**Task 5 – Program Modification / Reengineering.**

First I traversed the code listing in the article and wrote pseudo code. I then attempted to traverse the pseudo code to verify that it matched the original code. I copied the pseudo code into the java class I was developing in the Eclipse IDE and started to replace the pseudo code with actual code. I was not confident that I captured the behavior of the code, so I thought that using a flow diagram would be more reliable. I then prepared a flow diagram in MS Visio based on the diagram listed in the article. I used the flow diagram in combination with the original code to ensure my pseudo code represented the behavior of the original program. I then began to edit the java code. My strategy was to identify specific sections of code to work on, get a section to run, submit a commit for the change, and move on to the next section. I also was not concerned with the beauty of the code at this point and planned to go back and refactor after this initial activity. I was able to improve on some of the logic to make the code easier to read.

After getting the initial code running, I went back and made changes according to the Java Programming Style Guidelines. While doing so, I realized that this program should be refactored with inputs and outputs from the original subroutine as public class variables. I used some of the automated tools provided in the Eclipse IDE to clean up the code.

I made sure to commit changes for each significant change. I sometimes had to bundle multiple significant changes in one commit, because I would forget to commit a change. I also reworded some of the previously sent commits, because I felt the comments were incorrect or insufficient. I rebased the repository by mistake and notice that the commits were being pushed to the local repository but not to the remote repository. I then aborted the rebase to try and resolve the issue and it caused about four (4) hours of commits to be lost. I was able to recover the latest version of the java source code, so I have a four hour gap of commits in my repository.

When I was regularly doing this type of work some time ago, I remember that changes were controlled by using a development system. When a change was needed, it was incorporated into the development system and then moved to the production system. Changes weren’t authorized to be moved to the production system without adequate verification and review. This took time and was cumbersome. GitHub makes this process much easier.

The documentation in the article, the flow diagram, and the comments in the original code were helpful in understanding the code behavior and for verifying the converted code. It would have taken much longer to complete the task if given only the code.