Avnish Patel
40024628
Comp 346 Programming Assignment 1

Task 1: Atomicity Bug Hunt

The reason for the thread execution in this program is not atomic is because there is nothing stopping one thread interfering with another thread mid-execution. Let's take an example, when one thread starts it has several operations to perform within its given run method, yet there will be a context switch before it finishes execution. Thus there is no atomicity in that "interference" and this is the reason it effectively leads to memory inconsistency errors.

Task 2: Starting Order

The threading mechanism of the operating system determines the start order of the threads. It is software that will determine which thread to run first and for how long before it gets interrupted by another thread and so on.

Task 3: Critical section

It is two lines highlighted in yellow colour. It is also underlined,boled,italic.

```
public void debosit(double amount){

                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                double k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;

                        balance = balance + amount;

                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;

        }

        /**
        * A method that allows a customer to withdraw money from this account
        * @param amount A double that represents a withdrawal amount
        */
        public void withdraw(double amount){

                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                double k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
                        balance = balance - amount;

                // Waste some time doing fake computations
                // do not remove or modify any of the following 3 statements
                k = 999999999;
                for(int i=0;i<100;i++)
                        k = k / 2;
        }
```

Task 4: Method Level Synchronization

See coding “Task4”…

Task 5: Block Level Synchronization

See coding “Task5”…

Task 6: Synchronized Block vs. Synchronized Method

The advantage of a synchronized block over a synchronized method is that the synchronized block allows you to synchronize more specifically and tailor the code (the critical section), allowing for more throughput. Compared to method synchronization which locks the current instantiated object for entire duration of the method, when perhaps only part of it needs to be synchronized. Block level synchronization is more efficient in its ability to potentially produce more throughput from a given section of code.