Multitheading

Corso di Laurea in Informatica

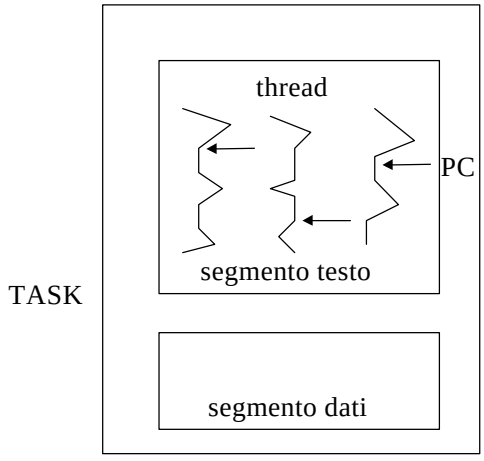
Sperimentazioni di
Linguaggi di Programmazione: Paradigmi di Programmazione

a.a. 1999/2000

Matteo Baldoni

1

Thread multipli all'interno di un Task



Un thread consite di:

- un program counter (PC);
- un insieme di registri;
- uno stack.

Condivide:

- sezione codice;
- sezione dati;
- risorse fornite dal sistema.

Non e' garantita protesione tra i thread di uno stesso task!
Ma e' lo stesso utente che li genera...

2

Come creare un thread

Creando una sottoclasse di ***Thread***

```
public class SimpleThread extends Thread {
   public void run () { … }
… }

new SimpleThread("Jamaica").start();
```

Implementando l'interfaccia ***Runnable***

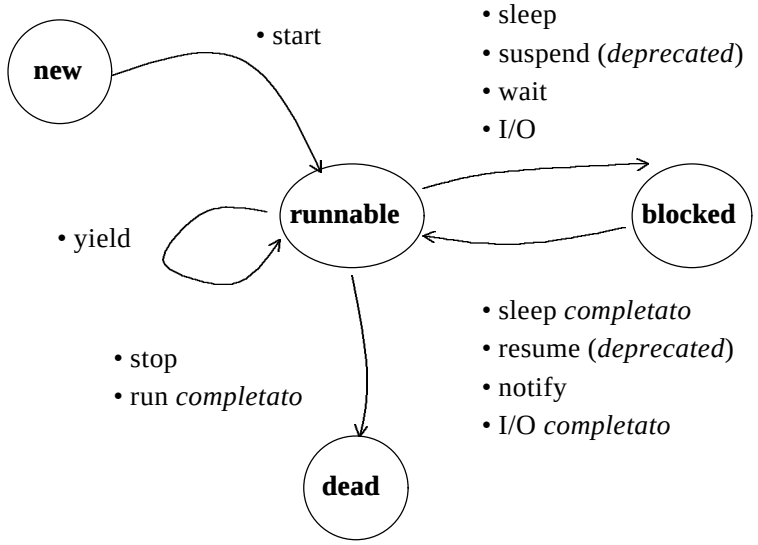
```
public class Clock extends Applet implements Runnable {
   private Thread selfThread = null;
   …
   if (selfThread = null) {
      selfThread = new Thread (this);
      selfThread.start (); }

   public void run () { … }  … }
```

Puo' ereditare da una altra classe ma un solo thread running per un particolare oggetto

3

Esempi: SimpleThread.java e Clock.java

Ciclo di vita di un thread



4

Gestire la priorita` dei thread

Algoritmo di **scheduling**: *fixed priority scheduling.*

Il sistema sceglie per l'esecuzioneil thread runnable con piu` alta priorita.
Se uguale priorita`: *round-robin*

Un thread rimane in esecuzione fino a che: *preempt*

- un thread con piu` alta priorita` diventa runnable;
- viene eseguito il metodo *yield* o *run* e` completato;
- *time-slicing* (quando supportato dal sistema)

5

Gestire la priorita` dei thread

Un thread eredita la priorita`
del suo creatore ma...

La priorita` puo` essere controllata con il metodo → setPriority(*value*)

Esempio 1: RaceTest e SelfishRunner

- time-slicing ma non sempre buona alternanza;
- ricorda: System.out.println porta a cedere CPU.

6

Gestire la priorita` dei thread

Esempio 2: RaceTest e SelfishRunner1

- time-slicing e cessione della CPU volontaria;
- migliore alternanza.

Esempio 3: RaceTest e SelfishRunner2 priorita` 2 e 3

- thread 1 non cede la CPU (solo per I/O a volte 0 stampa);
- yield non produce effetti.

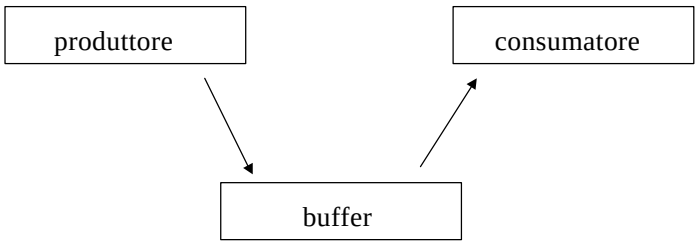
Esempio 4: RaceTest e SelfishRunner2 priorita` 2 e 3 e sleep

- thread 1 cede la CPU (per I/O e sleep);
- thread 1 vince comunque.

7

Sincronizzare i thread

Il problema del produttore e consumatore



Problema: e` necessario accedere in modo escusivo alla risorsa buffer in modo da coordinare produttore e consumatore

8

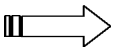
Esempio: ProducerConsumerTest.java prima versione

Sincronizzare i thread

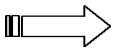
*Proteggere **regioni critiche** da eseguirsi come una operazioneatomica*

Esempio *Sharing1.java*:

L'incremento delle due variabili poiche` non eseguito in mutua esclusione potrebbe invalidare il test sulla uguaglianza dei due contatori.



Dopo alcuni passi il test dara` la non sincronizzazione dei due contatori, cosa sorprendente e inaspettata



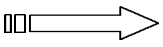
Colpa del time-slicing

9

Sincronizzare i thread

Soluzione: ***Accesso ad un oggetto in mutua esclusione!***

(almeno durante le sezioni critiche)



synchronized

```
public synchronized int get () { … }

public synchronized void put (int value) { …. }
```

10

Sincronizzare i thread

Il sistema un solo lock ad ogni istanza di un oggetto (anche se condiviso da piu` thread)

Se un thread chiama tale metodo "blocca" l'oggetto tramite il suo lock.

Altri metodi non possono chiamare metodi sinchronized sullo stesso oggetto.

Nota: ma non sinchronized SI!!
(fare attenzione a questo particolare)

11

Sincronizzare i thread

Esempio *Sharing2.java*:

```
public synchronized void run() {
  while (true) {
    t1.setText(Integer.toString(count1++));
    t2.setText(Integer.toString(count2++));
    try {
      sleep(500);
    } catch (InterruptedException e){}
  }
}
public synchronized void synchTest() {
  Sharing2.incrementAccess();
  if(count1 != count2)
    l.setText("Unsynched");
}
```

12

Esempio: *Sharing2.java*

Sincronizzare i thread

Esempio *Sharing2.java*:

Pero` in questo esempio qualcosa non funziona…

run () !!

Non rilascio mai il lock e non viene mai eseguito Test()!!

```
public synchronized void run() {
  while (true) {
    t1.setText(Integer.toString(count1++));
    t2.setText(Integer.toString(count2++));
    try {
      sleep(500);
    } catch (InterruptedException e){}
  }
}
public synchronized void synchTest() {
  Sharing2.incrementAccess();
  if(count1 != count2)
    l.setText("Unsynched");
}
```

13

Sincronizzare i thread

Esempio *Sharing2.java*: (seconda versione)

synchronized keyword:

```
synchronized (syncObject) {
   SEZIONE CRITICA
}
```

oggetto da bloccare

ma si poteva anche costruire un metodo ad hoc con sync...

```
public synchronized void run() {
  while (true) {
    synchronized (this) {
        t1.setText(Integer.toString(count1++));
        t2.setText(Integer.toString(count2++));
    }
    try {
      sleep(500);
    } catch (InterruptedException e){}
  }
}
public synchronized void synchTest() {
  Sharing2.incrementAccess();
  if(count1 != count2)
    l.setText("Unsynched");
}
```

14

Esempio: *Sharing2.java (seconda versione)*

Sincronizzare i thread

Ma in alcuni casi serve qualcosa di piu`…

A parte che non funziona (return mancante) ma in ogni caso non fa cioe` che ci aspettiamo.

deve aspettare e rilasciare la regione critica

```
public synchronized int get() {
      if (available == true) {
          available = false;
          return contents;
      }
    }

public synchronized int put(int value) {
          if (available == false) {
              available = true;
              contents = value;
          }
      }
```

15

Esempio: *ProducerConsumerTest.java*

Sincronizzare i thread

aspetta e rilascia la regione critica

risveglia i metodi in attesa

Simile al costrutto MONITOR

```
public synchronized int get() {
      while (available == false) {
          try {
              wait();
          } catch (InterruptedException e) { }
      }
      available = false;
      notifyAll();
      return contents;
  }

  public synchronized void put(int value) {
      while (available == true) {
          try {
              wait();
          } catch (InterruptedException e) { }
      }
      contents = value;
      available = true;
      notifyAll();
  }
```

16

Esempio: *ProducerConsumerTest.java con synch...*

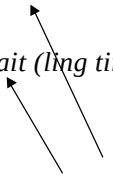
Sincronizzare i thread

Altri metodi:

• *notify()* sveglia un solo thread;

• *wait (long timeout)* svegliato o timeout;

• *wait (long timeout, int nanos)* come sopra.



Utili se si vogliono delle sleep ma con la possibilita`
di svegliare prima se occorre.

17

Ancora sui thread...

• suspend, resume e stop (deprecated): vedi tutorial;

• Deadlock: cercare di evitarlo (consiglio ovvio…);

• Gruppi di Thread: tutti I thread fanno parte di un gruppo gestibile nel suo complesso (suspend, resue, stop ma queste sono ora deprecated…)

18