Evan Wilson / Nicholas Thomas

CST-235

September 2nd, 2018

Instructor: Dr. Toure

CLC Mini Project 6

Green Group

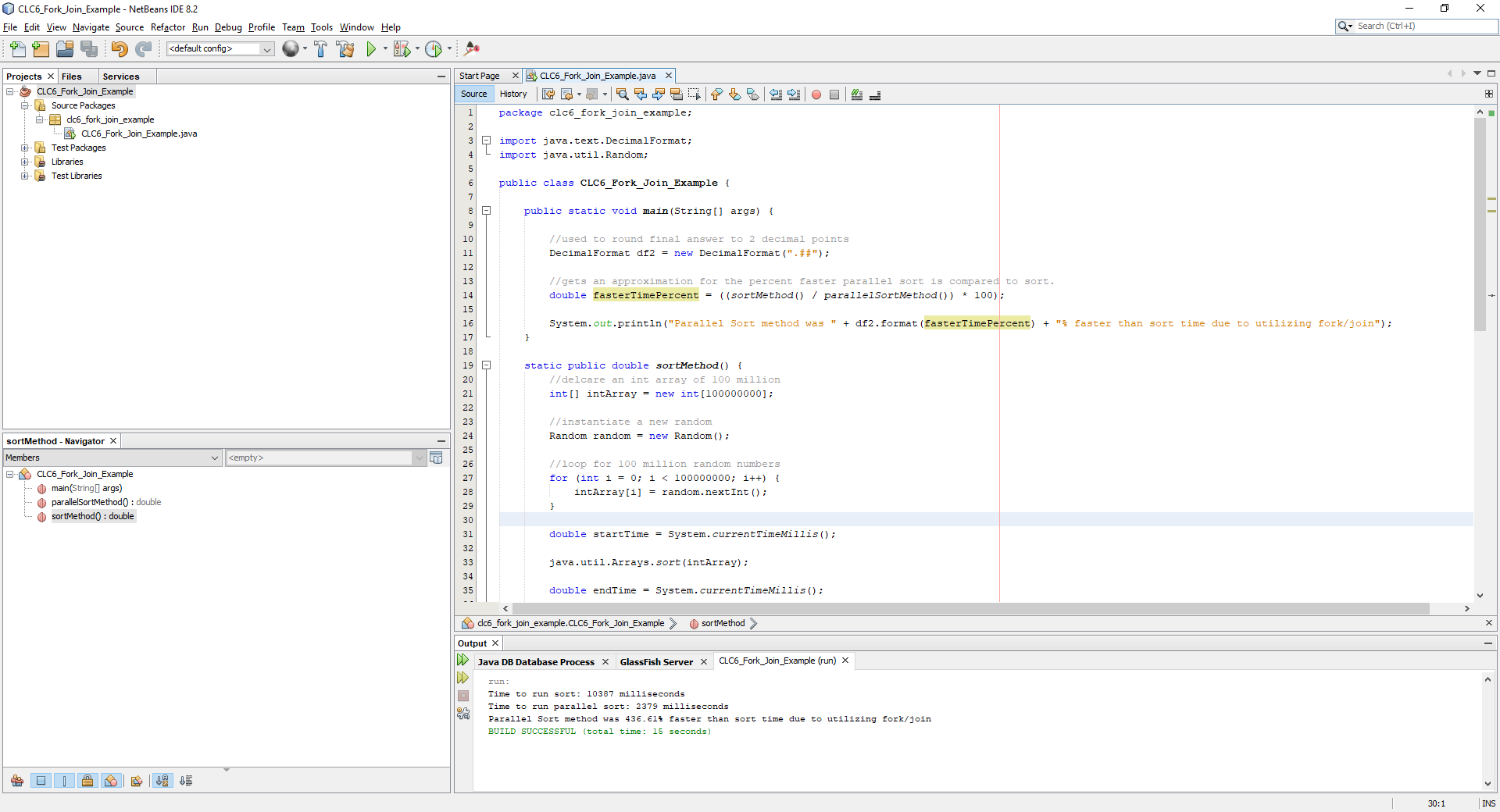
Github Link: <https://github.com/ewilson87/CST235_ewilson>

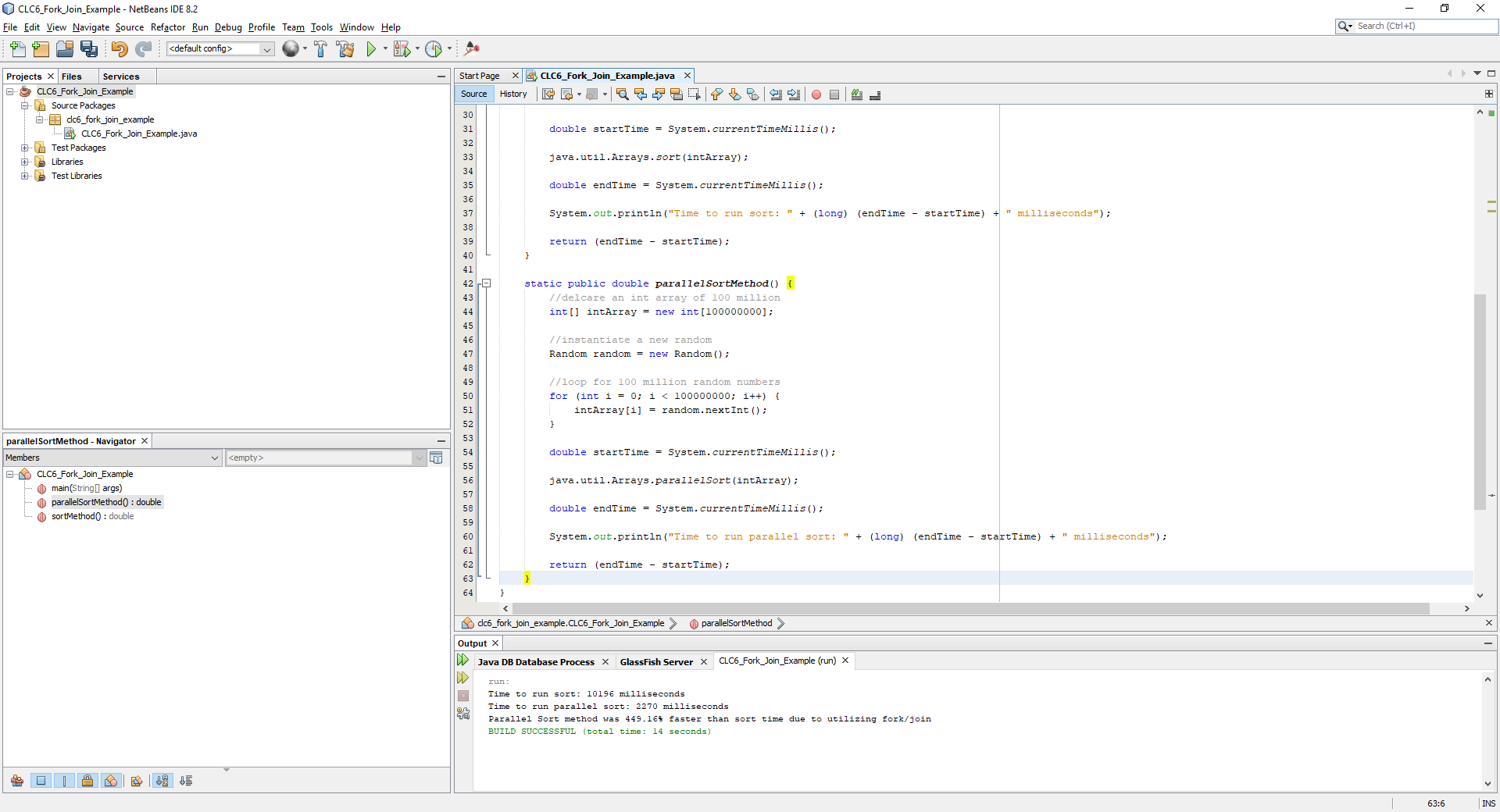
Folder: CLC6\_Fork\_Join\_Example

Question 1. Explain Fork and Join in Java

Fork/Join is a Java framework that got introduced in Java 7. It allows applications to run with more efficiency using parallel processing through the use of all available processing cores on through a recursive divide and conquer approach (Baeldung, 2018). It starts out by recursively forking the task into as many sub-tasks as it can to process concurrently amongst the available processors. Part of this process involves the task to first determine if it’s large enough to even make worthwhile use of multiple processors because there is an increase in processes to split up a task, so if the task is too small it will decrease performance (Oracle, 2017). After all sub-tasks are completed, the join process starts, which is essentially the same thing in reverse until the task is complete. All of this is centered around the ForkJoinPool class which implements the algorithm that detects if there are available processing power not being used, and to “steal” that work and use it amongst the free processors (Oracle, 2017).

An example of this is shown below in a simple Java application I created that creates two separate integer arrays, filled with 100,000,000 random integers. It then sorts one array using the old sort() method, and the other using the parallelSort() method introduced in Java 7 to utilize the function of the fork/join framework. It then lets the user know how much faster the parallelSort() method was.





# References

Baeldung. (2018, June 4). *Guide to the Fork/Join Framework in Java*. Retrieved from Baeldung: https://www.baeldung.com/java-fork-join

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