Practice

Due Dec 21, 2020 at 1pm **Points** 60 **Questions** 1
Available Dec 21, 2020 at 10:45am - Dec 21, 2020 at 1:15pm about 3 hours
Time Limit 130 Minutes

This quiz was locked Dec 21, 2020 at 1:15pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	106 minutes	32 out of 60

Score for this quiz: **32** out of 60
Submitted Dec 21, 2020 at 12:32pm
This attempt took 106 minutes.

Question 1

Not yet graded / 60 pts

Eötvös Loránd University – Department of Programming Languages and Compilers

Concurrent Programming Lab – Exam 1

Exercise (60 points)

Please keep in mind that the answers will be checked through a plagiarism checker. In case of copied solutions, you will receive one grade, divided by the number of people involved.

To be evaluated, your solution MUST COMPILE correctly (no compilation errors).

The exam tepmlate is (here)

Consider the following situation. A thread pool needs to be implemented for a small real-time system with particular scheduling features and a time-based execution policy. The thread pool has a fixed size and takes tasks in a FIFO order from its queue. It should execute each task it has been assigned in a time that is harmonic to a given time tick value. Time here is expressed in milliseconds.

You are asked to model this situation by defining the appropriate classes, in particular:

- **(35 points)** A `HarmonicThreadPool` class, which has the following properties:
    - A private final `int` attribute called `size`.
    - Two private final `long` attributes called `base_tick` and `max_time`.
    - A private `long` attribute called `time`.
    - Its constructor takes an `int n` that defines the size of the thread pool and two `long` values to initialize `base_tick` and `max_time`. The attribute `time` is initialized to the double of `base_tick`. This constructor **creates and starts** the thread pool (as standard).
    - A public `void` method `update(long nTime)`, which takes a `long` value and uses it to update the `time` attribute by the following policy: if the received value is greater than `max_time`, then the **error** `“Max time exceeded! Shutting down the thread pool”` is printed out and the thread pool is shut down. Else, if the duration of a task is greater than the current `time` property of the thread pool, it is used to compute a new value for `time` that is harmonic to the `base_tick` value (should be the multiple of `base_tick` that is immediately greater than `nTime`). On top of that, to keep the thread pool balanced, after every `base_tick` calls of `update(...)`, the `time` attribute should be reset to the double of `base_tick`.
    - A public `void shutdown()` method which should safely shut down the thread pool (**safety as defined during the lectures**). The statement `“<thread_name> has stopped”` is printed for every terminated thread, right after its termination. Of course, **an actual shutdown should happen only once**, no matter how many times the method is called. The statement `“Thread pool <i> has shut down”` is printed after all threads have successfully terminated.
- **(15 points)** A private inner `Worker` class that extends `Thread` and has the following properties:
    - After executing a task, `Worker` threads **update** the thread pool through the duration of each task.
    - Each `Worker` thread then sleeps until the end of the time slot defined by `time` (an amount of time such that `<task_duration> + <sleeping_time> = time`).
    - The threads should “pretty-print” themselves, the same way threads belonging to thread pools usually do (`“pool-<i>-thread-<j>”`).
- **(10 points)** A `Task` class that implements `Runnable`. The behaviour for tasks that are instances of this class is simply sleeping for a random amount of time such that `(sleeping_time) > 0`.

IMPORTANT:

The basic structures needed for the implementation of a thread pool are omitted from this specification and **defined in the exam template file**.

It is recommended to use `System.currentTimeMillis()` to perform the time calculations.

Method signatures **must be kept as in the specification**, but you are free to define additional helper structures or methods as you wish.

It is not specified which methods should be synchronized and which not. You should implement synchronization **as required to make the whole program thread safe**.

exam.rar (https://canvas.elte.hu/files/857941/download)

Quiz Score: **32** out of 60

This quiz score has been manually adjusted by +32.0 points.