Reti e Laboratorio 3
Modulo Laboratorio 3
A.A. 2024-2025

docente: Laura Ricci

laura.ricci@unipi.it

Correzione Assignment 6
”Compressione di Files”
15/11/2024

ASSIGNMENT 6: COMPRESSIONE DI FILES

- scrivere un programma che, dato in input una lista di directories, comprima tutti i file in esse contenuti con l'utility gzip

- ipotesi semplificativa:
  - zippare solo i file contenuti nelle directories passate in input,
  - non considerare ricorsione su eventuali sottodirectories
  - il riferimento ad ogni file individuato viene passato ad un task, che deve essere eseguito in un threadpool
  - individuare nelle API JAVA la classe di supporto adatta per la compressione
- NOTA: l'utilizzo dei threadpool è indicato, perchè i task presentano un buon mix tra I/O e computazione I/O heavy:
  - I/O intensive: tutti i file devono essere letti e scritti
  - CPU-intensive: la compressione richiede molta computazione

facoltativo: comprimere ricorsivamente i file in tutte le sottodirectories

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
package ZipFiles;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.zip.GZIPOutputStream;

/**

 * Questa classe rappresenta il generico thread Compressor

 * che effettua la compressione in formato GZIP di un file.

 * @author Matteo Loporchio

 */

public class Compressor implements Runnable {

    // Dimensione del buffer per la compressione.

     public static final int BUFSIZE = 2048;

     // Riferimento al file da comprimere.

     private File f;
```

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
/**
 * Metodo costruttore.
 * @param f riferimento al file da comprimere
 */
public Compressor(File f) {
      this.f = f;
    }
```

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
public void run() {
    System.out.printf("Compressing file: %s\n", f.getName());
    File outputFile = new File(f.getAbsolutePath() + ".gz");
    try (
       FileInputStream is = new FileInputStream(f);
       GZIPOutputStream zs = new GZIPOutputStream(new FileOutputStream(outputFile));)
         {
            byte[] buffer = new byte[BUFSIZE];
            int len;
            while ((len = is.read(buffer)) != -1) zs.write(buffer, 0, len);
            System.out.printf("Done compressing file: %s\n", f.getName());
        } catch (IOException e) {
             System.err.printf("Error while compressing file: %s\n", f.getName());
             e.printStackTrace(); } } }
```

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
package ZipFiles;
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.*;

/**
 * Reti e laboratorio 3 - A.A. 2024/2025
 * @author Matteo Loporchio
 */
public class Main {
    // Numero di thread del pool.
    public static final int N_THREADS = 8;
    // Pool di thread (di dimensione fissa).
    public static ExecutorService pool = Executors.newFixedThreadPool(N_THREADS);
```

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
public static void main(String[] args) {
   // Creo un nuovo oggetto Scanner per leggere l'input da tastiera.
   Scanner scanner = new Scanner(System.in);
   List<String> directories = new ArrayList<>();
   while (true) {
        // Leggo il percorso della directory dall'input.
        System.out.printf("Insert a directory: ");
        String dirName = scanner.nextLine();
        // Se ho letto il comando "\end", esco dal ciclo.
        if (dirName.equals("\\end")) break;
        directories.add(dirName);
    }
```

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
// A questo punto, scorro la lista una directory alla volta.
for (String dirName : directories) {
  // Apro la directory (e controllo che effettivamente si tratti di una directory).
   File directory = new File(dirName);
   if (!directory.exists() || !directory.isDirectory()) {
       System.err.printf("Error: %s is not a valid directory!\n", dirName);
       continue;
    }
// Leggo tutti i file in essa contenuti e per ciascuno
// creo ed eseguo un thread Compressor inviandolo al pool.
File[] contents = directory.listFiles();
for (File f : contents) {
    // Ignoriamo tutte le sottodirectory e i file nascosti.
    if (f.isDirectory() || f.isHidden()) continue;
       pool.submit(new Compressor(f));
  }
```

ASSIGNMENT 6: COMPRESSIONE DI FILES

```
  // Avvio la terminazione del pool.
  System.out.printf("Shutting down...");
   pool.shutdown();
   // Chiudo lo scanner.
    scanner.close();
}
}
```