CSCD 467/567 Assignment 2
Threaded Prime Finder in Java

Rules: Your code must use Java Language. If your program shows a compilation error, you get a zero for this lab assignment. To avoid compatibility issues, I encourage you to upgrade your JRE to latest version of SE 1.7 or 1.8.

Submission: Wrap up all your java files and a ReadMe text file into a single zip file. Name your zip file as *FirstInitialYourLastName*CSCD467Hw2.zip. For example, if your legal name is Will Smith, you should name your zip file as wSmithCSCD467Hw2.zip.

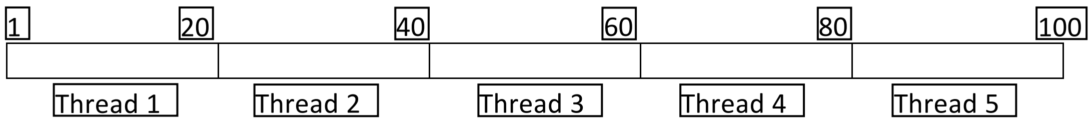
Before you leave the laboratory, please show the TA or the instructor how your program works, they will give you a score for this Lab assignment.

For archive purpose, please also submit your single zip file on EWU Canvas by following CSCD467-01 Course →Assignments→Hw2→ Submit Assignment to upload your single zip file.

Problem Description:

Based on the code provided along with this description, you are required to write a concurrent version of Prime finder. The program outputs the total number of prime within the range 1 to m using n threads. m and n are provided as command line parameters. You have to implement the following features. (The solution is partially given, please check all java classes and complete the "Write Me here!" section.)

1, You have to create n threads. Each thread *i* will search a separate subrange of numbers *(low, high)* to get a local accumulation of total *Ni* for that range. The length of each subrange is approximately (m / n);



The example above shows m = 100 and n = 5. The first thread will search the subrange of (1 through 20), the last thread will search the subrange of (81 to 100).

2, You have to create a Counter Class, which is used to add the number of primes found by each thread *Ni to* a total number. A Counter object will be shared by all threads.

3, Output the total number of prime found by all threads by calling total() method in Counter object.

4,Using the command System.currentTimeMillis() to time the execution time. You should compare the time cost of serial code with that of the concurrent version. You have to compute a **speedup ratio** = **timeCostSerialCode / timeCostConcurrentCode**.

5, Test case 1

When we use 4 threads to search range (1, 100000),

```
Time cost of serial code: 15 ms.
Time cost of parallel code: 8 ms.
The speedup ration is by using concurrent programming:  1.88.
Number prime found by serial code is: 9592
Number prime found by parallel code is 9592
```

6, Test case 2

When we use 4 threads to search range (1, 1000000),

```
Time cost of serial code: 202 ms.
Time cost of parallel code: 70 ms.
The speedup ration is by using concurrent programming:  2.89.
Number prime found by serial code is: 78498
Number prime found by parallel code is 78498
```

7, Test case 3

When we use 10 threads to search range (1, 10000000),

```
Time cost of serial code: 5237 ms.
Time cost of parallel code: 1457 ms.
The speedup ration is by using concurrent programming:  3.59.
Number prime found by serial code is: 664579
Number prime found by parallel code is 664579
```