Corso di Linguaggi di Programmazione

Lezione 22

Alberto Ceselli
alberto.ceselli@unimi.it

Dipartimento di Informatica
Università degli Studi di Milano

21 Maggio 2013

Unità concorrenti in Java

- Le unità concorrenti in Java sono *metodi* chiamati **run**
- Le entità in cui girano i metodi run sono dei *thread* (a differenza dei task di ADA, che sono *processi*)
- Ci sono due modi per definire una classe con un metodo run:
  - Estendere la classe `Thread` (predefinita)
  - Implementare l'interfaccia `Runnable` (predefinita)

Definizione di Threads in Java

Esempio: `java_threads`

Metodi della classe Thread

- `sleep`: il thread rimane in stato waiting per il tempo specificato (in millisecondi) (*statico*)
- `yield`: "preemption" volontaria, il thread passa in stato ready (*statico*)
- `join`: l'esecuzione del thread è sospesa, in attesa del completamento dell'esecuzione di un altro thread

```
public void anyThreadMethod() {
   ...
   Thread myTh = new Thread();
   myTh.start();
   // computazione
   myTh.join(); // myTh.join(2000);
   // computazione
}
```

- `interrupt`: setta un bit, che può essere controllato con isInterrupted() (e solleva l'eccezione `InterruptedException`)
- `setPriority()`.e `getPriority()`
- `stop`, `suspend` e `resume` sono *deprecated*.

Competition synchronization

- Simile a monitors
- Ciascun metodo può essere marcato come **synchronized**.
- Ci può essere **al più un** thread attivo (ready o running).
- Ogni oggetto con metodi `synchronized` conserva una coda di thread in attesa di ottenere l'accesso.

```
class BufferManager {
   private int shared_resource;
   ...
   public synchronized void put(int item) { ... }
   public synchronized int get() { ... }
}
```

Cooperation synchronization

- Gestita con metodi `wait`, `notify` e `notifyAll` della classe `Object`.
- Questi metodi possono essere chiamati solo all'interno di metodi `synchronized` (perchè sfruttano il lock sull'oggetto)
- `wait`: da running a waiting
- `notify`: un thread tra quelli in coda (waiting) sull'oggetto passa da waiting a ready (non-deterministico)
- `notifyAll`: tutti i thread in coda (waiting) sull'oggetto passano da waiting a ready.
- N.B. Utilizzando `notifyAll` è spesso necessario ri-controllare le condizioni per cui il thread si era messo in attesa (attesa **semi-attiva**).

Valutazione

- Semplice, ma efficace.
- Thread (lightweight) rispetto ai Task (heavyweight) di ADA: non adatti ad ambiente distribuito (librerie!)
- Si possono simulare monitor e semafori

Esercitazione: Buffer condiviso da lettori e scrittori

- Realizzare una classe Buffer
- Realizzare una classe Reader
- Estendere Reader, includendo una read multipla
- Implementare un Thread_Writer
- Implementare un Thread_MultipleReader
- Estendere Buffer, rendendolo *thread safe*