**Task 1: Atomicity Bug Hunt**

The reason thread execution in this program is not atomic is because there is nothing stopping one thread interfering with another thread mid-execution. For example, when one thread starts, its has several operations to perform within its given run method, yet there will be a context switch before it finishes execution and thus there is no atomicity in that “interference” and this is what effectively leads to memory inconsistency errors.

**Task 2: Starting Order**

What determines the start order of the threads is the threading mechanism of the

operating system. Basically it is software that will determine which thread to run first and for

how long before it gets interrupted by another, and so on.

**Task 3: Method Level Synchronization**

See Workspace Package “Task3”…

**Task 4: Block Level Synchronization**

See Workspace Package “Task4”…

**Task 5: 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 the current instantiated object for the entire duration of the method, when perhaps only part of it needs to be synchronized. Again, block level synchronization is more efficient in its ability to potentially produce more throughput from a given section of code.