1 Lezione del 23-10-25

1.1 Gestore di risorse equivalenti

Vediamo un altro pattern tipico della programmazione concorrente la **gestione di risorse equivalenti**.

Assumiamo di avere un insieme finito di **risorse**, gestite da un certo **gestore**. Il gestore ha il compito di offrire due metodi:

- `int richiesta()`: restiuisce l'indice della risorsa che viene attribuita al chiamante;
- `void rilascia()`: rilascia la risorsa assegnata al chiamante.

Potremmo implementare tale sistema come segue:

```
1  class Gestore {
2    boolean[] assigned;
3    int numAssigned;
4
5    public Gestore(int n) {
6      assigned = new boolean[n];
7    }
8
9    public synchronized int richiesta() throws InterruptedException {
10     while(numAssigned == assigned.length) {
11       wait();
12     }
13   }
14
15   public synchronized void rilascia() throws InterruptedException {
16
17   }
18 }
```

finisci

1.2 Metodo `join()`

Il metodo `join()` viene offerto dalla classe *Thread*, e serve ad aspettare che un altro thread finisca.

Viene chiamato su un istanza di oggetto di tipo thread (non corrispondente al thread corrente), e ha il seguente effetto:

- Il thread corrente si blocca finché il thread riferito non termina;
- Se il thread riferito ha già terminato, non ha effetto.

Esistono 3 versioni del `join()`:

- `public final void join() throws InterruptedException`: come quello appena discusso;
- `public final void join(long millis) throws InterruptedException`: come sopra, ma prevede un timeout di `millis` millisecondi;
- `public final void join(long millis, int nanos) throws InterruptedException`: di nuovo come sopra, ma prevede un timeout di `millis` millisecondi e `nanos` nanosecondi;