AC22005 - Computer Systems 2B

Coursework 3 – ATM Simulator using Threads in C#

# Introduction

The brief was to develop an ATM simulator, using a bank system, with more than one ATM whilst also having the features appropriate to simulate normal ATM functions. A key concept which was included in our iteration was to make use of multi-threading to resolve conflicts within the system with regards to the “data race” problem. We decided to make a fully implemented ATM simulator complete with multi-threading, a “data race” problem and solution and the necessary tools to demonstrate them.

# Features

The Project is a GUI based ATM system. There is a main Bank system from where each type of ATM can be launched.

The three types are: standard ATM, ATM with “data race” problem and ATM with “data race” solution. In concept all 3 ATM types would work the same; Login Display where user uses number-pad to enter account number and pin, with validation so only correct combinations can login; From the login page comes the ATM menu where user can choose to Display, Withdraw or exit the ATM; Display simply displays the balance, Withdraw allows user to withdraw 10, 50, 500 from the account, exit simply logs out the account and returns to the login display.

The Bank system allows for the generation of 2 ATM systems of each type allowing for the testing of the “data race” problem and solution.

# Design

Prior to the implementation stage, we first created a UI design to base the implementation on; we went through the sample code provided, taking the key code fragments.

After designing the ATM system, we split the implementation up into functions which would be equally distributed between us for implementation.

The first part of the implementation was to convert the sample code/logic into a GUI based program ATM system. A central “Bank” System was then implemented which would act like a main menu which would allow the system to choose simulating a Normal, Data-race problem or Data-race solution ATM.

Multi-threading was then added to the System to allow for simultaneous use of more one ATM on the same Account. The “data race” demonstration version of the ATM system was implemented with the problem and another with the solution.

# Problems Encountered and Solutions

The first problem encountered was to do with the GUI where there was a problem in how to make the UI match the design, all elements of the GUI were visible even though they were not needed. This was solved by changing the visibility of each GUI item making them invisible, where only the needed GUI items in an operation were made visible.

Another problem encountered was with the implementation of the multi-threading with its parameter. This issue had a quick fix in that this was due to the protection levels of certain classes and functions and so the appropriate changes were made to the protections to allow for the multi-processing to occur.

There was a problem in implementing the “Data race” problem, even though there was a delay programmed into the system, the “data race” would not occur. The was solved in that the artificial delay was in the wrong place to simulate the “data race”, the delay should’ve occurred before the change is made to the balance as at that point the other thread would use the same balance(unchanged as the transaction is delayed) even though there is made a transaction on it as the delay caused both threads to use the same balance as the first threads transaction had not processed resulting in the data race problem.

Matthew Aston – 170020225

Andrew Hart – 160013951

Afzal Miah - 170010026