IN3030/IN4330 Spring 2020 Written Exam

Eric Jul

Duration: June 3rd 09:00 – June 10th, 2020 ***09:00***

(Note: delivery time is **09:00**)

It is important that you read this page carefully before you start.

General information:

- Your submission must be uploaded as a PDF-file. How to make a PDF-file.
- Remember that your submission needs to be anonymous, do not write your name in your submission.
- All examination support materials are permitted. You need to gather information from available sources, assess the information quality, and put it together in a submission based on your own processing of the content. The submission must reflect your individual level of knowledge.
- For assignments where it is relevant to use sources and citations, it is important that you do this properly so that you are not suspected of cheating. Read more about sources and citations.
- You have to read UiO's upload assignment student guide
- You have to read IFI's rules about cheating on exams.

Digital hand drawing:

- If your submission includes digital hand drawings, you are free to use your preferred tools (scanning, cellphone-camera, *etc.)* as long as everything is readable and delivered as one PDF.
- Check out MN's/UiO's recommended solutions for digital hand drawings spring 2020.

NB!
You cannot apply for a postponement of the exam beyond the 7 days the exam is held. If you submit a self-notification about illness, you will be able to take the continuation exam in the courses that offer it, or take the exam the next time the course is held (applies to IN1150, IN1000 and ENT1000).
See: https://www.mn.uio.no/om/hms/koronavirus/eksamen-2020.html (Paragraph 9, 10 and 11)

Contact:

- User support for exams in the spring of 2020.

Good luck!

Question 1: Java and Synchronization

Question 1.1: Java Threads Startup (1 point)

In a Java program, there is one main thread that is automatically started when a program starts: it executes the `main` method of the program. This method can start other threads. Can those thread also start other threads? Provide a short explanation.

Question 1.2: Java Threads Execution on Multicore (3 points)

Explain briefly how 100 Java threads can *appear* to execute concurrently as if the CPU had 128 cores – despite it having only 4 cores.

Question 1.3 Java Synchronized (14 points)

There are alternatives to using the Java keyword `synchronized` including many very specialized ways of synchronization. Pick two of your favorite *alternative* to `synchronized` and explain why you prefer them and how they can replace `synchronized`. Your preferences can be based on *any* type of argument: be it performance, usability, convenience, or a specialized use. The most important part of your answer will be your presentation of the comparative advantages and disadvantages in connection with any specific application of your choice.

Question 2: Join using Semaphores

Question 2.1 Join Replacement (16 points)

You are to achieve the effect of Java's `join`, but instead of using `join`, you should use semaphores as in the Java class `Semaphore`.
The idea is to modify the `JoinP` Java program given below to NOT use `join`, but instead use semaphores. Your task is now to write a version of `JoinP` where you have the `join` with some Java code that makes the program work like the original `JoinP` program without using join but instead using semaphores.
Provide the resulting program along with a short explanation of your solution.
Here is the `JoinP` program:

```
import java.util.concurrent.*;
class JoinP {

   public static void main(String[] args) {
      int numberofthreads = 10;
      Thread[] t = new Thread[numberofthreads];

      for (int j = 0; j < numberofthreads; j++) {
         (t[j] = new Thread( new ExThread() )).start();
      }

      try {
         for (int k = 0; k < numberofthreads; k++) t[k].join();
      } catch (Exception e) { return; }

   }

   static class ExThread implements Runnable {

      public void run() {
         try {
            TimeUnit.SECONDS.sleep(10);
         } catch (Exception e) { return;};
      }

   }

}
```

Question 2.2 Test Case (12 points)

You are to write a Java program that demonstrates a test case for the program that you wrote in 2.1.

Explain the test that you chose and why you think it shows that your program from 2.1 works – at least for your chosen test case (it does not – at all – have to be comprehensive – just show a typical case). Each thread could, for illustration, print what it does at each step – be sure to include an id of the thread doing the printing.

Hint: You can "schedule" when threads actively try to do stuff by delaying them using, *e.g.*, `TimeUnit.SECONDS.sleep(10);`

Provide the program and its output and any comments that you might have.

Question 3: Double-bubblesort

Bubblesort is a sorting algorithm that sorts, *e.g.*, an integer array A, by repeatedly going thru the array from one end to the other comparing neighboring elements to each other – and swapping them when they are not in order. Double-bubblesort is a variant of bubblesort that works by comparing not two elements at a time but rather *three* elements at a time and swapping them so that they are in order.

Question 3.1 Sequential Double-bubblesort (5 points)

Write a sequential version of double-bubblesort in Java.
Provide the Java program as your answer.

Question 3.2 Testing Double-bubblesort (3 points)

Write a simple Java test program showing that your program from 3.1 can sort an array of 10 elements containing the numbers from 1 to 10 *in reverse order*. Print suitable test output from the program.
Provide the program and its output.

Question 3.3 Parallelizing Double-bubblesort (40 points)

How can your program from 3.1 be parallelized?
Describe the design of a solution that MUST be loyal to the algorithm, *i.e.*, splitting the array into *k* parts that are double-bubblesorted individually then merge sorted, is not loyal as much of the speedup is gained by using merging, which is much more efficient for large arrays than any bubblesort. Furthermore, you should explain your considerations of the caching effects of your solution.

Hint: spend time on describing the parallelization as this is central to the course.

Question 4.1 The Speed of Light in Vacuum (1 point)

What is the exact speed of light in vacuum?
Give your answer in m/s and make sure it is exact.

Question 4.2 Latency (5 points)

In 1988, the latency for a so-called ping request, that is the time to send a packet over the internet from one point, A, to another point B, and a reply making its way from B to A, was about 200 ms for a ping request from Scandinavia to the US West Coast – approximately 10,000 km. In 2020, the ping time is about 180 ms.
Explain why the time is *almost* the same after 32 years---despite great improvements in transatlantic bandwidth.

END OF EXAM