside of a TextArea to a file on the system
3. Redirecting standard in/out to and from a TextArea (for input the interpreter commands)
4. Creating a dark mode stylesheet for the editor (I am not the best when it comes to front end design with CSS)
5. By far the most difficult thing to implement was the stack-based calculator, taught me a lot about stacks, but also was quite difficult to get working

I decided to work on this project because I have always been intrigued by the tools used to develop software. Things such as text editors, compilers, terminal emulators, and shells always seem like wizardry until you get your hands dirty and look under the hood. I wanted a project that could be developed further in the future, and I could use this text editor to create other pieces of software as well. The strategy I used in creating this project was to break it apart into pieces (Java class files) and get one piece working before moving on to the next. I started with the calculator class, then moved onto the shell class, and brought it all together with the editor class. I hope this project seems as interesting as it was to make. Thank you.