





**QUESTION 2**

import *java*.*util*.*concurrent*.*Callable*;

import *java*.*util*.*concurrent*.*Executors*;

import *java*.*util*.*concurrent*.*ExecutorService*;

import *java*.*util*.*concurrent*.*Future*;

import *java*.*util*.*ArrayList*;

import *java*.*util*.*Scanner*;

*public* *class* Q2 {

/\*\*

*\* Write a multithreaded program to count the number of prime numbers between 1*

*\* and 10,000,000.*

\*/

*private* *static* *final* int MAX\_INT = 10\_000\_000;

*private* *static* int TOTAL\_PRIMES;

/\*\*

*\* A class CountPrimesTask that implements Callable interface, receives a range*

*\* (first, last), and calls the countPrime() method to count prime numbers*

*\* between (first, last).*

\*/

*private* *static* *class* CountPrimesTask *implements* Callable<Integer> {

int first, last;

*public* CountPrimesTask(int first, int last) {

this.*first* = first;

this.*last* = last;

}

*public* Integer call() {

int count = countPrimes(first, last);

return count;

}

}

/\*\*

*\* A method countPrimesConcurrently() that starts a number of threads to count*

*\* prime numbers concurrently. The number of threads is specified as a parameter*

*\* passed to countPrimesConcurrently(). This method returns the count of prime*

*\* numbers.*

\*/

*private* *static* void countPrimesConcurrently(int noOfThreads) {

System.*out*.println(

"\nCounting primes between 1 and " + MAX\_INT + " using " + noOfThreads + " number of threads:\n");

long startTime = System.currentTimeMillis();

ExecutorService executor = Executors.newFixedThreadPool(noOfThreads);

ArrayList<Future<Integer>> results = new ArrayList<>();

/\**\**

*\* Each sub-task processes at most 1000 integers. Use Math.ceil to round up the*

*\* number of sub-tasks to cater for MAX\_INT which is not a multiple of 1000. -1*

*\* because exceed by 1.*

\*/

int numberOfSubTasks = (int) Math.ceil(MAX\_INT / 1000.0);

for (int i = 0; i < numberOfSubTasks; i++) {

int start = i \* 1000 + 1;

int end = (i + 1) \* 1000;

/\*

*\* The last task in that case will consist of the last (MAX\_INT%1000)) ints.*

\*/

if (end > MAX\_INT)

end = MAX\_INT;

CountPrimesTask subTask = new CountPrimesTask(start, end);

Future<Integer> subResult = executor.submit(subTask);

results.add(subResult);

}

executor.shutdown();

/\**\**

*\* A method addToCount() that allows running threads to add their count of prime*

*\* numbers to a global variable total. Total indicates total prime numbers*

*\* counted so far.*

\*/

addToCount(results);

long elapsedTime = System.currentTimeMillis() - startTime;

/\**\**

*\* Subtract 1 from TOTAL\_PRIMES due to excess in calculation*

\*/

System.*out*.println("The number of primes is " + (TOTAL\_PRIMES - 1) + ".");

System.*out*.println("Total elapsed time: " + (elapsedTime / 1000.0) + " seconds.\n");

}

/\*\*

*\* A method addToCount() that allows running threads to add their count of prime*

*\* numbers to a global variable total. Total indicates total prime numbers*

*\* counted so far.*

\*/

*private* *static* void addToCount(ArrayList<Future<Integer>> results) {

for (Future<Integer> res : results) {

try {

TOTAL\_PRIMES += res.get();

} catch (Exception e) {

}

}

}

/\*\*

*\* A method countPrimes() that returns the count of prime numbers within a given*

*\* range (first, last).*

\*/

*private* *static* int countPrimes(int first, int last) {

int count = 0;

for (int i = first; i <= last; i++)

if (isPrime(i))

count++;

return count;

}

/\*\*

*\* A method isPrime() that returns True if a given number is a prime number, or*

*\* False otherwise.*

\*/

*private* *static* boolean isPrime(int x) {

/\**\**

*\* Use mathematical algorithm to quickly check if number is prime*

\*/

int limit = (int) Math.sqrt(x);

for (int i = 2; i <= limit; i++)

if (x % i == 0)

return false;

return true;

}

/\*\*

*\* A main method that declares and initialises all the constants and variables,*

*\* gets and validates the number (1-8) of threads the user would like to use,*

*\* and calls countPrimesConcurrently() to start the process of counting prime*

*\* numbers*

\*/

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

Scanner scanner = new Scanner(System.*in*);

int numberOfThreads = 0;

System.*out*.print("Enter number of threads (1-8): ");

while (numberOfThreads < 1 || numberOfThreads > 8) {

numberOfThreads = scanner.nextInt();

if (numberOfThreads < 1 || numberOfThreads > 8)

System.*out*.println("Please enter a number in the range 1 to 8!");

}

countPrimesConcurrently(numberOfThreads);

scanner.close();

}

}