Multithreading pratico

lunedì 13 marzo 2023 08:37

Threads = Astrazione del linguaggio
2 approcci in java per crearli:

1. Estensione classe
   Estendendo il thread main come extends, è possibile usare il thread main come anche Threads
   Implementando il run()

```
public class ThreadExample extends Thread {

  public void run() {
    System.out.println("Ciao!");
  }

  public static void main(String arg[]){
    ThreadExample t1=new ThreadExample();
    t1.start();
  }
}
```

   Con start() si prende il thread t e si porta in ready -> potrebbe non partire subito
   Noi dobbiamo chiamare start() e non run(), questo perché chiamando run() noi non creiamo un secondo thread.

1. Implementazione di run() della classe Runnable
   In questo caso noi dobbiamo praticamente implementare il tjread da noi

```
public class RunnableExample implements Runnable{
  public void run() {
    System.out.println("Ciao!");
  }
  public static void main(String arg[]){
    RunnableExample re = new RunnableExample();

    Thread t1 = new Thread(re);

    t1.start();
  }
}
```

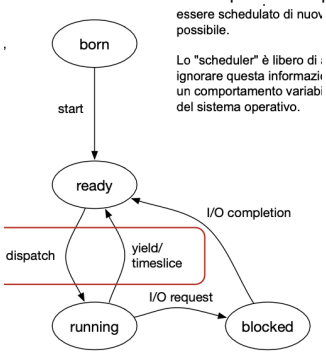
In questo modo `t1` contiene un riferimento a `re`

Quando il metodo `t1.start()` viene chiamato, si esegue il metodo `run()` fornito da `re`

Quando si stampa un thread come stringa, otteniamo: [Nome, Priorità, Gruppo di appartenenza]
La funzione start() Avvia la funzione run()
Quando nel costruttore noi facciamo l'override e richiamiamo il metodo start() dentro,
questo si chiama Pattern.

Nota: noi sappiamo l'ordine delle istruzione, però avviando 2 thread contemporenei non vuol dire che siamo sincroni con le istruzioni; questo è chiamato nondeterminismo
Nota: una volta che un thread è stato avviato non può essere dinuovo eseguito, quindi no 2 volte start()
-> Se accade throwing IllegalThreadStateException

La vita di un thread è come quella di un processo



Un thread passa da ready in running attraverso:

- Yeld : il thread lo chiede di venire tolto
- Timesliced : il tempo è finito

Se vogliamo terminare un thread, si chiama la funzione interrupt()
\-> Mand un flag al thread, e poi il programmatore deve gestirlo con isInterrupt()

Per fare aspettare: Thread.sleep(200)

I thread normalmente sono creati alla base, e poi assegnati col tempo.
Questo viene chiamato Thread pools:

- Noi abbiamo una lista di thread
- Quando abbiamo bisogno di un thread, noi lo prendiamo dalla nostra lista

Vantaggi:

- Rapido siccome la creazione di thread è lenta
- Threads limitati, e questo è un vantaggio siccome possiamo gestirci meglio

Fork = avvia un nuovo thread
Join = aspetta la fine