C++ TextEditor Documentation

Written by: Nate Fanning

1. Getting Started

The text editor will be downloaded as a folder of C++ files and a makefile. To create the binary executable and run the application, open a terminal window (command prompt) in the directory for the text editor and type `make`. Make will automatically compile the C++ code for you and create an executable file called *main*.

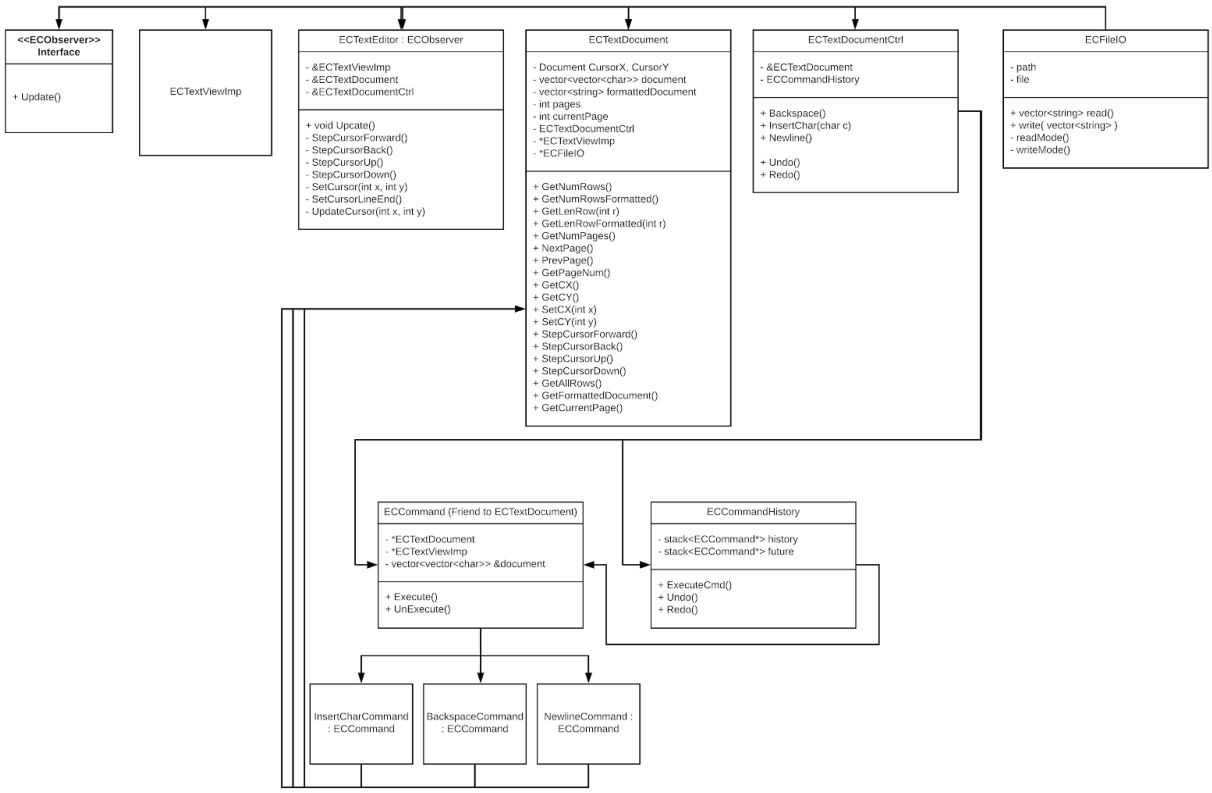
To run the application, simply run the *main* executable from the command line. Your terminal window will become your text editor!

2. High Level Objectives

Feature	**Status**	**Design Pattern**	**User Manual**
Cursor Movement	Bugs	Observer, Composite, MVC	Use the arrow keys to navigate the document. **Issues**: This was working entirely before I tried to implement pages. I had not backed up my computer since then and I forgot to push to my repository for this project, so I have lost all of that work in trying to add functionality for pages.
Insert Text / Newline / Backspace	Ok	Observer, Command, MVC	The text insertion itself works as intended. **Issues:** Text insertion relies on the cursor movement commands to move the cursor, so the cursor movement

			part of text insertion is also broken.
Undo / Redo	Ok	Observer, Command, MVC	Press ctrl-z to undo and ctrl-y to redo
File IO	Ok	Singleton	Allows opening, editing, and saving existing files, as well as creating new ones.
Text Wrapping	Ok	MVC	Works as intended
Multi Page Support	Semi functional	Observer, MVC, Factory methods	Changing pages works as intended, but as explained earlier this has completely broken support for cursor movement which most of the rest of the program relies upon. I have been trying to fix this for hours but I am out of time and have no way to roll back to my previous methods.

3. Design



This is an overall design of the text editor program

The overall design of the text editor program can be seen above. The ECTextEditor class is the main class that handles the execution of the program. The ECTextEditor, ECTextDocument, ECTextDocumentCtrl, and ECFileIO classes are all generated once as singletons in the main execution function and only one instance of each class is used in the program. As you can see in the diagram, the ECTextEditor stores references to the ECTextDocument, document controller, and other singleton classes.

The raw text is stored in a 2D vector of characters in the ECTextDocument class and editing to the document is controlled by the document controller by generating commands which allows undo / redo actions.

The document formatting for display happens in the interaction between the ECTextDocument and the ECTextEditor. The ECTextEditor asks the ECTextDocument for the formatted document. This tells the text document class to format the text depending on the window size and return the current page to the ECTextEditor. Then the editor class passes that formatted document to the text view which puts them on the display. This gets complicated in cursor handling, as there are now two cursors. The textView cursor is the position of the cursor on the screen that you can see in the textView. The document cursor is the actual position of

the cursor on the raw text document. The document cursor is used for the actual editing of text and the textView cursor is purely visual so that you can see where the editing will occur.

4. Interface

This program interfaces with the provided text view code in ECTextViewImp and the ECObserver observer interface. No other non-standard libraries or interfaces are used.

5. Environment

The program will run on any standard Unix environment and can be invoked as explained earlier by using the makefile. The standard c++ / gcc compiler will be used to compile with the c++14 standard.