Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Programmazione Concorrente e Distribuita
Thread: esercizi

Luigi Lavazza
Dipartimento di Scienze Teoriche e Applicate
luigi.lavazza@uninsubria.it

Esercizio 1

- Realizzare un sistema con un produttore e un consumatore
  - Usando un monitor
  - La cella è thread-safe, ma non blocca i consumatori su buffer vuoto o i produttori su buffer pieno
- Ovviamente, si osservano problemi...

Esercizio 1b

- Modificare la soluzione dell'esercizio 1 in modo che il sistema si comporti «bene», cioè
  - Il produttore aspetta quando il buffer è pieno
  - Il consumatore aspetta quando il buffer è vuoto

Esercizio 2

- Si realizzi un sistema con più produttori e consumatori

Esercizio 3

- Realizzare un sistema con un produttore e un consumatore
  - Usando i semafori, non i monitor
  - Tutta la sincronizzazione viene fatta nella coda (da realizzare)

Esercizio 4

- Realizzare un sistema con un produttore e un consumatore
  - Che fanno polling usando BlockingQueue
  - Usare
    - add (che solleva eccezione in caso di fallimento)
    - e poll (he restituisce null in caso di fallimento)

Dalla documentazione

- https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/concurrent/BlockingQueue.html

add

```
boolean add(E e)
```

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning `true` upon success and throwing an `IllegalStateException` if no space is currently available. When using a capacity-restricted queue, it is generally preferable to use `offer`.

Returns:

`true` (as specified by `Collection.add(E)`)

Throws:

`IllegalStateException` - if the element cannot be added at this time due to capacity restrictions

Dalla documentazione

- https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/concurrent/BlockingQueue.html

poll

`E poll()`

Retrieves and removes the head of this queue, or returns `null` if this queue is empty.

Returns:

the head of this queue, or `null` if this queue is empty

Esercizio 5

- Realizzare un sistema con un produttore e un consumatore
  - Con BlockingQueue
    - usando special value (offer e poll)

Dalla documentazione

- https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/concurrent/BlockingQueue.html

offer

```
boolean offer(E e)
```

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning `true` upon success and `false` if no space is currently available. When using a capacity-restricted queue, this method is generally preferable to `add(E)`, which can fail to insert an element only by throwing an exception.

Specified by:

`offer` in interface `Queue<E>`

Parameters:

`e` - the element to add

Returns:

`true` if the element was added to this queue, else `false`

Esercizio 6

- Realizzare un sistema con un produttore e un consumatore
  - Con BlockingQueue
    - con time out

Dalla documentazione

- https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/concurrent/BlockingQueue.html

offer

```
boolean offer(E e, long timeout, TimeUnit unit) throws InterruptedException
```

Inserts the specified element into this queue, waiting up to the specified wait time if necessary for space to become available.

Parameters:

`e` - the element to add

`timeout` - how long to wait before giving up, in units of `unit`

`unit` - a `TimeUnit` determining how to interpret the `timeout` parameter

Returns:

`true` if successful, or `false` if the specified waiting time elapses before space is available

Dalla documentazione

- https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/concurrent/BlockingQueue.html

poll

```
E poll(long timeout, TimeUnit unit) throws InterruptedException
```

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

Parameters:

`timeout` - how long to wait before giving up, in units of `unit`

`unit` - a `TimeUnit` determining how to interpret the `timeout` parameter

Returns:

the head of this queue, or `null` if the specified waiting time elapses before an element is available

Throws:

`InterruptedException` - if interrupted while waiting

Esercizio 7

- Realizzare un sistema con due thread che condividono una risorsa
- I due thread devono accedere alternativamente alla risorsa.
  - Cioè non deve mai succedere che lo stesso thread acceda due volte consecutive alla risorsa.