**Initial Ideas**

When starting the Undo/Redo project, there was an idea to do it with an image editor instead of the typical text editor. The idea was, due to having a static image location, it would be much easier to redo an undone item without having to consider if the position of a character was going to interfere with re-insertion if, for example, you place an additional word in front of the deleted character, moving the position into the middle of a word (and not a blank space anymore). Though this idea was later discarded as actually creating an image editor would be a challenge in itself, and open-source options were outside of our abilities. Open source options that we considered were Paint.NET (which has a version open source, but is no longer open source), and OpenOffice.

Next, the idea was to make a rick text word processor path. After a bit of tinkering, trying to learn Rich Text Format, it was decided that for the sake of the project, we would stick with a simple program, focusion strictly on the undo/redo interface (keeping in mind that said interface could be implemented in a proper word processor, or even an image editor). As a result, the final project choice was a general text editor.

**Design of the Editor**

Going into the project, the first thing we decided was how we wanted to improve the undo/redo system based on our personal annoyances with modern systems. This included the inability to see the past history, not being able to undo specific parts without going through all the ones after, and of course not being able to redo stuff after modifying a document. With this in mind, we went ahead with designing the layout of the editor.

**Note**: The basic layout of the document was decided right away and wasn't changed. This included a menu at the top with your standard File and Edit items, a tool panel right underneath, a text area for the document itself, and a panel at the bottom (which in theory would include information about the state of the document). All of these are the basic elements of a common text editor, which we stuck with for the sake of learnability.

The early prototype included a panel embedded to the left of the document, which could be either hidden or showing. This panel would be used to display the undo and redo history of the document in a list form. It would be resizable, and would allow the user to select specifically what they want to undo or redo. The basic undo/redo buttons would be present in the tool panel, specifically for new users who may not be familiar with the interface and for quick undos/redos. This design was ultimately discarded as it would be difficult to implement a resizing panel using Java's Swing.

Inspired from the original design, we decided to make a popup panel which would display the undo/redo history. This was used as it would allow the user to position it on the document in a way that's convenient to them. The specifics look of the list inside went through many aesthetical changes, dealing with font choices and how the history information would be presented. Initially, there was a limit set to the amount of characters shown about the users modifications. This was later removed as it didn't give the user access to all the information they may have needed. Instead, the panel become scrollable, allowing the user to read all the text by scrolling. It is important to note that at this point, programming the actual functionality of the undo/redo system had begun and issues were immeditaly apparent. Issues such as how to keep track of the position of what's being undone/redone led to the realization that certain actions would not allow any action prior to it to be undone until it itself was undone. We wanted to be able to visually display this to the user. This was done by greying out the font of elements which could not be undone, and this design was kept in the final product.

During the programming process, we were inspired with the potential of what our undo/redo could do. One of those which is lacking in many programs today is the ability to do multiple undos/redos, and with the current inteface we had designed, we realized this could be implemented. Using standard control or shift clicking, the user could select multiple items at once, and if proper conditions were met, the user could undo all of them in a click of a button. This would increase the efficiency of the interface significantly, without much need to learn as this style of multiple selection is used by many basic programs (such as deleting multiple files). If the user selects an action which would cause ambiguity in the final product, an error is displayed. Though throughout the design process we wanted to avoid using popups to give information to the user, it was deemed alright for this situation as the amount of changes the user was making deserved a proper explanation as to why it didn't work.

Now with the ability to do multiple undos or redos (not both at once for technical reasons), another idea came to mind; that being allowing the user to preview the result of their undos/redos without immediately applying the changes (and thus forcing them to manually reverse it again if it is an undesired result). Beneath the history panel, a new button was added. When clicked, a new document-like text area (which doesn't allow text modification) pops up displaying exactly what the document would look like should the changes be applied. Beneath are two buttons (Apply and Discard). If the user is satisfied, they would select Apply, and if not, Discard (the close button of the window has the same effect). While in the preview state, no modification to the document can be done as that could lead to ambiguities.

At this point, we were satisfied with the interface we had created. We got together and pondered improvements. It was at then when we decided to implement a final interface which would allow the user save the history alongside the document. In other words, when exiting the program, the user could reload the history of their document and undo and redo what they did in their previous session. When the user saves a files, they will be asked if they want to save the history as well. The saving file interface is your standard save-file interfaces (varies by operating system). Similarly, when opening the document, if the history file for a document is found it will automatically be loaded.

There were other changes that were made not associated to the undo/redo interface (but important nonetheless for the look and feel of the program, including the undo/redo interface). Changes included the color scheme of the program, and how the buttons look. The toolbar itself went through some iteration, starting with just a undo redo button alongside the pop-out window toggle. This was changed to have more icons and much better aesthetics, highlighting the button you hover over in red, then turning green when you click, giving a clear reaction to pushing the button. There were also early issues of the toolbar not being uniform, one button being larger than the others. This was fixed, all buttons sharing a universal size, and grouped into related actions amongst each other.

There was an extensive number of error fixing by the creator, ranging from null pointer errors to the Add/remove history window failing to update, or removing the wrong String. Unfortunately certain bugs still exist.

**Bugs**

The current state of the program is far from bug-free, though it is substantially better than past versions. Before explaining the bugs, it is important to note that the undo/redo algorithms are very complex, partly because the program is interfacing with Java's Swing libraries which occasionally have some strange behaviours which are not well documented or understood. If I had the time, I would rewrite the specific components used in order to ensure desirable behaviors. That being said, here are some know bugs.

-Pasting: Though technically supported, it may cause the entire undo/redo history to go out of sync, giving undesirable behavior.

-Moving cursor: Also technically implemented, somtimes moving the cursor will cause the history to go out of sync, giving undesirable behavior

These cause of these bugs are not well known, making it difficult to fix.