Introduction

The problem which was required to be solved was to design and create an ATM simulator in C#, making use of threads to run separate ATMs at the same time. We decided to use GitHub so we could both work on the ATM concurrently.

Approach to Creating ATM Simulator

To understand the logic and steps required to be carried out by an ATM, we used the system diagram in the assignment brief.

We then divided the functions of the ATM between us and set deadlines for each of the relevant tasks to be completed. This was effective as we completed the desired functionality of the ATM before the agreed deadline.

Once the basic ATM was completed, we both worked on completing the data race and the fix simulations. We researched into threads and semaphores to implement the different simulations.

After the simulations were running correctly, the ATM GUI was updated to be more realistic and aesthetically pleasing, making the actions required by the user to be as simple as possible.

Difficulties Encountered

Throughout the creation of the ATM Simulator, we came across several difficulties. One problem was allocating different functions to the different buttons on the keypad. To do this, the cancel and enter buttons have different event handlers. This allows the two buttons to carry out separate tasks. Another solution could be to store the words ‘cancel’ and ‘enter’ in the two buttons and use an if statement to differentiate between the number buttons and the two buttons in the same event handler. We did not feel as though this was the best solution as it was defeating the ‘one purpose per method’ idea of effective programming.

Another problem we encountered was understanding how to implement semaphores. To understand how they operated we made use of the lecture PowerPoints and conducted further research on the semaphore class. This allowed us to both understand their functionality, allowing us to make use of the methods within the semaphore class to allow the Data Race Fix option to operate successfully.

Furthermore, another issue that occurred was generating a log from the bank computer. We attempted to create another form and update it through this but we felt it was an unnecessary, complicated approach. To overcome this, we researched into printing the log to a file and displaying it through the console. We decided that the best approach would be displaying it through the console allowing the user to see the transactions in real time. We had to change the settings of the application so output would go to the console, allowing the console to be displayed with the other screens.

Features of ATM

One feature in the ATM is the generation of a receipt when money is withdrawn from an account. This adds authenticity to the ATM by adding visuals which appear on real ATM receipts.

Also, another feature of the ATM is the ability to see the bank log whilst the ATMs are in use. The bank log can show what the ATMs are currently doing and recording the time that the process has been carried out at. This allows the user to clearly understand the processes currently being executed.

Furthermore, a useful safety feature of the ATM is that if the user makes three incorrect attempts at entering their pin, the user will be blocked from using that account on the ATMs. This was added to recreate a similar process when using a real ATM.

Instructions

To begin the Data Race Simulator, click on the Data Race button on the main menu.
To begin the Data Race Fix Simulator, click on the Data Race Fix button on the main menu.

Word count: 598