1. I prepared for this project by first considering the basic needs of the game based off of the mechanics. Despite the unusual nature of the character backstories and mental weaknesses, I reasoned that I could still make it work by having a structure that handled how the player moves through the game, how the player fights, what the enemy and player stats were, and how all of those work together. From this I made an algorithm, then from that a code skeleton that would eventually be refined into the final product.
2. I did write a code skeleton, and it was useful to a degree, though I wish I was more thorough with it initially. It was meant to be general, and it did give general guidelines, but I made a lot of assumptions about how I could proceed in the project because of it, which slowed me down later on. Although it helped me define the functioning of the program, it also mislead me about how the structure would work, which meant I had to readjust a lot of basic aspects of the program as I got into the more complex functions.
3. I could have done better by focusing on the complexities of the project earlier on. As I got closer to the deadline I was facing larger and larger problems that meant changing more of my code--whether it was a logical error or a fundamental limitation of my design. If I had discovered these issues earlier on by writing out a more in depth algorithm, I could have allocated more time to the problems that needed greater attention. I feel that my finished project is solid, but if I could have foreseen all the necessary rewrites, I definitely could have made something more polished and intricate.
4. The biggest false start for this entire project had to do with a natural assumption that all the classes I made could access any information I wanted them to from the other classes without getting much more difficult to manipulate. My initial algorithm failed to consider

that the role of a game manager class would be to handle the passing of information, so I made the assumption that the functions within the various classes would do entirely different tasks than what they actually needed to in order to make functional components that I could piece together. I thought the fight class would actually perform the damage and looping elements of the fight that are integral to the fight, but I could not do this without incorporating other class objects into this class, which I would have had to continually pass the information of into the class objects in other classes. I ended up writing most of the fight structure into the game manager, which just relied on the fight class for menus and repeated actions that were used in fights. This trend existed in my plans for all the classes, so I almost had to flip my entire project once I started working out the final product.