Participants: Group 7 – Luca Zen, Jamie Troup, Duke Woy

Word Count:

Introduction

This assignment was about creating and solving a data race problem by simulating a bank computer and ATMs.

Approach

In our first meeting we decided on how we would collaborate and who would roughly work on what section.   
We originally decided to use GitHub as our main form of collaborating, however Discord become our main source of collaborating over time – sharing the code in the chat. We decided that each of us would build upon each other’s code one at a time. Whenever problems arose, we would pool together to offer solutions.  
The group identified some key topics to investigate and designated who should work on what: the design of the ATM was made by Luca, Jamie started implementing the base program using C# and Windows forms and Duke investigated handling multithreading.

Later, the group banded together to further transform the code given by the lecturer into a suitable Windows forms application. Luca spearheaded the discussion while, Jamie and Duke helped whenever problems arose.

Then the group spilt up again, this time with Luca working on using buttons in place of while loops, Jamie further refining the design of the central bank computer while also solving the data race by using semaphores and Duke implementing a log feature to make the data race problem more apparent.

Problems and Solutions

One of the most important problems was the while loop in the ATM class’s constructor method. This method in the original program was where the user input took place however within the Windows form, the group couldn’t figure out and didn’t know if there was a way to wait for user input to be entered which meant that the program would be stuck in an infinite loop – and worse, the ATM Windows form wouldn’t even show up.

Another problem that came up was handling the use of semaphores and their functions, at first it seemed like the semaphores didn’t work and would render the form unresponsive. This was seemingly solved by implementing the semaphore and its function within a try block and releasing it within the final section of the block.

Another problem was the text log, which is a child class of the text box, means that it’s methods could not be accessed in threads other than the thread it was initialised in because it’s unsafe. To fix this, we needed to introduce a delegate into the ATM class – doing so, along with a method which makes use of Invoke which gives the program a ‘safe’ way to access the original text log.

Future Ideas

We have identified several ways we could improve our program in the future.

One way is by spending more time using images and textures to make an appealing graphical interface, with more colours that would be representative of a real ATM.

The program could potentially try and branch into using other in-built classes that could also potentially deal with the data race problem, i.e., mutexes.