Programmazione Orientata agli Oggetti in Linguaggio Java

Tecniche di Programmazione: Thread Parte a

versione 1.0

G. Mecca – Università della Basilicata – mecca@unibas.it

Sommario

- Introduzione
- Terminologia
- Thread in Java
- Problemi con i Thread

Introduzione

- Programmazione asincrona
  - ⇨lo stile della programmazione grafica
- Una caratteristica centrale di questo stile
  - ⇨l'utilizzo di fili di esecuzione ("thread") multipli per la gestione degli eventi
  - ⇨questo è particolarmente significativo all'interno di Swing che ha una gestione particolare dei thread

Introduzione

- In questa unità
  - ⇨una introduzione alla gestione dei thread in Java, con particolare riferimento a Swing
- Attenzione
  - ⇨si tratta di un argomento molto complesso
  - ⇨la programmazione concorrente
  - ⇨non saremo in grado di vederne tutti gli aspetti; la presentazione sarà mirata

Terminologia

- Sistema Operativo
  - ⇨applicazione che gestisce l'hardware fisico e si incarica, tra le altre cose, di eseguire le applicazioni
- Processo
  - ⇨applicazione in esecuzione, con il proprio spazio di memoria indipendente (codice, stack, heap)

Terminologia

- Multitasking
  - ⇨i sistemi operativi moderni sono in grado di eseguire diversi processi ("task") contemporaneamente
  - ⇨per consentire agli utenti di "fare più cose assieme"
  - ⇨nei sistemi multiprocessore, ad ogni processore viene affidato un processo
  - ⇨nei sistemi monoprocessore, si utilizza una modalità di divisione di tempo ("time sharing")
  - ⇨ad ogni processo vengono alternativamente assegnati intervalli di lavoro della CPU

Terminologia

- Gestione dei processi (“scheduling”)
  - ⇨operazione fondamentale del sistema operativo
  - ⇨strategia secondo la quale il sistema operativo esegue i processi
  - ⇨nei sistemi monoprocessore: regole secondo le quali attribuisce il tempo di CPU a ciascuno dei processi in esecuzione

Terminologia

- Strategie di scheduling
  - ⇨ce ne sono molte
  - ⇨in alcuni sistemi operativi (es: Windows NT/2000/XP) il multitasking è “preemptivo”: il Sistema Operativo interrompe il processo quando il suo tempo è scaduto, anche se non ha finito il lavoro
  - ⇨in altri sistemi operativi il multitasking è “cooperativo”: deve essere il processo a rilasciare la CPU agli altri processi

Terminologia

- Il problema dei processi
  - avviare un processo è un'operazione costosa
  - richiede di caricare il codice
  - richiede di attribuire lo heap
  - richiede di attribuire lo stack
- In particolare
  - nei linguaggi a oggetti moderni il problema è ancora più complesso
  - è necessario avviare la macchina virtuale
  - è necessario collegare dinamicamente il codice caricando i componenti necessari

Terminologia

- In molti casi
  - il livello di concorrenza che serve non richiede processi distinti ma solo "fili distinti" nello stesso processo
- Esempi nei linguaggi a oggetti
  - esecuzione del codice
  - esecuzione del garbage collector
  - cattura degli eventi che si verificano nell'interfaccia grafica

Terminologia

- Thread
  - ⇨flusso di esecuzione all'interno di un processo
  - ⇨basata su un proprio program counter e una propria pila
  - ⇨un processo può contenere vari thread che vengono eseguiti concorrentemente
  - ⇨utilizzando algoritmi di scheduling simili a quelli usati a livello di processi

Terminologia

- Differenza tra processi e thread
  - ⇨nei sistemi ordinari, processi distinti non condividono memoria
  - ⇨i thread viceversa, hanno un proprio stack, ma condividono il codice e lo heap
  - ⇨di conseguenza sono molto più rapidi da avviare
  - ⇨possono comunicare in modo più semplice

Terminologia

- Un esempio: applicazioni Java
  - ⇨per avviare vari processi che corrispondono ad applicazioni Java è sufficiente il supporto del sistema operativo
  - ⇨basta eseguire più volte il comando java <classe>; viene avviata una macchina virtuale in ciascun processo
  - ⇨per avviare vari thread in un processo di Java è necessario sfruttare le funzionalità del linguaggio

Thread in Java

- In Java
  - ⇨i thread sono oggetti
  - ⇨si tratta però di oggetti speciali perchè i loro metodi non vengono invocati direttamente dal programmatore
  - ⇨ma dalla macchina virtuale, in modo da assegnare a ciascun thread una pila separata dagli altri

Thread in Java

- Creare un thread: il modo più semplice
  - ⇨estendere la classe java.lang.Thread
  - ⇨sovrascrivere il metodo run() ereditato da Thread (l'impl. standard è vuota)
  - ⇨chiamare il metodo start() per chiedere alla macchina virtuale di creare ed avviare il thread
- Attenzione
  - ⇨NON bisogna chiamare direttamente run()

Thread in Java

>> esempiothread.PrimoEsempio

- Un primo esempio molto semplice
  - ⇨esempiothread.PrimoEsempio
  - ⇨una classe che crea due thread che effettuano stampe sulla console
  - ⇨e li avvia
  - ⇨i due thread vengono eseguiti concorrentemente

```
package esempiothread;
public class PrimoEsempio extends Thread {

    private int id;

    public PrimoEsempio(int id) { this.id = id; }

    public void run() {
        for (int i = 0; i < 20; i++) {
            System.out.println("In esecuzione il thread " + this.id);
            try {
                sleep(1000);
            } catch (InterruptedException e) {}
        }
    }

    public static void main(String[] args) {
        PrimoEsempio thread1 = new PrimoEsempio(1);
        PrimoEsempio thread2 = new PrimoEsempio(2);
        thread1.start();
        thread2.start();
        System.out.println("Main concluso");
    }
}
```

il metodo sleep() mette il thread “a dormire” usato in questo caso per rallentare l’esecuzione lancia InterruptedException

Thread in Java

- Semantica della chiamata a start()
  - ⇨viene predisposta una nuova pila
  - ⇨viene avviata l’esecuzione del metodo run() del thread nella nuova pila
  - ⇨e via via l’esecuzione di tutti gli altri metodi chiamati da run()
  - ⇨l’esecuzione continua finchè tutti i metodi terminano (“morte naturale” del thread)

Thread in Java

- Per vedere i thread in funzione
  - ⇨basta utilizzare la combinazione di tasti “ctrl + break” (ctrl + interruzione)
- Si vede che
  - ⇨la macchina virtuale ha effettivamente creato un thread per ciascun oggetto
  - ⇨ci sono altri thread di sistema in esecuzione
  - ⇨tra questi è rilevante il “Main-thread”: il thread lungo il quale viene eseguito il main

Thread in Java

- Nota
  - ⇨nella stampa dei thread non viene riportato il Main-thread; infatti, il metodo main() termina appena dopo aver avviato i due thread
  - ⇨ma l’applicazione prosegue
- Regola generale
  - ⇨l’esecuzione di un’applicazione Java termina solo quando sono terminati tutti i thread avviati

Thread in Java

- Il metodo sleep()
  - ⇨“mette a dormire” il thread per un certo periodo di tempo, ovvero suggerisce al sistema operativo che il thread non dovrebbe essere eseguito per X millisecondi
  - ⇨il sistema operativo può decidere di adeguarsi o meno
  - ⇨se risveglia il thread prima del tempo si verifica una InterruptedException

Thread in Java

- Nota
  - ⇨dal momento che non è possibile prevedere la strategia di scheduling utilizzata dal S.O., tutti i thread dovrebbero periodicamente eseguire sleep(), anche solo per 1 millis.
  - ⇨per dare spazio ad altri thread
  - ⇨in questo caso i thread vengono detti “educati” (“well behaved”); altrimenti sono detti “egoisti” (“selfish”)

Thread in Java

- Attenzione
  - ⇨questa è la difficoltà fondamentale per i thread
  - ⇨si tratta di un meccanismo complesso
  - ⇨sul quale il programmatore non ha nemmeno il completo controllo
  - ⇨perchè molto dipende dalla strategia di scheduling del sistema operativo

Thread in Java

- Un'alternativa
  - ⇨il metodo yield()
  - ⇨interrompe l'esecuzione del thread per passarla ad un altro metodo
  - ⇨senza però sospendere il thread
  - ⇨in altri termini, il thread resta pronto a riprendere immediatamente se non ci sono altri thread

Thread in Java

- Interrompere un thread
  - in effetti non è possibile farlo
  - tranne che eseguendo il metodo stop() – deprecato e da evitare
- Il metodo interrupt()
  - richiede al thread di interrompersi
  - imposta un flag consultabile con il metodo interrupted(); se il thread è sospeso lancia InterruptedExceptionh
  - il thread può consultare periodicamente interrupted() e, nel caso sia true, decidere se terminare

Thread in Java >> esempiothread.PrimoEsempioRunnable

- Creare un thread: modo alternativo
  - se la classe i cui metodi devono essere eseguiti in thread separati non può estendere Thread, è possibile implementare l'interfaccia Runnable definendo il metodo run()
  - da un oggetto Runnable è possibile poi creare un oggetto di tipo Thread
  - e chiamare start(); l'esecuzione di run() è delegata al Runnable

```
package esempiothread;
public class PrimoEsempioRunnable implements Runnable {

   private int id;
   public PrimoEsempioRunnable(int id) { this.id = id; }

   public void run() {
      for (int i = 0; i < 20; i++) {
         System.out.println("In esecuzione il thread " + this.id);
         try {
            Thread.currentThread().sleep(1000);
         } catch (InterruptedException e) {}
      }
   }

   public static void main(String[] args) {
      PrimoEsempioRunnable runnable0 = new PrimoEsempioRunnable(0);
      PrimoEsempioRunnable runnable1 = new PrimoEsempioRunnable(1);
      Thread thread0 = new Thread(runnable0);
      Thread thread1 = new Thread(runnable1);
      thread0.start();
      thread1.start();
      for (int i = 0; i < 20; i++) {
         System.out.println("In esecuzione il main");
         try {
            Thread.currentThread().sleep(1000);
         } catch (InterruptedException e) {}
      }
   }
}
```

in questo caso non posso chiamare sleep() perchè PrimoEsempioRunnable NON eredita sleep(); devo acquisire un riferimento al thread in cui il metodo viene eseguito e chiamare sleep sul rif. al thread

in questo caso è evidente che il main() viene eseguito lungo un proprio thread

Thread in Java

- Fork (“separazione”)
  - ⇨ogni volta che un thread ne avvia un altro, si dice che effettua un’operazione di “fork” (avvio di una esecuzione concorrente)
  - ⇨corrispondentemente, i thread possono effettuare operazioni di join
- Join (“ricongiunzione”)
  - ⇨operazione eseguita da un thread per attendere l’esecuzione di un altro thread

Thread in Java

>> esempiothread.PrimoEsempioJoin

- In concreto
  - ⇨viene effettuata chiamando nel thread A il metodo join() sul riferimento al thread B con cui A si vuole ricongiungere
- Esempio: PrimoEsempioJoin
  - ⇨il main thread chiama join() sui due thread
  - ⇨viene sospeso e messo in attesa che l'esecuzione dei due thread termini
  - ⇨in quel momento viene riavviato


```
package esempiothread;
public class PrimoEsempioJoin implements Runnable {

    public static void main(String[] args) {
      PrimoEsempioJoin runnable0 = new PrimoEsempioJoin(0);
      PrimoEsempioJoin runnable1 = new PrimoEsempioJoin(1);
      Thread thread0 = new Thread(runnable0);
      Thread thread1 = new Thread(runnable1);
      thread0.start();
      thread1.start();
      try {
         thread0.join();
         thread1.join();
      } catch (InterruptedException e) {}
      System.out.println("Main terminato");
    }
```

il main thread termina solo dopo che sono terminati i due thread avviati

Problemi con i Thread

- Riassumendo
  - ⇨Java consente di eseguire thread diversi nelle applicazioni
  - ⇨i thread sono utilizzati estensivamente dalla macchina virtuale per operazioni di sistema
  - ⇨i thread sono chiaramente collegati alla programmazione asincrona e alla gestione degli eventi
  - ⇨si tratta quindi di una funzionalità importante

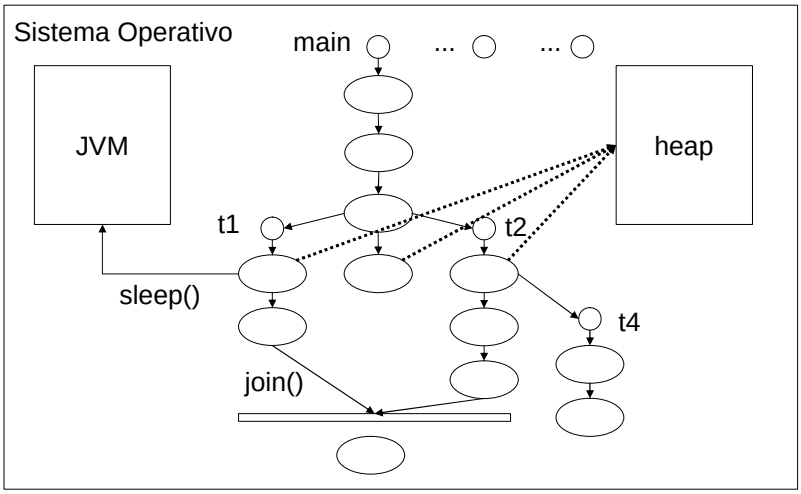
Problemi con i Thread

- Ma...
  - ⇨si tratta anche di uno strumento di programmazione molto complesso
  - ⇨che ha vari effetti collaterali e può produrre gravi errori nelle applicazioni se non viene utilizzato appropriatamente
- Il problema fondamentale
  - ⇨l'elaborazione avviene in modo molto diverso rispetto al modo sequenziale

Problemi con i Thread

- Il modello di programmazione
  - ⇨molto diverso da quello sequenziale
  - ⇨schematizzabile come una sequenza di fork che genera vari fili di esecuzione
  - ⇨che poi possono riunirsi con operazioni join
- La comunicazione
  - ⇨i vari thread condividono l'accesso allo heap (comunicano attraverso gli oggetti)

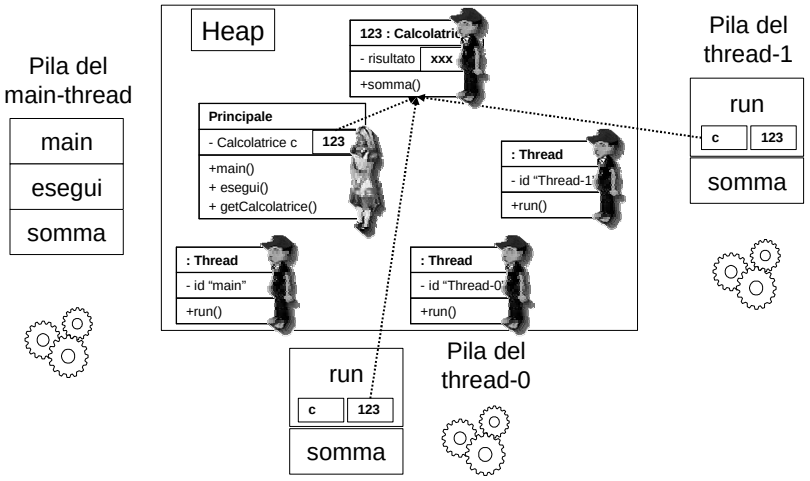
Problemi con i Thread



Problemi con i Thread

- Condividere lo heap
  - ⇨significa che thread diversi possono avere riferimenti allo stesso oggetto
  - ⇨e chiamarne i metodi dell'oggetto ciascuno lungo la propria pila
  - ⇨e quindi modificare concorrentemente lo stato degli stessi oggetti

Problemi con i Thread



Problemi con i Thread

- E' necessario
  - ⇨che i thread si "accordino" per evitare di interferire nell'accesso agli oggetti
  - ⇨ovvero che stabiliscano regole di sincronizzazione

Riassumendo

- Introduzione
- Terminologia
- Thread in Java
- Problemi con i Thread

Termini della Licenza