**Programming Assignment 1**

**Tianlin Yang 40010303**

**Gaoshuo Cui 40085020**

**Task 1.**

**Explain why the main requirement above (i.e. consistent state of the account array) is not met. What atomicity problem does it pose? Find the bug that causes it. In no more than three sentences, explain what went wrong.**

It because in the provided code, multiple threads (‘withdraw’ & ‘deposit’) a share of the processing unit(‘balance’). It takes risk when threads are being kicked out of the running state to ready state (context switch) time to time to give other threads a chance to run.[[1]](#endnote-1) Therefore, the value updated in ‘balance’ are non-atomically.

**Is there any way that** **two operations can write at the same time?**

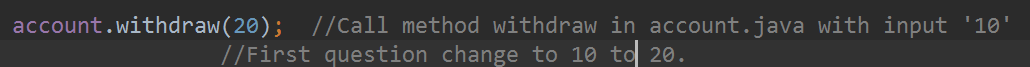
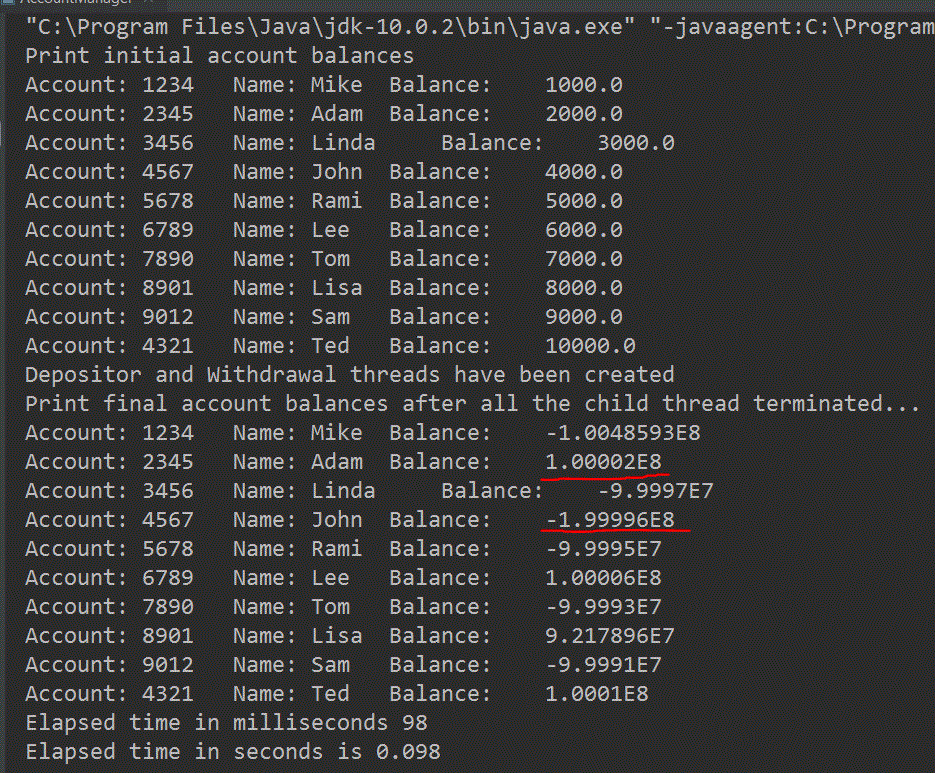
For write signal file at the same time, Java can map several regions of one file to different buffers, each buffer can be filled separately by a separate thread.[[2]](#endnote-2)

For two files, program can write two files at same time by parallel running two threads.

For same variable, if we have 2 threads running at the same time for example 22:00:15s, Multi cores computer can process these 2 threads at the same time only if there is no control of the interrupt. Therefore, in JVM it will be scheduled and executed as an unpredicted order.

**What is happened if you put X=10 CAD and Y=20 CAD. Modify the code, run it and see the results. Explain the reason no more than three sentences. What is the equivalent situation in operating system?**

When modify the withdraw Y=10 to Y=20, the result as shown below:

****

The ‘deposit’ and ‘withdraw’ actions do not have synchronized statement, the ‘balance’ will have correctness risk here. It’s the corruption to shared data situation.[[3]](#endnote-3)

**Task 2.**

**Explain, in about one sentence,** **what determines the start order of the threads. Also, very briefly, explain the lifetime of a thread: its creation, execution, and termination. Experiment with the start order of any of the threads. Is the consistency of the accounts preserved?**

The start order of the threads are determined by the JVM and/or operating system, this order does not have to be the same order in the code where they have located.

1. Creation: the thread is created in a ‘new’ state but has not been processed yet.
2. Runnable: In the JVM, the thread is executing after call the ‘Thread.start()’ method, but it may be waiting for other resources from operating system such as processor. Which means the scheduler has not selected it yet.
3. Running: When scheduler select the thread, it is in a running state.
4. Blocked: Current thread is waiting for another thread to release the lock.
5. Terminated: The thread is in a terminated state when its run() method exits.

The consistency of the accounts is not preserved, because the start order of the threads has been changed.[[4]](#endnote-4)

**Task 3.**

Attached all files in task3a and task3b.

**Task 4.**

Attached all files in task4a and task4b.

**Task 5.**

**Considering the results of task 3A vs task 4A, what is the advantage of synchronized block over synchronized method?**

1. Synchronized block can only focus on ‘critical section’ of a method, skip the rest of sections in the method. This improves performance drastically because locking is only happened when it is needed. [[5]](#endnote-5)
2. Synchronized block provide granular control over lock, as you can use arbitrary any lock to provide mutual exclusion to critical section code.

**Reference**

1. COMP346 Tutorial-2 Synchoronization.pdf, 2018 Fall, Page 19 [↑](#endnote-ref-1)
2. <https://docs.oracle.com/javase/1.5.0/docs/api/java/nio/channels/FileChannel.html> [↑](#endnote-ref-2)
3. COMP346 Tutorial-3 Synchoronization.pdf, 2018 Fall, Page 3 [↑](#endnote-ref-3)
4. COMP346 Tutorial-1 Processes and Threads.pdf, 2018 Fall, Page 12.

   And <https://www.javatpoint.com/life-cycle-of-a-thread> [↑](#endnote-ref-4)
5. COMP346 Tutorial-2 Synchoronization.pdf, 2018 Fall, Page 26.

   And <https://stackoverflow.com/questions/20906548/why-is-synchronized-block-better-than-synchronized-method> [↑](#endnote-ref-5)