Michael Weaver, Caleb Miller, Mathew Paschall

CST – 235

Brandon Bass

12/17/17

**CLC 6**

GitHub Link: <https://github.com/battousairurik/CST-235>

**Assignment Approach**

1. For this assignment I followed an example that I found online. I use the concurrent Executor Service to utilize several threads to solve the problem of finding the sum of the work. Each recursive call breaks the work down into smaller sections until a work minimum is met and then the sum generated. For ease, the steps are written to the console.

**Questions**

1. The fork/join framework is an implementation of the Executor Service interface which aims to make use of multiple processors. It breaks down work into smaller portions recursively and assigns each smaller portion to a different processor to enhance processing power and performance. The service distributes work between a pool of threads, and unique to the fork/join framework is the work stealing algorithm which allows portions of work to be taken by inactive threads from other threads that are still active (Oracle, 2017). Specifically, Fork is when the work is broken down and passed into different processors. Join is when the work is merged and passed back up the chain to be processed for some final solution.
2. Answer

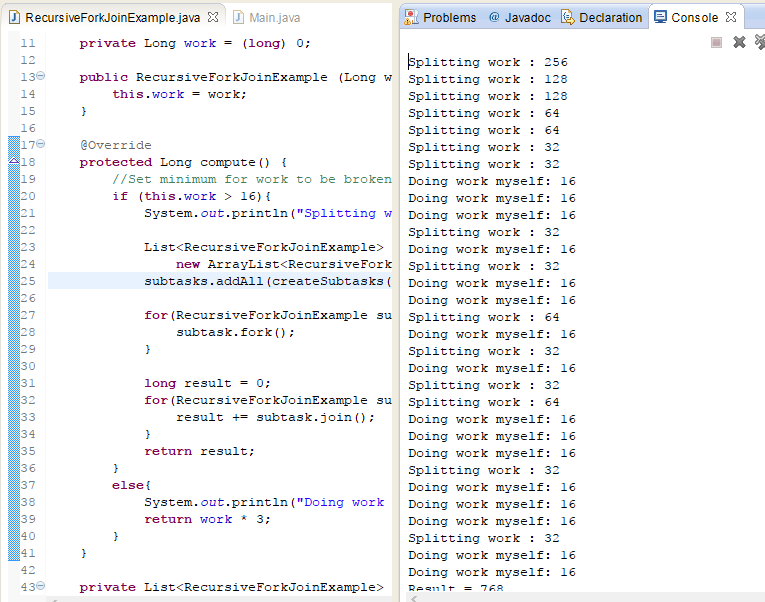
**List of Classes, Methods, Variables, etc.**

*Fork/Join Example*

RecursiveForkJoinExample.java (containing work field, constructor, compute, and create subtasks methods) and Main.java (containing only the main method to run an example).

**Screenshots**

Fork/Join Example

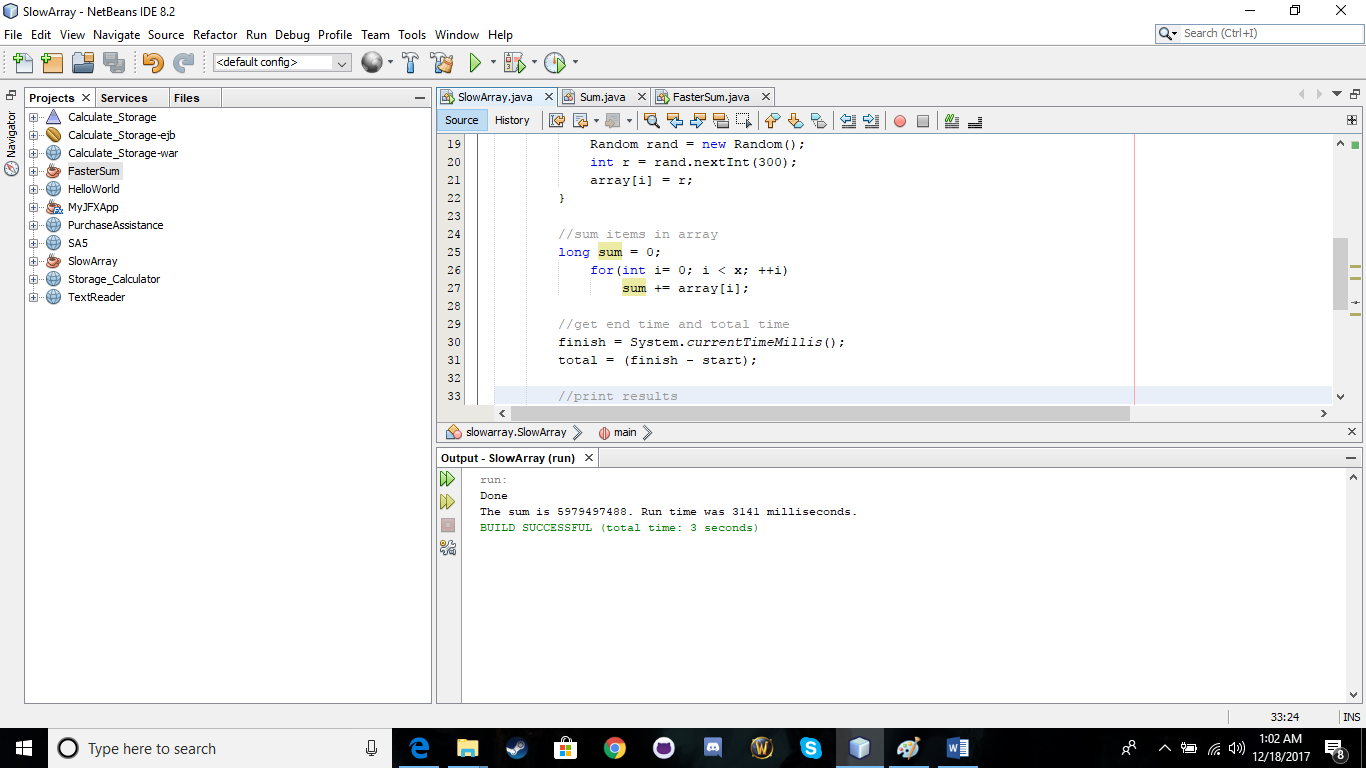


**Mathew**

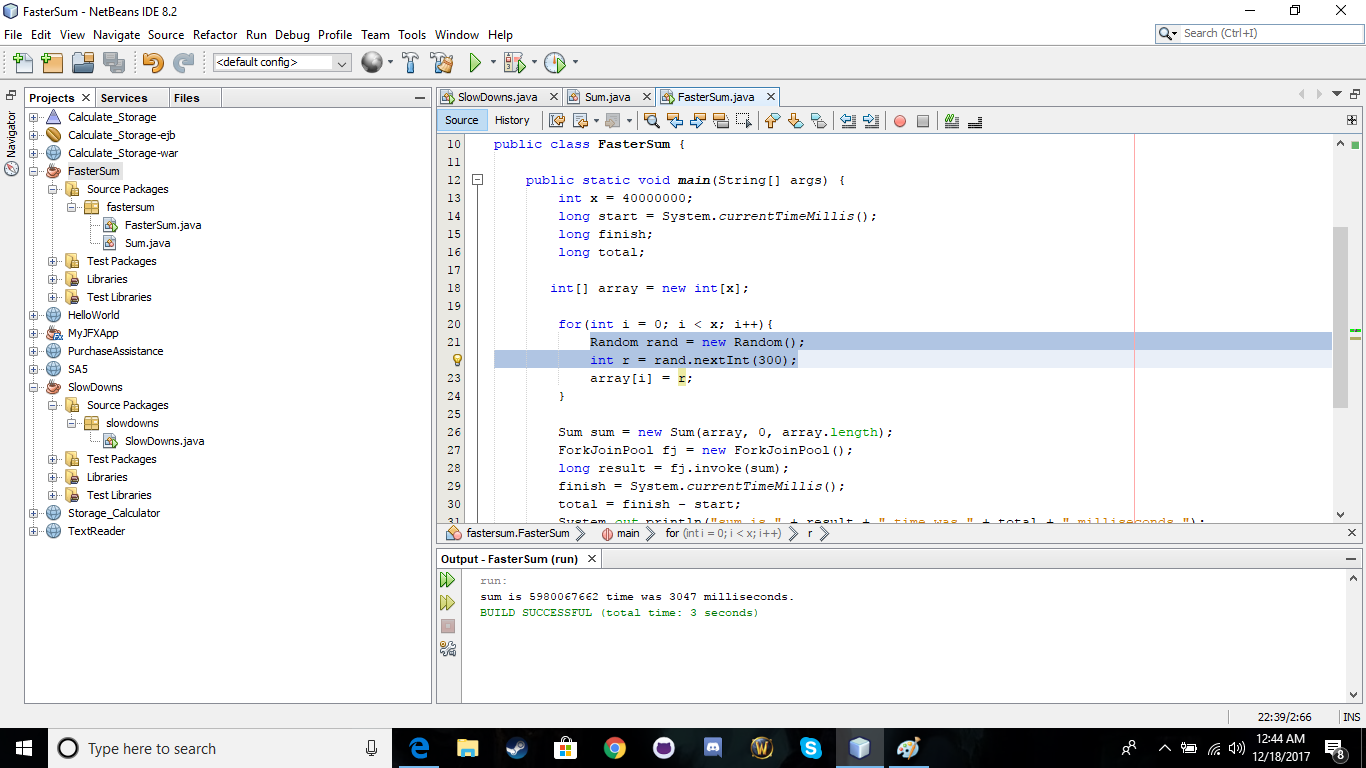
Approach:

I created a very large array of random integers and had the program add them all together. I had some trouble using the fork method to make this work properly, but after a lot of research I got it to work properly. I think with more computations the fork method would speed up processes with arrays, especially sorting. I also think that loading images would be a good thing to use parallel programming to achieve.

Array Sum Slowdown



Faster Array Sum



**References**

Oracle. (2017). *Fork/Join*. Retrieved from <https://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html>

Jenkov, Jacob. (2015). *Java Fork and Join using ForkJoinPool.* Retrieved from <http://tutorials.jenkov.com/java-util-concurrent/java-fork-and-join-forkjoinpool.html>