Hayes Crowley

CSC\_204\_LAB\_12

Prof. Papademas

**OUTPUT OF MODIFIED PROGRAM:   
n columns and rows = 2, total elapsed time = 3ms**

**Text

Description automatically generated**

**n columns and rows = 3, total elapsed time = 4ms**

**Text

Description automatically generated**

**LAB QUESTIONS**

1. Performance should not differ between different Java *development* environments (e.g., Eclipse vs. JetBrains vs. VSCode), however it would differ between different versions of Java and its JVM, as this is where the code is compiled and run, and of course the performance of Java programs would also be dependent on the hardware it is run on, since the JVM is itself a program and is not the real machine.
2. Some factors that determine the performance of matrix addition in the Lab program are number of rows and columns, as well as the data type of the numbers being used (e.g., floats and decimals would take longer than integers). Other factors include the Java version and hardware it runs on, as discussed above.
3. The “pre-time” and “post-time” long-type variables were used to benchmark our program’s performance in milliseconds, using a standard lib function that returns the system’s current clock time in milliseconds
4. In the program, a nested loop (a loop over the row and then the columns) was used to build each matrix, display each matrix, and then sum and display the result. I count 6 nested loop structures in all, each with O(row\*column) complexity (i.e., it would take 2 \* 2 iterations to display a 2X2 matrix), giving this program a total complexity of O(6 \* (row \* column))
5. Yes, as this program only prompts for entering dimensions once, and does not have any logic for handling addition between matrices of differing dimensions.