**Programming Assignment 1**

You are advised to develop a generic, single-threaded queue, to get yourself ready to complete the multithreaded portions of the assignment. Your queue should have the ability to enqueue and dequeue objects of any type. I recommend you develop your generic queue without using outside sources, so that you fully understand how it works, but you may choose to use an outside source, so long as you cite that source.

You are taking a statistics class, and your instructor has found a website claiming a weird conspiracy theory that M&M colors are rigged, and she has asked you to determine the frequency of M&M colors from data collected by an M&M color-determination robot.

Your task is to design and implement a set of classes that process a collection of Integer objects, corresponding to an M&M color.

The integers are stored in a queue by a driver. In particular, the following code must compile error free and execute correctly on your implementation. You should implement your own generic Queue class. Although it is recommended you do not do so for this assignment, if you borrow code from elsewhere, cite your sources. Your queue must import no Java libraries.

**public** **static** **void** main(String[] args) {

Queue<Integer> collection = **new** Queue<Integer>();

**for** (**int** index = 0; index < 10000; index++) {

**int** candidate = ((**int**)(Math.*random*() \* 10000)) % 5;

Integer integ = **new** Integer(candidate);

collection.enqueue(integ);

}

RequestProcessor processor = **new** RequestProcessor(collection);

**int** numberOfThreads = 5;

Thread[] threads = **new** Thread[numberOfThreads];

**for** (**int** index = 0; index < threads.length; index++) {

threads[index] = **new** Thread(processor);

threads[index].start();

}

**try** {

**for** (**int** index = 0; index < threads.length; index++) {

threads[index].join();

}

} **catch** (InterruptedException ie) {

ie.printStackTrace();

}

ThreadStatisticsSetup.print();

}

One possible output is:

Tabulator: 1 Count 1137 for color Red=20.695303967965053%

Tabulator: 1 Count 1077 for color Brown=19.603203494721516%

Tabulator: 1 Count 1090 for color Yellow=19.839825263924283%

Tabulator: 1 Count 1094 for color Green=19.912631962140516%

Tabulator: 1 Count 1096 for color Blue=19.949035311248636%

Tabulator: 2 Count 232 for color Red=19.414225941422593%

Tabulator: 2 Count 232 for color Brown=19.414225941422593%

Tabulator: 2 Count 243 for color Yellow=20.334728033472803%

Tabulator: 2 Count 216 for color Green=18.07531380753138%

Tabulator: 2 Count 272 for color Blue=22.761506276150627%

Tabulator: 3 Count 79 for color Red=21.066666666666666%

Tabulator: 3 Count 82 for color Brown=21.866666666666667%

Tabulator: 3 Count 76 for color Yellow=20.266666666666666%

Tabulator: 3 Count 85 for color Green=22.666666666666668%

Tabulator: 3 Count 53 for color Blue=14.133333333333333%

Tabulator: 4 Count 550 for color Red=20.161290322580644%

Tabulator: 4 Count 573 for color Brown=21.00439882697947%

Tabulator: 4 Count 564 for color Yellow=20.674486803519063%

Tabulator: 4 Count 501 for color Green=18.365102639296186%

Tabulator: 4 Count 540 for color Blue=19.794721407624632%

Tabulator: 5 Count 44 for color Red=21.153846153846153%

Tabulator: 5 Count 45 for color Brown=21.634615384615383%

Tabulator: 5 Count 44 for color Yellow=21.153846153846153%

Tabulator: 5 Count 32 for color Green=15.384615384615385%

Tabulator: 5 Count 43 for color Blue=20.673076923076923%

==Totals==

Color Red composes 20.42% of the total

Color Brown composes 20.09% of the total

Color Yellow composes 20.17% of the total

Color Green composes 19.28% of the total

Color Blue composes 20.04% of the total

The program was run with 5 threads. There are 10000 M&Ms. The number of M&Ms and portion calculated by each tabulator are printed by the class ThreadStatisticsSetup.

Each thread in the RequestProcessor class must use a lock before accessing the collection. The threads must be as concurrent as possible, within reason. Each thread must have its own private area to store the total and individual color count processed by it. This private area must be supplied by ThreadStatisticsSetup. Moreover, these private areas must be stored in a queue by ThreadStatisticsSetup, so that class can easily print the queue in response to the call ThreadStatisticsSetup.print(). The generic queue can be used both for the collection of colors and the collection of private areas. Each thread must dequeue a single M&M color from the collection, process the M&M by updating the sum for the color to which that M&M corresponds. Each thread must count the colors it dequeues, not enqueue individual colors for ThreadStatisticsSetup to count. The private data area for each thread must be a count of colors, not a queue of separate M&Ms.

Your program may either use ThreadLocal for counting on each thread, or it may use separate class instances. The lock must be held only where absolutely necessary. Code that uses locks to run almost entirely in a single thread will receive very few points. You should verify that your code behaves correctly by testing it with many more than the required number of M&Ms, verifying that the totals you produce make sense, and adding a brief delay using Thread.sleep(10) (which you should remove before submission), and printing messages outside of locked code to verify that threads run concurrently. You can determine the unique id of the thread that is running by making Thread.currentThread().getId() call. The code you submit does not need to print this information.

When printing the candidates, the following colors correspond to each integer:

0: Red

1: Brown

2: Yellow

3. Green

4. Blue

If you wish, you may substitute your favorite M&M colors, or other objects that have an interest to you.

Zip the source files and submit to the dropbox. Gross violations such as missing files or not submitting a zip file will incur penalties. If the program has syntax errors, the grade will be 0: no exceptions.

**Grading**

This program will be graded based on the Program Grading Rubric. Your program should process all entries, and produce correct counts. No entry should be processed by more than one thread. The number of M&Ms should be correctly computed. Your program should adhere to the assignment description.