Comp Sci

Amanda Nakanishi

Writeup

When I started this project, I was under the erroneous conclusion that because I had thought a lot about what I wanted to do, my project was well thought out. At the beginning of the project, I thought all I would really need to do was make a few methods and figure out how to print things to Terminal (Initial Commit). As I soon realized, that was a rather oversimplified version of everything.

The Frequency Analysis part of my code was pretty simple, well, not necessarily simple, but more straightforward than the Monoalph one. I actually think that the Frequency Analysis part was better organized than the Monoalph one, so that adds and additional layer of simplicity. Also, because it was straightforward and there weren't that many choices to be made (none, I think), I came out with a working product pretty quickly (Commit 3). In contrast, the Monoalph needed a lot more work. At first, I jumped into it, without really thinking it through because I thought it would be pretty easy to accomplish. However, I quickly realized that doing that would be stupid. So then I started pseudocoding it, and that only made me come up with more questions as opposed to clarifying my path (Code in commit 3). Then, when I actually started coding, I realized there was more questions that I had that were unanswered, and things that I thought were finished were actually unfinished. For example, I forgot to actually implement something that would sort through the arrays that held the letters and replace the message's letters with the desired letters. So then I had to go back and do that

(Commit 2/12/14). Fortunately, what I did manage to lay out did serve as the main foundation for my code in the end.

One of the concepts that I hoped to learn more about from this project was arrays. I used arrays as the base of my Monoalph programme. Reflecting on it, I think I could have probably used a two-dimensional array to do what I wanted to do just as well as a one dimensional array did it. The reason that I didn't use it though, was because I wasn't sure that using a two-dimensional array would use less space and because I wasn't exactly 100% sure of arrays, so I thought I should know arrays first. The largest problem that occurred as a result of this was that I wasn't sure how to perform certain functions. In some cases, I thought I knew how to, but in reality, I didn't and just typed out useless bits of code and then when it didn't work, I had no clue what I did wrong, or I attributed it to other things (Commit:2/14/14).

Another problem that I had was that I originally didn't set up my code so that it would fit too easily into being more object oriented. The reason that I did that was because I thought it would be easier to test the code individually if each of them were their own programme. I thought that it wouldn't be too difficult to combine them, and it wasn't. The only real problem was that I didn't really know how to change it into object oriented programming, so I was stuck and confused for a bit (Commit: 2/19/14). However, I figured it out and now I know how to set that up and can (hopefully) do it now with ease.

Once I finished figuring out how to set the frame of my programme up, I decided to start learning how to print text into the programme and saving text from the programme into another separate file. Ms. Nagoshi helped me out a lot here. She

already knew how to do those things and so she showed me how to do them, or well, at least helped me towards the answer (Commit 2/21/14). It saved me a lot of time, because before that, I was mainly looking on the internet and in my textbook for ways to do that. There were ways that I found, and they did use what I ended up using, but for me it was really confusing until I had someone there to help me with syntax and explain the general idea of what the particular things did (Record of Thinking 2). Additionally, there were some things that Ms. Nagoshi knew about that the internet didn't necessarily warn me about. For example, I wasn't aware that we had to throw a null pointer exception in the line of the called method in order for it to not error out.

By this time, once I finished learning how to read in things to my programme, I considered making it a bit more user-friendly, as really only I would know certain requirements for the programme, and certain set things, such as which part goes first. I considered doing that, and started it on a couple of occasions for different aspects and choices in the program, but I always stopped and considered whether or not I actually needed it, and in many cases I didn't need to add the code that I wrote to give options to the user. On the other hand, I could make it really obscure because only I would use it and due to the nature of the programme, stereotypically, it shouldn't be too easy to use. However, I did eventually reach a balance between making it have too many options and making it have very few (Record of Thinking 3). I think I did pretty well in this because when we were having other people test it, Zach and Caitlin had a pretty easy time using it.

One thing in my programme that I am really proud about is that there aren't very many remaining bugs. Or, at least, when Zach tested it, he couldn't get it to error out or

to come out incorrectly. So I was really happy with that, because it meant that everything was working. The way I got around replacement of letters and having them not switch places was simply by only having the 26 letter set to choose from. The methods wouldn't replace punctuation or spaces because punctuation wasn't in the letter set that gets replaced. Another thing I like about my code (though I admit it wasn't intentional) was the fact that you can just end the code in the beginning if you accidentally typed something wrong. That is to say, you can just choose an unlisted option and get out of the loop without much trouble.

One thing I am slightly worried about though is that my code has a tendency to rewrite the existing text in a document if it shares the same name. So I considered comparing the document names and if it matched any names at all, I would send a warning, but I didn't really get around to doing that. However, I came to the conclusion that if I'm stupid enough to use the same name and risk a rewrite, then I deserve what's coming to me.

Next time I would probably start by pseudocoding the whole thing first and then coding on top of it. I think I'd also be less indecisive on which features I would discard and keep. All in all though, I'm pretty content with my project.