This course has been archived (Saturday, 17 December 2022, 00:00).

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

Assignment description | My submissions (1/10)

Semaphore Blocking Queue -- Concurrent data structures based on low-level concurrency mechanisms.

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

Semaphore Blocking Queue

Another way of implementing a Blocking Queue is using semaphores. As we have encountered in the previous exercises a 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 semaphore, on the other hand, is an important abstract data type used to control access to a common resource required by multiple execution units (threads) in a concurrent system.

Code

Download the assignment template here

Task

In this exercise, we will implement a Blocking Queue based on a Semaphore similar to what we have implemented in the last exercise. You are given a Semaphore which has `acquire()` and `release()` methods. You will need to base your Blocking Queue implementation on this Semaphore. The full interface definition of `Semaphore` is as follows:

```
class Semaphore(private val capacity: Int) extends Monitor {
    var permits = capacity
    def acquire(): Unit
    def release(): Unit
    def availablePermits(): Int = permits
}
```

Hint

Take a look at the hints in the previous exercises related to Semaphore and Monitor Blocking Queue.

SemaphoreBlockingQueue.scala

Choose File No file chosen

Submit