

Reti e Laboratorio III

Modulo Laboratorio III

AA. 2023-2024

docente: Laura Ricci

laura.ricci@unipi.it

Correzione Assignment 3

”Simulazione Laboratorio di Informatica”

26/10/2023

ASSIGNMENT 3: SIMULAZIONE LABORATORIO INFORMATICA

Il laboratorio di Informatica del Polo Marzotto è utilizzato da tre tipi di utenti, studenti, tesisti e professori ed ogni utente deve fare una richiesta al tutor per accedere al laboratorio. I computers del laboratorio sono numerati da 1 a 20. Le richieste di accesso sono diverse a seconda del tipo dell'utente:

- i professori accedono in modo esclusivo a tutto il laboratorio, poichè hanno necessità di utilizzare tutti i computers per effettuare prove in rete.
- i tesisti richiedono l'uso esclusivo di un solo computer, identificato dall'indice i , poichè su quel computer è installato un particolare software necessario per lo sviluppo della tesi.
- gli studenti richiedono l'uso esclusivo di un qualsiasi computer.
- i professori hanno priorità su tutti nell'accesso al laboratorio, i tesisti hanno priorità sugli studenti.
- nessuno però può essere interrotto mentre sta usando un computer.

ASSIGNMENT 3: SIMULAZIONE LABORATORIO INFORMATICA

Scrivere un programma JAVA che simuli il comportamento degli utenti e del tutor. Il programma riceve in ingresso il numero di studenti, tesisti e professori che utilizzano il laboratorio ed attiva un thread per ogni utente.

Ogni utente accede k volte al laboratorio, con k generato casualmente.

Simulare l'intervallo di tempo che intercorre tra un accesso ed il successivo e l'intervallo di permanenza in laboratorio mediante il metodo `sleep` della classe `Thread`.

Il tutor deve coordinare gli accessi al laboratorio.

Il programma deve terminare quando tutti gli utenti hanno completato i loro accessi al laboratorio. Simulare gli utenti con dei thread e incapsulare la logica di gestione del laboratorio all'interno di un monitor.

ASSIGNMENT 3 : LE CATEGORIE DI UTENTI

```
/**  
 * tipo enumerato per rappresentare le varie tipologie di utenti  
 * che accedono al laboratorio.  
 * @author Matteo Loporchio  
 */  
public enum Categoria {  
    PROFESSORE,  
    TESISTA,  
    STUDENTE  
}
```

ASSIGNMENT 3: L'UTENTE

```
import java.util.List;
import java.util.concurrent.ThreadLocalRandom;
/**
 * Questa classe rappresenta il generico Utente del laboratorio.
 * a seconda della Categoria, puo' trattarsi di uno Studente,
 * di un Tesista oppure di un Professore.
 * @author Matteo Loporchio
 */
public class User implements Runnable {
    public Categoria categoria; // Tipologia di utente.
    public int id; // Identificativo numerico dell'utente.
    public int numAccessi; // Numero di accessi previsti per l'utente.
    public long workDelay; // Tempo in cui l'utente utilizza il laboratorio
    public long breakDelay; // Tempo che intercorre fra un accesso e l'altro.
    public int maxAccessi = 5; // Massimo numero di accessi,
    public long maxWork = 5000; // Massimo tempo di lavoro
    public long maxBreak = 2000; // Massimo tempo di pausa
    private Laboratorio lab; // Riferimento al laboratorio.
```

ASSIGNMENT 3: L'UTENTE

```
/**
 * Costruttore della classe Utente.
 * @param categoria la categoria dell'utente
 * @param id identificativo numerico dell'utente
 * @param lab riferimento al laboratorio
 */
public User(Categoria categoria, int id, Laboratorio lab) {
    this.categoria = categoria;
    this.id = id;
    this.lab = lab;
    numAccessi = ThreadLocalRandom.current().nextInt(1, maxAccessi+1);
    workDelay = ThreadLocalRandom.current().nextLong(maxWork+1);
    breakDelay = ThreadLocalRandom.current().nextLong(maxBreak+1);
}
```

ASSIGNMENT 3: L'UTENTE

```
/**
 * Tutti gli utenti richiedono l'accesso al laboratorio, lo utilizzano per un certo
 * intervallo di tempo e poi escono per fare una pausa. Tutto viene ripetuto per
 * numAccessi` volte.
 */
public void run() {
    try {
        for (int i = 0; i < numAccessi; i++) {
            List<Integer> assegnati = lab.entrata(this);
            Thread.sleep(workDelay);
            lab.uscita(this, assegnati);
            Thread.sleep(breakDelay); }
        }
    catch (InterruptedException e) { System.out.printf("%s con id=%d interrotto.\n",
        categoria.name(), id); return; }
    System.out.printf("%s con id=%d ha terminato il lavoro.\n",
        categoria.name(), id); }}
```

ASSIGNMENT 3: IL LABORATORIO

```
import java.util.*;

public class Laboratorio {

    public final int numComputer = 20; // Numero di computer nel laboratorio (costante).
    public final int idComputerTesiisti = 19; // Identificativo del computer richiesto dai
                                                tesisti.

    private List<Thread> thread; // Lista dove memorizzo i riferimenti ai thread.

    // Array per tenere traccia dei computer liberi e occupati.
    // L'elemento i-esimo e' false se e solo se il computer e' libero.

    private boolean[] computer;

    // Utilizzo questi contatori per tenere traccia di professori
    // e tesisti in attesa di entrare nel laboratorio.

    private int profWaiting = 0;
    private int tesiWaiting = 0;
```

ASSIGNMENT 3: IL LABORATORIO

```
/**  
 * Costruttore della classe Laboratorio  
 * che inizializza tutte le strutture dati.  
 */  
  
public Laboratorio() {  
    thread = new ArrayList<>();  
    computer = new boolean[numComputer];  
}
```

ASSIGNMENT 3: IL LABORATORIO

```
public void start(int numProf, int numTesiisti, int numStudenti) {  
    System.out.println("Laboratorio aperto.");  
    for (int i = 0; i < numProf; i++) // Creo gli utenti, un thread per ognuno di essi.  
        {thread.add(new Thread(new User(Categoria.PROFESSORE, i, this))); }  
    for (int i = 0; i < numTesiisti; i++)  
        {thread.add(new Thread(new User(Categoria.TESIISTA, i, this))); }  
    for (int i = 0; i < numStudenti; i++)  
        {thread.add(new Thread( new User(Categoria.STUDENTE, i, this))); }  
    // Per simulare l'arrivo degli utenti in ordine casuale, eseguo uno shuffle della  
    // lista di therad prima di avviarli.  
    Collections.shuffle(thread, new Random(System.currentTimeMillis()));  
    for (Thread t : thread) t.start();  
    // Attendo la terminazione di tutti i thread usando la join().  
    for (Thread t : thread)  
        { try {t.join();} catch (InterruptedException e)  
            {System.err.println("Interruzione durante l'attesa dei thread!");}  
        } System.out.println("Laboratorio chiuso."); }  
}
```

ASSIGNMENT 3: ENTRATA PROFESSORI

```
public synchronized List<Integer> entrata(User u) throws InterruptedException {  
    List<Integer> assegnati = new ArrayList<>();  
    System.out.printf("%s con id=%d in attesa di entrare.\n", u.categoria.name(),u.id);  
    // Quindi procedo in maniera diversa a seconda del tipo di utente.  
    switch (u.categoria) {  
        // I professori attendono finche' tutti i computer non  
        // sono disponibili e quindi occupano tutto il laboratorio.  
        case PROFESSORE:  
            profWaiting++;  
            while (!libero()) wait();  
            profWaiting--;  
            for (int i = 0; i < computer.length; i++) {  
                computer[i] = true;  
                assegnati.add(i);  
            }  
            break
```

ASSIGNMENT 3: ENTRATA TESISTI

```
// I tesisti occupano sempre uno specifico computer.
```

```
case TESISTA:
```

```
    tesiWaiting++;
```

```
    while (profWaiting > 0 || computer[idComputerTesisti])
```

```
        wait();
```

```
    tesiWaiting--;
```

```
    computer[idComputerTesisti] = true;
```

```
    assegnati.add(idComputerTesisti);
```

```
    break;
```

ASSIGNMENT 3: ENTRATA STUDENTI

```
// Gli studenti occupano il primo computer libero.
```

```
case STUDENTE:
```

```
    int id = primoComputerLibero();
```

```
    // Lo studente attende finche' ci sono professori che stanno aspettando, oppure se non  
    // ci sono computer disponibili o se il computer assegnato e' quello dei tesisti e ci  
    // sono gia' tesisti prenotati per l'entrata.
```

```
    while (profWaiting > 0 || id == -1 || (tesiWaiting > 0 && id == idComputerTesisti)) {
```

```
        wait();
```

```
        id = primoComputerLibero();
```

```
    }
```

```
    computer[id] = true;
```

```
    assegnati.add(id);
```

```
    break;
```

```
    default: break; }
```

```
System.out.printf("%s con id=%d entrato.\n", u.categoria.name(), u.id);
```

```
return assegnati; }
```

ASSIGNMENT 3: USCITA UTENTI

```
/**
 * Metodo invocato dall'utente all'uscita del laboratorio.
 * @param u l'utente che desidera uscire dal laboratorio
 * @param occupati lista con gli id dei computer da liberare
 */
public synchronized void uscita(User u, List<Integer> occupati) {
    // Libero tutti i computer occupati.
    for (Integer id : occupati) computer[id] = false;
        // Risveglio tutti gli utenti in attesa.
        // NOTA: al proprio risveglio, ciascun utente controllera'
        // la validita' della propria condizione di attesa.
    notifyAll();
    System.out.printf("%s con id=%d uscito.\n", u.categoria.name(), u.id);
}
```

ASSIGNMENT 3: FUNZIONI DI UTILITA'

```
/**
 * Restituisce true se e solo se tutto il laboratorio e' libero.
 * @return true se tutti i computer non sono occupati
 */
private boolean libero() {
    for (int i = 0; i < computer.length; i++)
        if (computer[i]) return false;
    return true;}

/**
 * Restituisce l'identificativo del primo computer libero.
 * @return l'id del primo computer libero
 */
private int primoComputerLibero() {
    for (int i = 0; i < computer.length; i++)
        if (!computer[i]) return i;
    return -1;
}
```

ASSIGNMENT 3: FUNZIONI DI UTILITA'

```
public static void main(String[] args) {  
    // Verifica e parsing dei parametri da riga di comando.  
    if (args.length < 3) {  
        System.err.println("Esegui come: Laboratorio " + "<numProf> <numTesiisti>  
            <numStudenti>");  
  
        System.exit(1);  
    }  
  
    int numProf = Integer.parseInt(args[0]),  
        numTesiisti = Integer.parseInt(args[1]),  
        numStudenti = Integer.parseInt(args[2]);  
  
    // Creo il laboratorio e faccio entrare gli utenti.  
    Laboratorio lab = new Laboratorio();  
    lab.start(numProf, numTesiisti, numStudenti);  
}  
}
```