ANTHONY CLARK

PROJECT 3 NOTES

Because this week's project is so similar in nature to the word guess project, I obviously wanted to attempt to integrate my old, already tested code, into my phrase guess program. The already tested part became a key element while I was laying out the way I wanted to design my project. Because it had already been shown to work, I wanted to have as few changes as possible the my original playHangMan function. The only changes I ended up implementing were to entirely remove the section where the user gives the word to be guessed (including all of the validation that went along with it), adding a return for when the user did not guess the word within their allotted guesses (so that I can drop them out of the game in the main function), and removed the play again function and moved that into the main function so that it would run my new code as well. Keeping the old work separate from the new helped me understand how to use functions properly – including how to best set their parameters, how to use the return of the function in different methods, and generally how crucial it is to keep things separate while writing larger pieces of code. This is an area of programming that I need to continue to develop to improve efficiency and readability of my overall program.

I set up the new code so that the number of words in the phrase was randomly determined, and so that the 12 words that I manually input were randomly selected from the array as well; this was done in my generateWord function which returned one word at a time. I then took that word and plugged it into the playHangMan function as its argument and ran it as many times as there were words in the phrase, or until the user failed to guess a word correctly. The use of an array in place of the user, or player one, entering a word made my playHangMan function far simpler as all of the validation, input, and variables that were previously used in that part of the program could be completely eliminated, and did not have to be altered to the array since you don't have to rely on user input and their potential mistakes.

One issues I ran into with this project, was finding a way to save the word so that I could show the user their phrase when the game had ended, either through their guessing each word in the phrase correctly, or by their losing the game. To compensate for this, I had to make a series of if statements that stored the word depending on which word they were currently on (through a counter that kept track of each time the program had run). This was simple enough, and to compensate for when the user did not guess all of the words in the phrase, I still generated the number of words they were supposed to have, through the generateWord function, and printed them out to let the user know what else was going to be in their phrase, had they won.

The best way I have found to test my projects is to have people in the class test them, and then once I fix any errors that they may run into, I have people outside of the class run them. Because people in the class are aware of how the game should be run, they often have certain things they want to try to break the code, but they don't have the mindset of just playing a game that I want the user to have to see how they would normally behave had they not known how it should be running. I of course test it myself too, but having these outside perspectives has been very beneficial for me, either as a confidence booster, or to point out that I have something that I need to look into correcting.