Criterion C – Development Analysis

Primary Issues/Comments

1. GUI Implementation
2. Filtering User Input & Exceptions
3. Usage of CSV for Data Storage
4. Data Persistence & Exporting Entries to CSV
5. Searching CSV for Previous Entries

1. In order to display a graphical user interface, this program utilized Java's Swing API. One of the primary concerns of the program was to have it look clean and presentable, and make an efficient use of space. Initially, I planned to display JButtons with ActionListeners across the top of the screen as a "navigation bar", to have the user navigate between the different GUI pages of the program. However, this looked clunky and made inefficient use of both GUI and system resources. During my research, I found TabbedPanes, which enabled me to add a clean and easily organized navigation design to the program.
2. This program relies heavily on user input to generate, store, and recall data from an external file. However, with user input, there comes the danger of invalid or unrealistic entries that may be entered into the program. For example, a user may choose to enter a value far greater than 12 for a field that asks for month data. In this case, I have included code that restrict user input to specific ranges. The program will reject most user attempts to enter numerical values that do not represent a realistic value.
3. Initially, I planned to use a plaintext (txt) document to store values and settings for the program. However, research yielded the Comma Separated Value format, which is praised for its universal nature and easy accessibility compatibility by multiple other programs. The format is simple to generate and work with, as text and other entries can be converted to strings and delimited by commas and new lines. I decided to use this format due to its ubiquity and easily searchable nature.
4. Exporting entries to CSV proved to be challenging, in part because it required the use of the external FileWriter class. Delimiting entries also posed a unique challenge, since the data was not formatted into an array or easily appended string before storage. As such, I did research and found sample code by ashraf on Java Code Geeks, which I used as a template and adapted for additional functionality.
5. Searching the CSV files proved to be challenging due to the similarities of the dates. Different entries in the CSV may all begin with the year 2017, for instance. I therefore created a "date code" for each entry, by concatenating the year, month, and day strings together. This created an easily searchable entry ID.