This course has already ended.
The latest instance of the course can be found at: Concurrent Programming: 2023

« Round 3 - Synchronization primitives Course materials 2 Semaphore -- A mechanism to solve critical section problems and achiev...
CS-E4110 / Round 3 - Synchronization primitives / 1 Blocking monitor -- Low-level concurrency and synchronization

Assignment description | My submissions (1/10)

Blocking monitor -- Low-level concurrency and synchronization

In Scala (Java) synchronization is built in. Every object that has a synchronized method is associated with an intrinsic lock also referred to as monitor lock. This lock plays a vital role when we need to enforce exclusive access to an object's state or establish a happens-before relationship and create thread-safe data structures. Whenever control enters a synchronized method, the thread that called the method acquires the monitor lock for the object whose method has been called. Other threads cannot call a synchronized method on the same object until the monitor lock is released.

One example use of monitor locks is in implementing Monitor Blocking Queue. A Monitor Blocking Queue is a queue that blocks when you try to take from it and the queue is empty, or if you try to put items to it and the queue is already full. A thread trying to take from an empty queue is blocked until some other thread inserts an item into the queue. A thread trying to put an item in a full queue is blocked until some other thread makes space in the queue.

Code

Download the assignment template here

Task

In this exercise, we implement a simple thread-safe Blocking Monitor Queue with `take()` and `put()` methods.

Hint

Look into and use `synchronized, notifyAll() and wait()`. Read more from here.

MonitorBlockingQueues.scala

Choose File No file chosen

Submit

Earned points
25 / 25

Exercise info

Assignment category
Programming exercises

Your submissions
1 / 10

Deadline
Monday, 22 November 2021, 14:00

Late submission deadline
Monday, 29 November 2021, 14:00 (-30%)

Total number of submitters
51

« Round 3 - Synchronization primitives Course materials 2 Semaphore -- A mechanism to solve critical section problems and achiev...