Reti e Laboratorio III
Modulo Laboratorio III
AA. 2025-2026

docente: Laura Ricci

laura.ricci@unipi.it

Correzione Assignment 1

"Non è tutto oro quello che luccica"

13/10/2025

ASSIGNMENT 1: MULTITHREADING

- *non è tutto oro quello che luccica….ovvero, non sempre il multithreading è conveniente….*
- scrivere una applicazione JAVA che
  - crea e attiva n thread.
  - ogni thread esegue esattamente lo stesso task, ovvero conta `il numero di interi minori di 10.000.000 che sono primi`
  - il numero di thread che devono essere attivati e mandati in esecuzione viene richiesto all'utente, che lo inserisce tramite la `CLI (Command Line Interface)`
- analizzare come varia il tempo di esecuzione dei thread attivati a seconda del loro numero
- sviluppare quindi un programma in cui si creano n task, tutti eseguono la computazione descritta in precedenza e vengono sottomessi a un threadpool di dimensione prefissata

ASSIGNMENT 1: MULTITHREADING

```
public class CountPrimesThread extends Thread {
      /* Ogni thread è identificato da un identificatore
        MAX è la quantità di numeri da controllare */
      int id;
      int MAX;
      public CountPrimesThread(int id, int max)
         {this.id=id;
          this.MAX=max;}
      public static int countPrimes(int min, int max) {
           int count = 0;
           for (int i = min; i <= max; i++)
               if (isPrime(i))
                    count++;
           return count;
       }
```

ASSIGNMENT 1: MULTITHREADING

```
/* test di primalità:
 * dato un numero di input n, si deve verificare se esiste un intero m
 * compreso tra 2 e n – 1 tale da dividere n.
 * se n è divisibile per almeno un m allora n è composto, altrimenti è
 * primo. Il limite n-1 può essere abbassato alla radice quadrata di n
 * in quanto se tutti i fattori fossero maggiori di questo valore,
* il loro prodotto sarebbe necessariamente maggiore di n, che è
   assurdo */
  private static boolean isPrime(int x) {
      assert x > 1;
      int top = (int) Math.sqrt(x);
      for (int i = 2; i <= top; i++)
          if ( x % i == 0 )
              return false;
      return true;
```

ASSIGNMENT 1: MULTITHREADING

```
 public void run() {
   long startTime = System.currentTimeMillis();
   int count = countPrimes (2,MAX);
   long elapsedTime = System.currentTimeMillis() - startTime;
   System.out.println("Thread " + id + " counted " +
           count + " primes in " + (elapsedTime/1000.0) + " seconds.");
    }
}
```

ASSIGNMENT 1: MULTITHREADING

```
import java.util.Scanner;

  /* un programma che esegue diversi thread, ognuno dei quali esegue la
*    stessa computazione . */

  public class PrimeCounter {

   private final static int MAX = 10_000_000;

   /* ogni thread conta i numeri primi compresi tra 2 e MAX.

    * il numero di threads, tra 2 e 30, deve essere dato in input dal

    * programmatore */

   public static void main(String[] args) {

       int numberOfThreads = 0;

       Scanner sc = new Scanner(System.in);

       System.out.print("How many threads do you want to use  (from 1 to 30)
                        ?  ");

       numberOfThreads = sc.nextInt();
```

ASSIGNMENT 1: MULTITHREADING

```
while (numberOfThreads < 1 || numberOfThreads > 30) {
            System.out.println("Please enter a number between 1 and 30 !");
            numberOfThreads = sc.nextInt();
        }
    System.out.println("\nCreating " + numberOfThreads + " prime-counting
                                                          threads...");
    CountPrimesThread[] worker = new CountPrimesThread[numberOfThreads];
    for (int i = 0; i < numberOfThreads; i++)
        worker[i] = new CountPrimesThread( i, MAX);
    for (int i = 0; i < numberOfThreads; i++)
        worker[i].start();
    System.out.println("Threads have been created and started.");
    sc.close();
    } }
```