Ishva Patel
Prof. Salazar
CS 4348.502
11/1/2022

Project 2 Write up

Approach to the project:
My approach to the project was to first understand the interaction between the teller threads and the customer threads before building up the entire simulation. To this, I had to break down what information is being exchanged between the teller threads and the customer threads. For the customer threads, I identified that the customer is giving their id and the type of transaction they want to be done. For the teller threads, I identified that the teller is exchanging their id with the customer. After that, I created objects for both the teller and the customer. This was so that they can store the information all the information that is being exchanged in a single object. Then I worked on creating the threads and making them interact with each other. While also identifying the shared resources, such as the door and the manager. I created semaphores for all of the shared resources while also creating semaphores I think were needed for synchronization. Then I worked on increasing the number of tellers and customers and made adjustments to the interactions until all 50 customers entered, did a transaction, and left the bank. Then I worked on closing the threads to end the program.

Organization of the project:
The project has two nested classes, in addition to a couple of functions for entering the bank and doing the transactions. The two classes are for the teller and customer so that they can be objects and store information that is needed for the transaction. For the customer class, it assigns the values to the id and the transaction type. There is also a function to convert it to a string. Also, there is a function that gets the transaction type, which is randomly selected. In the teller, the class assigns the Id, as well as contains an str function that converts it to a string. The main program contains two primary functions. One function is used to get the customers into the bank and into the line which is a queue. The second function handles the overall transaction of the teller and the customer. In that function, there is a while loop that keeps going until the line is empty (queue) is empty. In the while loop, the teller indicates to the customer that is open, and the customer goes to the teller and gives the transaction type. Then there is an if-elif, that is used to differentiate between the withdrawal transactions and the deposit transactions. After that, it goes to the safe and finishes the transaction. To create the threads, there are two for loops. One for loop creates the customer threads. The target of this thread is the enter bank function which allows the customer to enter the bank and adds them to the queue. The second for loop is for creating the teller threads. The target for these threads is the transactions function which handles the interaction between the customer and the teller. Finally, two for loops are used to close the threads.

Problems Encountered:
Some problems that I encountered were how to get the threads to interact with each other. Another problem that I encountered was figuring out a way to organize the customers in a way

that allows them to be ordered. In addition to this, I had trouble figuring out a way to hold the information that was being exchanged by the threads.

Problems Solved:
I was able to solve the first problem, by slowly building the project up. I initially had started trying to fix the interaction for the 50 customers and the 3 tellers, but I soon realized it was better to figure out the interaction between 1 teller and 1 customer and then slowly increase the number of tellers and customers. I was able to solve the organization of the customers. Initially, I thought I would need something that can store multiple values for a specific customer. In the beginning, I was leaning towards placing the customers in a dictionary type. Then I realized it might be easier to organize the customer with an array using a FIFO type of system, which pushed me in the direction of a queue filled with customer objects. I was able to solve my last problem using the idea from OOP, of creating objects for each customer and tellers threads. Using those objects to hold the information that is being exchanged. This was to keep all the information together.

What I learned from the project:
I learned how to use threads to interact with each other to run a bank simulation. I learned how to create threads and semaphores using the threading submodule in python. I learned how to use nested classes in python and how to ensure they interact with the main part of the program. This is important because it can help with creating more complex python programs that rely on multiple different objects. In addition to that, I learned how to utilize semaphores for the synchronization of threads and how to maintain the constraints on shared resources.