Practical Assignment – COMP 346

Dimitre Bogdanov: 4000 4252

Miqdad Amirali: 4000 4201

**Task 1: Atomicity bug hunt**

The main reason why the requirement is not met is because the operations done on multiple threads trying to access the same object are not done in sequence, where one would wait for the previous to finish, instead they access and modify the object as soon as the system allows them to. This situation causes the data in the account objects to be inconsistently modified. The bug that is causing it is the fact that the `Account.debosit` and `Account.withdraw` methods are not synchronized, allowing concurrent access to the object which causes data corruption.

**Task 2: Starting Order**

The Thread Scheduler determines the start order of the threads at an unpredictable random order. A thread is started upon the invocation of the `.start()` method on the given thread. Upon start, the thread is in a Runnable state, it waits for the thread scheduler to pick it up. It's status then becomes as Running until an I/O block or end of the thread. A thread when in a Non-runnable state means that it is alive but not eligible to run.

**Task 3: Method level synchronization**

Refer to code

**Task 4: Block level synchronization**

Refer to code

**Task 5: synchronized block vs synchronized methods**

Synchronizing at the block level, when properly done, allows you to synchronize the data that you need and prevent any race condition or data corruption, without the need to lock the entire object which is what happens when you do a method level synchronization