**Coding Assignment Solution – Nevelex Corporation**

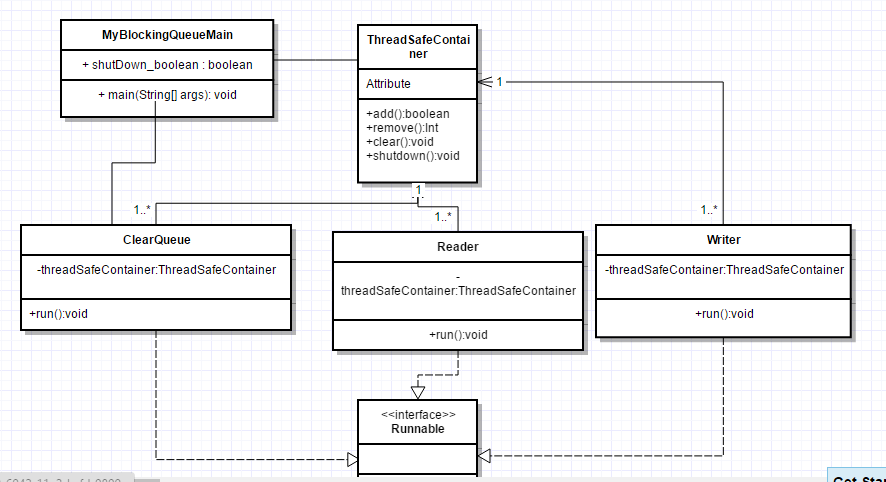
1.b Explain how your design is Thread Safe? Is it safe for n readers and m writers?

I have used synchronization while adding, removing, clearing or using shutdown method from the queue class I developed. Hence at a point of time only one thread which has access lock on the queue object would be able to perform changes. Hence my design is thread safe.

It is safe for any number of Readers and Writers.

1.c Test cases are included in the code. I have used Junit for testing. I was able to write some test cases and the remaining test cases are performed manually.

1.d UML Class Diagram



2. The normal value is an int and hence it is not Thread Safe. There is a definite chance in a multi-threading environment that two threads try to update it at the same time. There is no synchronization and hence it results in unexpected result. Synchronizing the increment method is one way to fix it, however AtomicInteger in java is the best way to fix it. The code after refactoring is as follows:

package volatiletest;

import java.util.concurrent.atomic.AtomicInteger;

public class VolatileTest {

private volatile AtomicInteger value;

public void increment () {

value.incrementAndGet();

}

public int square () {

return value.get()\*value.get();

}

}

3.

* 1. How can you determine which one is missing?

**Approach 1:**

I will solve it with two for loops. The first for loop sums the integers from 1 to 1,000,000 inclusive. I store this in a long variable called total. Then, I will add the values in the array by parsing once and store in another long variable called sum. The difference of sum and total is the missing number.

**Approach 2:**

The same can be done with XOR. First doing an XOR of 1 to 1,000,000 and storing in sum variable and then doing XOR of this variable with all the values of the array. The resultant sum after these two loops will have the missing element.

* 1. Can you think of a way to do it while iterating through the array only once.?

Both the ways mentioned above parse the array only once.

* 1. Is overflow a problem in the solution? Why not?

For Approach 1, overflow is a problem if we maintain the variable sum as int otherwise if its declared as long, overflow is not a problem.

For Approach 2, overflow straightaway is not a problem as XOR of a set of numbers would never got out of bound as it’s a bitwise operation and no new bits are created by this operation.