When I first started this project variety show was in the process: practice had been going on for nearly a month. With the independent project, I wanted to make something useful and practical, and I remembered how when those who were trying out were singing up, all the names and measurements and such were being written down on paper. I don't think it made the time significantly longer, but I thought it might be cool to have something similar to the check in screen at the health center. Maybe there just wasn't time to make one?

Before the project was announced, I had the idea of making a variety show manager. I thought about it before I slept and wondered what else I could do for the project, but when the time came to officially choose our projects I had decided that this was something I wouldn't mind spending time on. I also wanted to learn how to save stuff from DrJava or Eclipse onto text files because it's kind of sad when cool things are done in the interactions pane and things aren't saved. I guess.

That it was a major independent project made it a little intimidating, but I just started coding it in the order that I would usually use the application: open, add data, (delete and edit if I make mistakes), save, and maybe later when I open it again, search. Having it start with just opening and adding also made it seem like I was just making additions to something that worked.

But before coding, I did spend some time drafting what I'd do on paper, like the code to translate back and forth from the text files and how to sort the methods in the classes. Although once the code got a bit messy and hard to understand (with when to put fileWriter or File or other into parameters...), so I rewrote the code into Eclipse (I started with one class, and split parts later to get a better idea of connections, arguments/parameters... Record of Thinking #2). This was at an early stage and I like Eclipse a lot better than DrJava, actually. And when I started coding, magic poured out of my fingers and made something amazing.

Just kidding, I had a ton of errors. My drafts gave me a general direction and organization to work off of, but I did not pseudocode and I did not know the exact workings of some of the things, which was probably a reason why I had to rewrite the beginning code. But I like to think of that early code as a first draft, or just figuring out how file readers/writers work. At least what I had ran and allowed me to test as I go, though I'm not sure how much time this saved or wasted (questioning the "guess-and-check-ish" method of coding: Record of Thinking #2).

Among the challenges I expected were learning how to utilize text files and trying to figure out how to put a string into an array. The very first time I tried to write to and read from text files, I tried using the iCT textbook, but I couldn't really follow what was going on (later I found that this was similar to Amanda's code—she explained to me her method, which was very simple). So I found a website (http://www.mkyong.com/java/how-to-read-file-from-java-bufferedreader-example/) which I copy-pasted code from to at least get me started (this one looked more like David's). The more I coded around this, the more I understood how it worked, and that allowed me to use it specifically for my purpose. (Reading from .txt and taking code from the Internet discussion in Record of Thinking #1.)

On a side note, Makana (T.) was also doing something that involved text files. I explained my reader to him, and then Amanda explained hers, and Makana decided that Amanda's was better for his project.

With the arrays, on paper I worked on a way to translate the data into a single string and how to translate it (attached to Record of Thinking #1 on Canvas). After I started coding, however, I saw the need for more drafting because it didn't make sense while writing code. Perhaps it still needs revisions: the file does not open successfully.

And that was one of the unexpected errors. I got lots of out of bounds errors, and after stressing about it for two hours, I called it a night and left it for tomorrow. Because that tomorrow was the day that the project was due, I still have not fixed that error yet. I even asked David for help, and neither of us could figure it out. However, I still plan to because I want this to be something that could potentially be used for variety show. (Originally, it could open and retrieve data for *one* actor without errors, and this is why it was not caught during beta testing.)

Another unexpected error happened when I was trying to put String[]s into the ArrayList—I had one String[] named actors and assigned values to it, but it wasn't until later that I realized that the elements of the String[] actors was being changed whenever I wanted to move onto the next actor—the result was that I had one set of data for one actor when I should have had more. This error helped clear up a misconception for me, though: when I asked Mr. Kiang about it, I found that the String[]s didn't *need* to have names, and that I could add elements to it later. Later, I also found that as long as I added the same number of elements when creating that array that I desired, I could modify those elements later (so I added them all blank in the latest version, and assigned their values later). Knowing that all String[]s did not need a name to modify them was very, very helpful.

And another small unexpected error was that when I tried to use a nextLine() or nextInt(), the interactions pane would not allow any input. The first time it happened, I asked for help from Mr. Kiang, and we added some code before and after it and it worked. The second time, however, I didn't see a similar solution, so I tried adding the same line under it like so...

```
something = kb.nextInt();
something = kb.nextInt();
```

and it worked fine. I have no idea why it happened or why that works, but I'm glad it did (Record of Thinking #3).

At the least, I'm glad I got some experience working between .java and .txt files. This opens up new doors. I think if I do want to make this something others can use, however, I would want to use a different interface, something like Aliya's, maybe.

I tended to try to do things on my own. I might have asked Amanda a few small questions, and David more offered to help than I asked for it, and I asked only a few questions to Mr. Kiang. Maybe that was all I needed? Or maybe not, since my code still doesn't work as I intended it to.

Some things I probably would have found useful are knowledge of how to use try/catch statements, more efficient ways to add data to arrays, and pseudocoding beforehand. Try/catch would help with checking that a said existing file *does* exist (now it does not check, and crashes if there is no such file), and maybe with checking to see whether an int is inputed where it's expected and not a String, etc. My translating things was a hassle and upsets me that it still can't open things, and pseudocoding might have saved me time in the end, though the planning would have taken a while.

I guess sometimes I can't really tell what's a draft from what's going to be part of my final product.

Overall, I'm happy with my add, edit, and delete actor methods (because they work and they're great in general). Out of everything, though, I am most proud of my search filters. I had some confusion early on with when to pass what, but I decided it would be a lot easier to use one ArrayList of String[]s because I could recycle the ArrayList for more searches! It was easy once I figured out how to search for one thing—copy-paste followed. So the cast list can be searched for any part of the name, grade, scene, etc., and the resulting search can be searched for any of those things in the same way. Yay! Surprisingly this wasn't the hardest part to code, but I think it was the coolest. Maybe I just planned it out the best.

Now if only I can get those arrays to take in the data correctly...