For: Melissa Bakke

Assignment: Exercise 22.14 Execution time for prime numbers

|  |
| --- |
| **Screenshot(s)** |
|  |
|  |

|  |
| --- |
| **Code** |
| *import java.util.ArrayList;*  *import java.util.List;*  */\*\**  *\* Class: Exercise 2214*  *\* Developer: Melissa Bakke*  *\* Date: 03/06/2017*  *\* Purpose: Program that obtains the execution time for finding all the prime numbers*  *\* less than 8000000, 10000000, 12000000, 14000000, 16000000, 18000000 using algorithms in*  *\* Listing 22-5, 22-6 and 22-7.*  *\*/*  *public class Exercise2214 {*  */\*\**  *\* @param args the command line arguments*  *\*/*  *public static void main(String[] args) {*  *int[] numbersArray = {8000000, 10000000, 12000000, 14000000, 16000000, 18000000};*    *// Print table header*  *System.out.printf(" %9d %9d %9d %9d %9d %9d\n", numbersArray[0], numbersArray[1], numbersArray[2], numbersArray[3], numbersArray[4], numbersArray[5]);*    *// Label for listing 22.5 and loop to print out execution times*  *System.out.print("Listing 22.5");*  *for (int i = 0; i < numbersArray.length; i++) {*  *System.out.printf("%9d ", primes22\_5(numbersArray[i]));*  *}*    *// Label for listing 22.6 and loop to print out execution times*  *System.out.print("\nListing 22.6");*  *for (int i = 0; i < numbersArray.length; i++) {*  *System.out.printf("%9d ", primes22\_6(numbersArray[i]));*  *}*    *// Label for listing 22.7 and loop to print out execution times*  *System.out.print("\nListing 22.7");*  *for (int i = 0; i < numbersArray.length; i++) {*  *System.out.printf("%9d ", primes22\_7(numbersArray[i]));*  *}*    *System.out.println("\n");*  *}*    *// Method to get the execution times of the code in Listing 22-5 based on numbers from array*  *public static long primes22\_5(int n){*  *int number = 2; // a number to be tested for primeness*    *// Get current time for start time*  *long startTime = System.currentTimeMillis();*  *// repeatedly find prime numbers*  *while (number <= n) {*  *// assume number is prime*  *boolean isPrime = true;*    *// test if number is prime*  *for (int divisor = 2; divisor <= (int)(Math.sqrt(number)); divisor++) {*  *if (number % divisor == 0) {*  *isPrime = false;*  *break;*  *}*  *} // end for*    *// check if next number is prime*  *number++;*  *} // end while*    *// Get current time for end time*  *long endTime = System.currentTimeMillis();*  *// Return execution time*  *return endTime - startTime;*  *}*    *// Method to get the execution times of the code in Listing 22-6 based on numbers from array*  *public static long primes22\_6(int n){*  *// A list to hold prime numbers*  *List<Integer> list = new ArrayList<>();*    *int number = 2;*  *int squareRoot = 1;*    *// Get current time for start*  *long startTime = System.currentTimeMillis();*    *// Repeatedly find prime numbers*  *while (number <= n) {*  *// Assume number is prime*  *boolean isPrime = true;*    *if (squareRoot \* squareRoot < number) {*  *squareRoot++;*  *}*    *// ClosestPair if number is prime*  *for (int k = 0; k < list.size() && list.get(k) <= squareRoot; k++) {*  *if (number % list.get(k) == 0) {*  *isPrime = false;*  *break;*  *}*  *}*    *// If prime, add to list*  *if (isPrime) {*  *list.add(number);*  *}*  *// Increment to check next number*  *number++;*  *}// end while*    *// Get current time for end*  *long endTime = System.currentTimeMillis();*  *// Return execution time*  *return endTime - startTime;*  *}*    *// Method to get the execution times of the code in Listing 22-7 based on numbers from array*  *public static long primes22\_7(int n){*  *boolean[] primes = new boolean[n + 1]; // Prime number sieve*    *// Get current time for start*  *long startTime = System.currentTimeMillis();*    *// Initialize primes[i] to true*  *for (int i = 0; i < primes.length; i++) {*  *primes[i] = true;*  *}*    *for (int k = 2; k <= n / k; k++) {*  *if (primes[k]) {*  *for (int i = k; i <= n / k; i++) {*  *primes[k \* i] = false; // k \* i is not prime*  *}*  *}*  *}*    *// Get current time for end*  *long endTime = System.currentTimeMillis();*  *// Return execution time*  *return endTime - startTime;*  *}*  *}* |
|  |