**Module 5 Option 1 Program Analysis**

James Strauser

Colorado State University Global

CSC 450

Jul 2, 2023

For this program analysis I am supposed to discuss performance issues with concurrency, vulnerabilities exhibited with use of strings, and security of the data types exhibited. I will be doing these in no particular order, starting with the discussion of strings… vulnerabilities with strings really only exist when the user has access to the system but that is not part of the program so strings are very secure for this program, and in Java they are a lot more secure than C++. Next is the security of the data types exhibited, there are two. First is the int used as a counter. It is a global variable so it is not very secure but it would be difficult to make it exhibit undefined behavior as the values of 20 and 0 are very far away from overflow values. Next is the thread data type, lots of difficulty there in terms of making it work properly but nothing inherently bad with security. For Java threads behave differently than those in C++, first off is the lack of locks, or that is at least what I thought, in reality there are a lot more options for locks in Java and it is up to the programmer which to use as they all have pros and cons. You may notice I did not use any locks, that is because I misread something when making the program and thought it said I do not need locks and that Java handles it automatically, this is false. In fact there is a strong chance the bug in my countdown thread where I required a sleep in the first loop that checks for the counter to be high enough was caused by it not having a lock on the variable, but I do not know for sure all I do know is adding a sleep condition made the program work every time I ran it. Speaking of the sleep commands, in Java it requires a whole try catch block because threads can be interrupted and it needs to know what to do then, this is actually pretty good in terms of security because it lets you unlock variables or lock them if the thread gets interrupted at a bad time. For this instance though, that does not happen. Aside from those and bulkier code, Java and C++ threads are relatively similar. Purely in terms of my implementations however, the C++ is leagues ahead, this Java program is a very poor job at threads compared to C++ where I understand the implementations a lot better. However when we go towards best case scenarios, I think Java has more secure multithreading, I am certain both languages offer all features in some form but Java makes error handling a requirement for compiling and has what seem to be simpler error handling.