Vincent Wong
VYW180000
CS4348.001

Summary

This project was about using semaphores to simulate an elevator with 49 passengers. The maximum capacity of the elevator was 7, so there would need to be 7 trips to ferry all the passengers to their destination, a floor from 2 to 10. The threads representing the passengers and the thread representing the elevator needed to work in sync with each other, so that passengers do not attempt to enter or exit the elevator until the elevator door is open and they are on the correct floor.

This meant utilizing semaphores to ensure each thread waits its turn before completing its given task. A queue was utilized to track what floors the passengers wanted to access. The elevator then checked the queue using a nested for loop to check if a passenger wanted to exit the elevator from a certain floor. When I was writing this nested for loop, I was thinking about how the time complexity of this loop would be O(m*n) but upon further thought I realized that O(7*10) really isn't that large, and time complexity is only relevant when there is a very large input. I used the nested for loop to increment a count of passengers wishing to exit on a certain floor, resetting after each floor. When this count was greater than zero, the elevator would stop, open doors, signal the passengers, then close door and continue moving.

Something I took a bit of willful creativity with was the naming of the passenger class. Funny enough, I casually asked what I should name the class to a friend studying with me off to the side, and he told me to name the class "StairHaters". Feeling a bit of playfulness, I kept the name because it made perfect sense. Even if I wanted to go to the top floor of the building, I would blanch at the idea of having to wait for six full elevators to make their journey before I would get my turn. This is even more incomprehensible for passengers who have a very close destination, such as floors two, three, and four. Are they seriously that lazy, or are they simply allergic to the stairs?

I had many bugs in my code that I kept track of, most of which were simple to solve and easy to understand. Mostly due to misplaced semaphores, threads releasing their permits when they weren't supposed to, or sometimes a missing semaphore entirely. These bugs were easy to find and easy to fix, such as passengers exiting before the elevator opened, elevator opening on the same floor several times instead of just once, passengers entering the elevator before the elevator returns to the first floor, and elevator skipping a requested floor. Something slightly more difficult was how I resorted to using Thread.sleep() instead of a semaphore during testing phases, since I wasn't confident with semaphores until working with them for a while. This purposeful design fault was left to the end to fix, but once I got around to fixing it, it turned out to be quite simple.

Another bug was caused by passengers releasing their exit permit when they weren't supposed to, causing later batches of passengers to willfully exit the elevator when they were supposed to be able to. I had difficulty understanding the cause of the bug initially, so I simply used drainPermits() at the end of each elevator cycle as a workaround. I found the bug and removed permit draining once I had the program fully working and discovered the issue while commenting the purpose of every semaphore. The most difficult bug I had to solve was CurrentModificationException()'s thrown by the elevator thread. At first it had to do with the syntax I used to reset the queue after each journey, but even after that I kept getting the problem. For some reason, the error was occurring on the line of my enhanced for loop to check through the queue. I could not discover the cause of the bug since it did not appear to match the cause documented online, but it resolved itself afterward once I stopped using Thread.sleep() in favor of semaphores. I'm still not sure why it was happening.

Perhaps something else I found interesting was that I preferred to write out the code slowly in java and write the pseudocode afterwards once I fully understood how semaphores worked through the extensive usage of trial and error. Was this efficient, I'm not sure. But I certainly felt less lost once I was able to write the program and had visible errors to teach me where I was wrong.

As of this moment, the middle of Sunday 10/24 I am having difficulties accessing the linux server. I am going to bring a USB drive down to the ECS labs and try and access the server there. If I am unsuccessful, I will bring up this issue during the next class during the time allocated for project related problems. I kind of feel like I am writing a journal at this moment, so wish me luck!

UPDATE: As of about thirty minutes later, I have managed to make the program work as long as the instructions in readme.txt as followed. Annoyingly enough, I cannot sign in to the cs1 server from my laptop in the UV apartments or ECSW, but it does work while inside the ECS labs. How strange, but at least it works.