UNIVERSITÀ DI PISA

Laboratorio di reti

Assignments Correction

Prof. Laura Ricci (ricci@unipi.it)

Assistant: Andrea De Salve (desalve@unipi.it)

10-05-2016

2015/2016

Assignments

- Alarm Clock
- Pi Greco
- Ufficio Postale
- Gestione Auto
- Gestione Lab

- WebLookUp
- Calcolatrice TCP
- Mini FTP
- Query Flickr

➔ Cover different topics

Java Concurrent

Java Basic I/O

Java Networking

Assignments

- Alarm Clock
- Pi Greco
- Ufficio Postale
- Gestione Auto
- Gestione Lab

- WebLookUp
- Calcolatrice TCP
- Mini FTP
- Query Flickr

➔ Cover different topics

Java Concurrent

Java Basic I/O

Java Networking

Gestione Laboratorio: testo

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:

a) i professori accedono in modo esclusivo a tutto il laboratorio
b) i tesisti richiedono l'uso esclusivo di un solo computer, identificato dall'indice i
c) 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.

Gestione Laboratorio: Testo

I computers del laboratorio sono numerati da 1 a 20. Le richieste di accesso sono diverse a seconda del tipo dell'utente:

a) i professori accedono in modo esclusivo a tutto il laboratorio

b) i tesisti richiedono l'uso esclusivo di un solo computer, identificato dall'indice i

c) 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.

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. Intervallo di tempo che intercorre tra un accesso ed il successivo e l'intervallo di permanenza in laboratorio mediante il metodo sleep. Il programma deve terminare quando tutti gli utenti hanno completato i loro accessi al laboratorio.

Gestione Lab: Actors

```java
public class Studente implements Runnable
{
    private Tutor tutor;
    private int hostId;
    private int userId;

    public Studente(int i, Tutor tutor)
    {
        this.tutor=tutor;
        this.userId = i;
    }
```

Studente

```java
public class Professore implements Runnable
{
    private Tutor tutor;
    private int profId;

    public Professore(int i, Tutor tutor)
    {
        this.tutor=tutor;
        this.profId = i;
    }
```

Professore

```java
public class Tesista implements Runnable
{
    private Tutor tutor;
    private int tesistaId;

    public Tesista(int i, Tutor tutor)
    {
        this.tutor=tutor;
        this.tesistaId = i;
    }
```

Tesista

- A **Runnable** object

Gestione Lab: Methods

```
public class Studente implements Runnable
{
    private Tutor tutor;
    private int hostId;
    private int userId;
```

```
public class Professore implements Runnable
{
    private Tutor tutor;
    private int profId;
```

```
public class Tesista implements Runnable
{
    private Tutor tutor;
    private int tesistaId;
```

```
private Tutor getTutor() {
    return tutor;
}
```

```
public int getHostId()
{
    return hostId;
}
```

```
public void setHostId(int pc)
{
    hostId = pc;
}
```

Gestione Lab: Run

```
public class Studente implements Runnable
{
    private Tutor tutor;
    private int hostId;
    private int userId;
```

```
public class Professore implements Runnable
{
    private Tutor tutor;
    private int profId;
```

```
public class Tesista implements Runnable
{
    private Tutor tutor;
    private int tesistaId;
```

```
public void run()
{
    try
    {
        getTutor().startStudente(this);
        System.out.println("Start Studente " + userId );
        Thread.sleep(new Random().nextInt(1000));
        getTutor().endStudente(this);
        System.out.println("End Studente " + userId );
    }
    catch(InterruptedException e)
    {
        e.printStackTrace();
    }
}
```

The Monitor: Fields

```
public class Tutor
{
    public static final int LAB_SIZE=20;
    public static final int HOST_TESISTA=5;

    boolean[] laboratorio = new boolean[LAB_SIZE];

    boolean professorePresente = false;

    int utentiInLaboratorio = 0;
    int tesistiSospesi = 0;
    int professoriSospesi = 0;
    int userPc = getAvailableHost();
```

The Monitor: Fields

```
public class Tutor
{
    public static final int LAB_SIZE=20;
    public static final int HOST_TESISTA=5;

    boolean[] laboratorio = new boolean[LAB_SIZE];

    boolean professorePresente = false;

    int utentiInLaboratorio = 0;
    int tesistiSospesi = 0;
    int professoriSospesi = 0;
    int userPc = getAvailableHost();
```

Numero di Host: 20

Host riservato a tesista: Host Id 5

The Monitor

```
public class Tutor
{
    public static final int LAB_SIZE=20;
    public static final int HOST_TESISTA=5;

    boolean[] laboratorio = new boolean[LAB_SIZE];

    boolean professorePresente = false;

    int utentiInLaboratorio = 0;
    int tesistiSospesi = 0;
    int professoriSospesi = 0;
    int userPc = getAvailableHost();
```

Get hosts from the lab:

```
private synchronized int getAvailableHost()
{
    for (int i = 0; i < laboratorio.length; i++)
    {
        if (laboratorio[i] == false)
            return i;
    }
    return -1;
}
```

The Monitor: Start/Stop Student

```
public synchronized void startStudente(Studente stud)
{

    while (professorePresente
        || professoriSospesi > 0
        || (userPc == HOST_TESISTA && tesistiSospesi > 0)
        || userPc == -1)
    {
        try
        { wait(); }
        catch (InterruptedException e)
        { }
    }

    stud.setHostId(userPc);
    laboratorio[userPc] = true;
    utentiInLaboratorio++;
}
```

```
public synchronized void endStudente(Studente stud)
{
    laboratorio[stud.getHostId()] = false;

    utentiInLaboratorio--;
    notifyAll();
}
```

The Monitor: Start/Stop Tesista

```
public synchronized void startTesista(Tesista tesista)
{

    while (professorePresente
        || professoriSospesi > 0
        || laboratorio[HOST_TESISTA] == true)
    {
        try
        {
            tesistiSospesi++;
            wait();
            tesistiSospesi--;
        }
        catch (InterruptedException e)
        { }
    }

    laboratorio[HOST_TESISTA] = true;
    utentiInLaboratorio++;
}
```

```
public synchronized void endTesista(Tesista tesista)
{
    laboratorio[HOST_TESISTA] = false;
    utentiInLaboratorio--;
    notifyAll();
}
```

The Monitor: Start/Stop Professore

```
public synchronized void startProfessore(Professore prof)
{

    while (utentiInLaboratorio > 0)
    {
        try
        {
            professoriSospesi++;
            wait();
            professoriSospesi--;
        }
        catch (InterruptedException e)
        { }
    }

    professorePresente = true;
}
```

```
public synchronized void endProfessore(Professore prof)
{
    professorePresente = false;
    notifyAll();
}
```

The Main

```
public static void main(String[] args) throws InterruptedException
{
        int nStud = 3;
        int nTesi = 3;
        int nProf = 1;
        int k = new Random().nextInt(10);
        List<Thread> threads = new ArrayList<Thread>();
        Tutor tutor = new Tutor();
```

The Main

```
    for (int i = 0; i < k; i++)
    {
        for (int j = 0; j < nStud; j++)
        {
            Thread t = new Thread(new Studente(j, tutor));
            threads.add(t);t.start();
        }
        for (int j = 0; j < nTesi; j++)
        {
            Thread t = new Thread(new Tesista(j, tutor));
            threads.add(t);t.start();
        }
        for (int j = 0; j < nProf; j++)
        {
            Thread t = new Thread(new Professore(j, tutor));
            threads.add(t);t.start();
        }
        Thread.sleep(new Random().nextInt(1000));
    }

    for (Thread t : threads){   t.join(); }

}
```