HOMEWORK WRITE UP

**Additions to makeData.java and dp.java**

The programs makeData.java and dp.java still run as described below. However, I have added a couple of things. MakeData.java now has timing where it times the time it takes for the entire function to run and the time it takes to create the file separately. Dp.java has switched T\_W and T\_O because they were messed up on the instructions. Dp.java also does not include headers on every print statement so that the file can be generated via python. (The makeData.java and dp.java have been carried from last assignment the descriptions are the exact same. However, all the screen shots are completely new)

**Gather\_data.py**

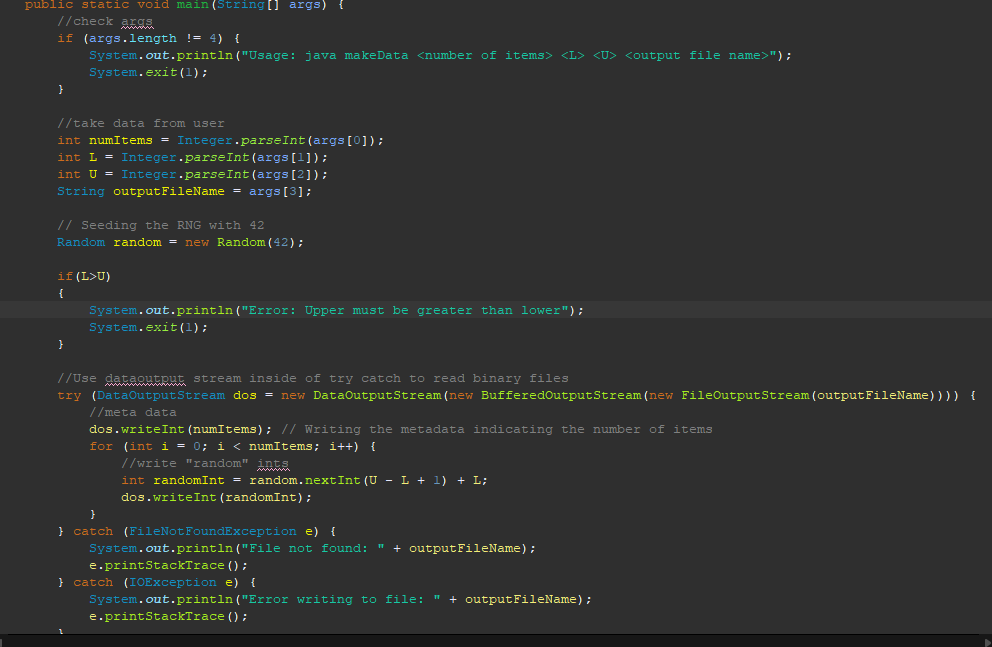
This function is responsible for automating the process of collecting data for multiple problem sizes by running the makeData and dp Java programs. It takes parameters specifying the range of problem sizes (N to M with a step size), the lower and upper bounds for generating random data, and the output file where the results will be saved. For each problem size in the specified range, it executes makeData to generate input data files and then runs dp to calculate the dot product using these files. The output of dp is written to the results file, allowing for easy analysis of the performance of the algorithm for different problem sizes.

**Plot\_data.py**

This function is responsible for visualizing the collected data by creating a plot of the total time (T\_O) and loop time (T\_W) versus the problem size (N). It reads the data from a results file and plots it using Matplotlib, with appropriate labels and formatting for clarity. The resulting plot provides a clear visualization of how the total time and loop time of the algorithm vary with the size of the input data. This helps in identifying any trends or patterns in the algorithm's performance and can aid in making decisions for optimization or further analysis.

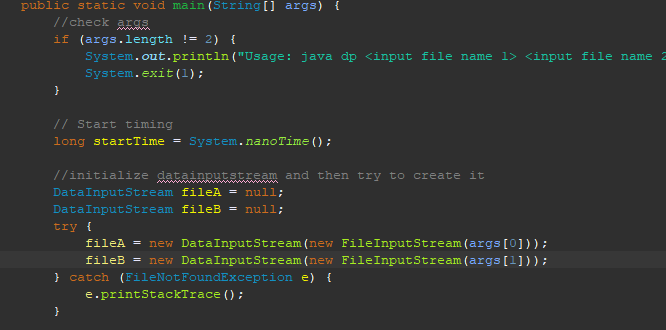
**makeData.java**

Make data starts off with the usual for these homework's. Checking for the argument size and getting the user data. Then we seed our random number object with 42 as specified. Next, we check if lower is greater than upper. If it is, this will cause an error. The next thing we do is use a try catch block and initialize the binary output stream. Inside of the try block we write our meta data to the file, that being the number of items to print. Next, we loop through the amount of number of items and create our random numbers and write that to the file. I did not include timing in this program as it was not specified.

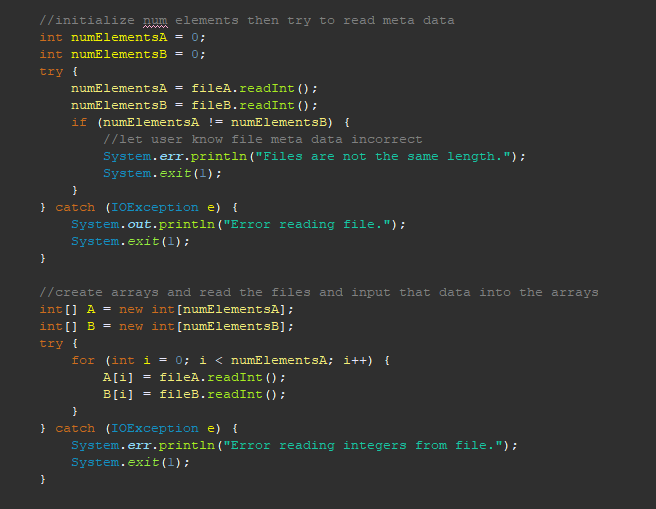


**DP.java**

Checks args for elements and then starts the timer. Program attempts to initialize datainput stream to read binary data inside of a try block.



Next thing is to try and read the file for the number of elements. This is done inside of a try catch. If there is no error the program, then creates the arrays initialized with the number of elements and then reads the rest of the data out of the file inside of a try catch and for loop.



The next thing the program does is time the calc\_dp function and execute that function. The calc\_dp function sums up the multiples of each element in the array. Next, the program finishes its total time and prints specified data to the screen.



HOMEWORK SCREENSHOTS

