

i Information

Written Exam in IN3030/IN4330

2022 Spring Semester

Duration: 4 hours

Date: June 7th, 2022

General information:

- IN4330 students are given the same questions as IN3030 students but are expected to give better and more in-depth answers.
- Your answer should reflect your own independent work and should be a result of your own learning and work effort.
- Concerning Answers that involve Java Programming: the prime concern is what the program does; it is a secondary concern that the program is entirely syntactically correct, so good pseudo-code where boring or less relevant Java details are missing is acceptable: Spend your time on getting the algorithms right rather than the syntax.

Support material:

All written materials.

The answer must be written with an academic standard.

Skoleeksamen IN3030/IN4330

Vår 2022

Varighet: 4 timer

Dato: 7. juni 2022

General information:

- IN4330 students are given the same questions as IN3030 students but are expected to give better and more in-depth answers.
- Your answer should reflect your own independent work and should be a result of your own learning and work effort.

Support material:

All written materials.

The answer must be written with an academic standard.

1 1.1 The Cache-friendliness of Bubblesort

```
// Bubble sort in Java
import java.util.Arrays;
class Main {
    // perform the bubble sort
    static void bubbleSort(int array[]) {
        int size = array.length;
```


Words: 0

Maks poeng: 13

2 1.2 Bubblesort Improvements

Suggest improvements to Bubblesort so that it becomes more cache-friendly. Your improvements must respect the basic principle of Bubblesort including that it works by comparing two adjacent elements and exchanging them, if they are out of order, thereafter moving on one element further in the array and repeating. However, the individual comparisons might be reordered to achieve better cache performance.

Write your suggested improvements here:

Maks poeng: 15

3 1.3 Programming a Cache-Friendly Bubblesort

Pick ONE of your suggested improvements of the `bubbleSort` method from the previous question, OR describe a new one. Program the improved method in Java. Include your description of the chosen improvement in the program as comments in the Java code.

Write the improved version of the method here

Maks poeng: 15

4 2.1 Parallel BubbleSort using a new type of Synchronization

We can define a new synchronization method, let us call it **NoPass**, that works as follows: Each thread involved in the synchronization is given a unique *id*, which is merely a non-negative integer assigned consecutively from zero and up, so if there are N threads involved, the *ids* will be from zero to N-1. When a thread calls the method *NoPass(id)*, the thread gives its *id*, and first *NoPass* checks whether or not there is a thread with an id of *id+1* that is blocked and has called *NoPass* FEWER times than the thread with id of *id*. If so, it unblocks that thread. Second, the thread checks whether or not there is any thread with a lower *id* that has called *NoPass* a FEWER OR THE SAME number of times than the thread with id of *id*. If so, the thread with id of *id* blocks itself.

NoPass thus ensures that no thread with a higher id ever catches up with any thread that has a lower id.

Assume that we have a class **NoPassC** that implements this type of synchronization. The class has one public method, *NoPass(id)*, as described above.

Describe an efficient, parallel version of BubbleSort that uses the **NoPass** synchronization mechanism described previously.

Note: you do *not* need to incorporate the cache-friendly suggestions from question 1 into your parallel version - here, merely focus on the parallelization and making that efficient.

Describe your parallel version using NoPass here

Maks poeng: 15

5 2.2 Implementation of Parallel BubbleSort

Write the Java code for the parallel version of BubbleSort that you described in the previous

question. You may freely copy the presented earlier and modify it instead of starting from scratch. Include any descriptions in comments in the code.

Write the Java code for the method here

```
1 |
```

Maks poeng: 20

6 3.1 Speed of light in vacuum

What is the speed of light in vacuum? Choose the most precise answer.

Choose ONE alternative:

- 300 000 m/s
- 299 792 458 m/s
- 299 458 792 m/s
- 299 458 792 km/s
- 300 000 000 km/s
- 300 000 km/s

Maks poeng: 2

7 4.1 IVar Synchronization Class

An **IVar** is a concurrency controlled object that can be written to once, but read many times. A feature is that you can ask to read the variable before you have written to it.

The **IVar** have the following two methods.

put (value) stores an integer value in the internal storage.

get returns the value that is stored in the IVar.

When the **IVar** receives the first `put` it saves the value to an internal variable and returns `true`. The following `put` messages will not update the variable, but return `false`.

When the **IVar** receives a `get` it returns the stored variable. All `get`'s should be blocking, until the **IVar** has received the first `put`. Thus, a process sending a `get` before another process send a `put` should block and wait until the **IVar** receives the first `put` and responds to all waiting clients.

Implement a class **IVar** in Java using semaphores as the only synchronization. The class must implement the above methods and the value can be any kind of type specified when the object is created.

Write your IVar Java Class here:


1	
---	--

Maks poeng: 7

8 4.2 IVar Synchronization Explanation

Give a brief explanation of your class from 4.1. Argue how and why the class ensures proper and correct concurrent access to an **IVar** object.

Fill in your brief explanation here

Format - | | ↺ | | |  |

Σ | ✖

Words: 0

Maks poeng: 8

9 4.3 Test Program for IVar

Write a Java test program that demonstrates a representative testing of your implemented **IVar** from question 4.1. Ensure that the output of the test explains the steps of the execution.

Explain the test that you chose and why it gives good test coverage.

Write the Java test program below. Include, as comments, your explanations.

Write your Java test program here:

1 |



Maks poeng: 5