<



A This course has been archived (Saturday, 17 December 2022, 00:00).

Course



CS-E4110

Course materials

Your points

MyCourses

X Zulip Chat

This course has already ended.

The latest instance of the course can be found at: Concurrent Programming: 2023

« 1 Blocking monitor -- Low-level concurrency and synchronization 3 Semaphore Blocking Queue -- Concurrent data structures based on low-... Course materials CS-E4110 / Round 3 - Synchronization primitives / 2 Semaphore -- A mechanism to solve critical section problems and achieve synchronization.

Assignment description

My submissions (2/10) ▼

Semaphore -- A mechanism to solve critical section problems and achieve synchronization.

Semaphore

Semaphore is another useful tool to prevent race conditions and solve other such critical section problems. A semaphore is an important abstract data type used to control access to a common resource required by multiple execution units (threads) in a concurrent system. Simply put, a semaphore is a variable used to record how many units of a particular shared resource are available. Of course, for such a variable it is necessary to make sure the record is safely adjusted to avoid any race conditions.

Code

Download the assignment template here

Task

In this exercise, we implement a simple semaphore with acquire() and release() methods. We will make use of Java Monitors to implement our semaphore.

Hint

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

Semaphore.scala

Choose File No file chosen

Submit

« 1 Blocking monitor -- Low-level concurrency and synchronization

Course materials

3 Semaphore Blocking Queue -- Concurrent data structures based on low-...

Earned points

25 / 25

Exercise info

Assignment category

Programming exercises

Your submissions

2 / 10

Deadline

Monday, 22 November 2021, 14:00

Late submission deadline

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

Total number of submitters

53