**Assignment 1 Report**

1. I estimate that it will take me until the 26th of this month to completely finish this assignment. I feel that this is a good project schedule to have because it forces a constraint on me to finish before the actual project due date, which may come in handy in case something comes up and my plans get pushed back. In that case I will still have a two day window to complete the assignment. Also, that gives me 12 days to work on the project and since I was born in December which is the twelfth month of the year, I figure that must be a good sign.

2. For this project I am going to assume that it will take me about 24 hours to complete every aspect of the project requirements and functionalities. There are several reasons for choosing this number. The first is that it gives me a lot of leeway. I am hoping to finish this project in less than half of this time, but if Murphy's Law has taught me anything, it's that if there is something that can go wrong, it will. There is no telling what kinds of errors I may incur when implementing the code portion, and what errors will incur when I attempt to fix those previous errors. And that's assuming that I didn't make any errors when I planned things out in the first place. Just from reading the description it seems easy enough, but I would still prefer to finish ahead of my self-imposed schedule rather than behind. Also, it gives me more time to spend planning and making a better design so that the coding is easier. Lastly, thought this was really my first thought in choosing this number, is that 24 hours reminds me of my ailing procrastination problem. I've always had a problem with procrastination going all the way back to my high school days, and I've lost count of how many projects and assignments I've put off until the last minute. All those all-nighters I endured. It may have worked then, but I've grown to realize the harmful effects that it had on me and the kind of quality of work that I was producing. Fortunately, it's gotten better and I do not plan to even consider the idea of doing that for any assignment this semester, or from here on out for that matter. Having a constraint of 24 hours simply reminds me of those simpler times and is my own personal humor.

3. I'm going to estimate that it should take about 400 lines of code to completely implement this assignment. There have been several programs I've written that were quite large, though a lot of that was due to not breaking down the problem enough, along with code reuse issues. With some proper implementation I should be able to optimize it to no more than 400 lines of code.

4. I was actually able to complete the assignment to the point where it worked, was correct, and performed all the necessary functions in 9 days instead of the 12 that I originally thought, however that did not include all of the comments and it did not mean that my program was as modularized as it could have been. Unfortunately, since I had finished all the actual requirements, I ended up procrastinating until the 27th on finishing up on all the remaining details.

The actual time-on-task to complete this assignment, was a total of around 11 hours and somewhere between 20 – 45 minutes. While I did manage to finish ahead of time, it took me a lot longer than I would have liked on fixing some errors that were more annoying than they should have been.

The total amount of lines of code that I wrote were: 573, not including those of TestOperations which was used to play around with JUnit a little. This meant I busted on my estimate by nearly 200 lines… well it could have been worse.

5. After reading through the description I immediately knew that I wanted to have a Patient class, that much was obvious. Not long after that I figured that I should also include a PatientGroup class. Those were the two main ones that struck me quickly. After that I couldn't decide if I wanted to have a FileIO sort of class to deal with file operations. However, there were a few leftover methods that I wanted to implement as well that didn't exactly fit with the other two classes necessarily, so I decided to lump them all together into an Operations class to deal with reading and writing files, creating Patients, and constructing the process for displaying the reports.

6. To make sure that my program met the requirements I first made sure that I knew what the problem was asking of me, what those requirements were. Also, I made sure to keep a good, basic design that addressed each issue. I also tried my best to ensure that my design was broken down in manageable pieces that I could use as methods. After the formation of classes and the implementation of the code I ran a few simple tests on some fundamental pieces, such as making sure that I could read in the file. After tweaking any bugs that came up I checked my final answer with the example given in the assignment to make sure they matched up. When that worked I played around with the input file data by adding patients and values and checked to see if it still worked. I did use JUnit on some cases, mainly just to learn about it and try to get into the habit of it. I also made use of certain conditions in certain methods, such as when I go to tokenize a line. The method will check to make sure that the number of fields is not greater than the constant variable MAX\_FIELDS, which has a value of 5 since there should only be 5 pieces of information for each patient. Any more or less causes a message to display to the console and the program will exit, forcing the user to start over.

I am quite confident in my program due to these instances. The tests I ran worked as expected and desired and while this does not mean that my program is necessarily correct or will be correct in all cases, the fact that I could read in a file and get the same results as the example given in the assignment based on the same input reassures me enough to have a relaxed confidence in it.

7. Looking back on this assignment, I am both content and dissatisfied. I am content with how it turned out and how, for the most part, I was able to meet my estimates with the exception of the lines of code written. However, while I did work on this project diligently for the first week or so, after reaching a certain point I got lazy and began to put it off. Fortunately I did not come across any major bugs that would have forced me to reconsider how I set things up. Even still, I procrastinated and so here I am, typing up this report at 10 pm, just under two hours before the assignment is due for submission. Even though my program has been done, I still procrastinate in regards to my report. Also, I found that I could not focus on the assignment for more than about 2 hours at a time, but on the occasions where I worked for a shorter time period, say about 25 minutes, my focus also decreased. It was also much harder to get back into my groove at times when I had quick, short, sessions. However, I did find that once I got into it time become less and less relevant. That said, a complete and detailed report of how I went about for this assignment is as follows:

The very first thing I did when I started this assignment was read the description. Then I went ahead and typed up a quick answer to the first three questions regarding my estimation for the duration and length of completing this assignment. Once finished I took the opportunity to read the description again, only this time I made a note on paper of the different requirements.

Now that I had a list of all the requirements that I needed to implement in order for the assignment to be correct, I started to look for requirements that could be grouped together, requirements that shared enough commonalities that they could be placed in the same class.

I initially decided upon three classes, the first of which was a Patient class, obvious enough since the assignment dealt with information about patients. By looking at the information for each patient in the file, as given by the example shown in the assignment, I deduced that I could use that information as the fields for the Patient class, these being the patient’s id, gender, total cholesterol level, HDL value, and the patient’s triglyceride level. These 5 pieces of information would be enough to instantiate a new Patient. For the LDL value and the patient’s cholesterol ratio value, I used two accessor methods which performed the computations based on the formulas given in the assignment and returned those values to the caller method. Likewise, I separated the classifications of all of the values and returned a string representation of that patient’s risk. One method I did add as an afterthought, was a method that returned the group that a particular patient belonged to, whether that be the control or the treatment, which was found by getting the char value of the last character of the patient’s id. I also included a method to return a string representation of the results of this patient. This would be the information that would be printed to the file as part of the patient’s report. The last thing I did for this class, oddly enough, was construct the constructor and ensure that certain preconditions were met, such as a patient’s gender being male (m/M) or female (f/F) and that the values initializing the fields were positive.

The next class on my mind dealt with the idea for a group of patients, PatientGroup. Like the Patient class, this class also had five fields. These fields would hold the sums of the values for the patient. I made the constructor first this time and initialized the values. I provided getter methods for each of the fields. Now I decided to add an additional field, an ArrayList, in order to add Patient objects of a group to a list. I implemented a method to make the sums of all five fields using the list of patients as references to get the values they held and increment them to the fields of sums.

Now I came to a dilemma. I knew that I needed to retrieve the averages of the values for the patients in the control group and the treatment group. However, looking at the values that I should get based upon the example given, I realized that while the first four columns containing numerical data, namely all those except for the ratio, were integer approximations. The ratios were doubles. Because of this I opted to go ahead and create two separate getAverage() methods and two separate getDifference() methods, one of which would return an integer representation and the other a double for both sets. I then created a method to return an ArrayList containing the averages of all of the sums of the group. The last method was a display method to return a string representation of the values in the list of averages.

The third class I implemented was simply an application class which prompted the user for a filename and sent that filename, along with the output filename, specified as report.out, to a method I denoted as createProfiles which was a part of my fourth, and final class for this assignment.

My last class, which I named Operations (since it dealt with various operations of file reading and writing, creating patients, and creating the profile reports. The constructor initialized two ArrayLists, one to hold the list of patients and the other to hold the list of lines from the file. Getter methods were provided for both, along with a method to add patients to the list of patients. I then created methods to read in a file and add its lines to a list of lines, and a method to write out the personal report for the patient. I created another method to make the report that would be displayed to the console.

Now I implemented a method for creating patients. It was one of my more longer methods at around 30 some lines, but it was simply intended to take the list of lines, tokenize those lines (ensuring that the maximum tokens per line was exactly 5 for the 5 fields in the Patient class), create a new instance of a patient and finally add that patient to the list of patients.

With all of this done I wrapped it up by creating a method that would call the methods to read from the file, create the patients, generate the reports, and write the personal report to the output file. The only thing left to do was document, since I always tend to leave commenting as the very last thing to do. Most of the time, anyway. Once finished, I typed up my response to the final few questions for this report and, well, here we are.