Sulayman Hafizullah

4/20/2022

CS 4348.002

Project 2 Summary:

In this project, I simulated a movie theater using threads for customers, box office agents, ticket takers, and concession stand workers. The general flow of the simulation was very easy to follow. The customer enters a line and sees a box office worker. The customer then selects a movie, and the box office agent gives the customer a ticket for that movie unless it is sold out. The customer then proceeds to the ticket line. Once the customer’s ticket is torn, they randomly choose whether to stop at the concession stand or not. That is the final step before entering the theater. Once the customers entered the theater to watch their movie, the customer threads are joined. The employee threads would not terminate as they are set to run forever. I remedied this issue by using a system exit call after all the customer threads were joined so the program would not continuously run.

Overall, I did not have any difficulties in writing the code to simulate the customer passing through and communicating with these other threads. My difficulties began when trying to get each thread to remember the customer and choices accurately using semaphores. When I first began having the customer communicate with a box office agent, everything went well as long as only one customer was allowed in at a time. I also was able to successfully run the program with only once box office agent. The presence of two box office agents presented the largest problem for me in this project. The method I ended up discovering and implementing was to create two lines that the customers waited in. Each line directed the customer to a certain box office agent. This would also prevent all the customers from going to only one box office agent. If one line was not free, the customer would check the next line.

The next issue I faced when writing the code for this simulation was accurately storing the customer’s data. I found that when using a single variable to store data, the variable would just get overwritten by other customer threads that were created. The method I used to prevent this was to create an array that used the customer ids as indexes. This allowed all threads to easily communicate data without having it be overwritten by the next customer that comes along. I used these arrays to store the movie choices and concession choices.

Semaphores were the main focus with this project and were necessary to accurately simulate the theater. I used semaphores to make sure only two customers at a time could interact with the box office agents because there were only two box office agents. I also used semaphores so only one customer could interact with the ticket tearer, and concession stand worker. Without the use of semaphores, endless amounts of customer threads would try to communicate with the employee threads at the same time and each customers data would not be accurately stored.

Throughout this project, I learned how to implement a few new topics. These topics are threads and semaphores. We had already covered both of these topics during the first half of the semester, but I was not sure how to implement them into code until I worked on this project. The examples that were given on elearning really helped me understand exactly how to create multiple threads by implementing the Runnable interface. There were also examples that showed the syntax and how to use semaphores in Java and C. Just those simple examples allowed me to easily understand the concepts that were necessary for this project.