Cory Hershman

ThreadedSort Post-Mortem

This post-mortem will discuss the steps that were taken to complete the ThreadedSort program. First, the requirements were analyzed and listed. Then, time was spent learning how to use multithreading in java. This encompassed about 60% of the total time spent on the project. The most useful resource was Jacob Jenkov's tutorials on concurrency/multi-threading. URL for his tutorials is listed at the end of this document. Learning multi-threading for java was extremely important for this project and will be useful afterwards. Afterwards, time was spent analyzing the problem and designing a solution using whiteboard and paper. This encompassed about 30% of the total time spent on the project. Finally, the classes for the program were created and tested. This encompassed about 10% of the total time spent on the project. After spending plenty of time designing, the coding went very smoothly. Very few problems/revisions encountered converting from the pseudo-coded outline of the program to java.

The biggest revision was removing an unnecessary merge method in the "ThreadedSort.java" class. This method originally was meant to merge the two final halves of the list together after the MergeThreads did their job. However, this was deemed unnecessary as a change to the implementation of the MergeThreads allowed the MergeThreads to finish the last step of the merge.

A concern going into the project was figuring out where Thread best-practicing needed to be implemented, such as synchronization blocks, and volatile data types. However, upon closer inspection, the program on a whole is Thread safe. The only concern was that the writes to the list might have been kept in the CPU's cache while processing the rest of the changes, so a volatile variable was added to flush the CPU's cache to main memory. Future improvements would include a fail-safe in "MergeThread.java" to protect the MergeThreads from each other if used in a way not originally intended, such as having MergeThreads overlap each other in the array, to abide OOP best-practices and allow "MergeThread.java" to be more easily reusable without "ThreadedSort.java"

Learning how to implement multi-threading in java is the most useful information learned from this project going forward.

Jacob Jenkov's tutorials: http://tutorials.jenkov.com/java-concurrency/index.html