Counting Primes

Purpose

To become more familiar with using threads.

Problem

A prime number is a number that is only evenly divisible by two things: the number 1 and the number itself. For example, 7 is prime because it is only evenly divisible by 1 and 7, while 8 is not prime because in addition to 1 and 8, it is also evenly divisible by 2 and 4. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, …

Here is a method that determines if a given number is prime or not:

```
        public static boolean isPrime(int n) {

                if (n <= 1) return false;

                if (n <= 3) return true;

                if (n % 2 == 0 || n % 3 == 0) return false;

                for (int i=5; i*i <= n; i+=6)
                        if (n % i == 0 || n % (i+2) == 0)
                                return false;

                return true;
        }
```

Write a class called PrimeThread that extends the basic Thread class. Your PrimeThread class should count the number of primes between two values, start (inclusive) and end (exclusive). For example, if your PrimeThread class is given the values 5 and 17, it should come up with the answer 4, because there are four prime numbers starting at 5 and up to but not including 17 (5, 7, 11, and 13).

Next, write a driver program that takes two command line parameters: the number of threads to use and a value n. Your program should spawn the desired number of threads and count the number of primes between 1 and n. Time how long this takes for 1, 2, 3 and 4 threads and n = 10,000,000 and compute the speedup.

Note: If you do this on repl.it you may not see any speedup, even if your code is correct. That's ok — you will be assessed only on if your code is correct with respect to spawning the threads and collecting the results.