i

Information

IN3030 Spring 2019 Written Exam

Date and time: June 7th, 2019 at 09:00.

Duration: 4 hours

Material allowed: All written material is allowed. No electronic aid is allowed. The lecture slides are attached as a PDF.

Point for Multiple Choice with more than one alternative: Points are given for correct answers; points are deducted for wrong answers. If the sum for a single question is below zero, zero points are given for the question.

1

Spørgsmål 1.1: Java Program Avslutning

Question 1.1: Java Threads Termination

In a Java program that contains more than one thread, when does the program normally (ignoring exceptions and other error conditions) finish execution?

Select an alternative:

- When all the threads created in the main() method have terminated AND any thread created by any other thread in the program has terminated AND the main() method also has terminated.
- When all the threads created in the main() method have terminated.
- When all the threads created in the main() method have terminated AND the main() method also has terminated.
- When the main() method terminates.

Maximum marks: 5

2 Spørsmål 1.2: Kjøring av tråder i Java på multikjerne maskiner

Question 1.2: Java Threads Execution on Multicore

Which of the following statements are true for Java Threads that are created in the same program? (More than one statement might be true, so select all alternatives that are true.)

Select one or more alternatives:

- Java threads execute in parallel: Every core runs a thread until it either terminates or has used up a time slice, whereafter another thread (possibly the same thread) is run on the core - and so on.
- If there are more threads in a program than available cores, the program cannot be run.
- Java threads appear to execute in parallel independently of one another even on single-core machines because the threads are time multiplexed.
- Each core runs one Java thread. When that thread terminates, another Java thread is selected and run until it terminates - and so on.
- All Java threads execute at the same speed because each thread is assigned to its own core .
- Java threads can execute in parallel independently of one another on multi-core machines.
- Java threads always execute one at a time.

Maximum marks: 6

3 Spørsmål 1.3: Java Synchronized

Question 1.3: Java Synchronized

Can we ensure that only one thread at a time is executing a method by prefixing the method with the **synchronized** keyword? (Select all alternatives that are true.):

Select one or more alternatives:

- Yes, furhtermore, if another thread tries to call another syncrhonized method in the same object, the Java system ensures that only one thread is executing within any of the synchronized methods in the object.
- Yes, furthermore, if another thread tries to call a different synchronized method in the same object, the threads are allowed to execute in parallel.
- Yes, if more than one thread calls the method, the Java system ensures that only one thread is executing within the object.
- No, inside the method, we must add a synchronization of some kind, e.g., using a semaphore.

Maximum marks: 4

4

Spørsmål 1.4: Ytelsen til tråder i Java

Question 1.4: Performance of Java Threads
Hvilket av de følgende svarene er nærmest virkeligheten?
Select an alternative:

- Metoder i Java med synchronized nøkkelordet er like effektive som metoder uten dette nøkkelordet.
- Metoder i Java med synchronized nøkkelordet er billigere å bruke enn for eksempel CyclicBarrier.
- Metoder i Java med nøkkelordet synchronized er like effektive som om man bruker en CyclicBarrier.
- CyclicBarrier er billigere å bruke enn synchronized nøkkelordet.

Maximum marks: 5

i

Spørsmål 2: Join ved hjelp av Cyclic Barrier

Question 2: Join using Cyclic Barriers
In this question, you should consider the program JoinP.java attached as a PDF-document. In the program, the main() method starts a number of thread that each (pretend) to do some work for 10 seconds (in this simplified version, the threads merely wait 10 seconds thus simulating 10 seconds of work). After starting the threads, the main() method awaits the completion of all the threads that it started by using join.

5

Spørsmål 2.1: Cyclic Barrier erstatter Join

Question 2.1: Cyclic Barrier Replacement for Join
You are now to achieve the effect of join, but instead of using join, you should use a CyclicBarrier. The idea is to modify the JoinP.java program to NOT use join, but instead use a single CyclicBarrier. Attached is a modified version of the JoinP.java program, called CyclicBarrierJoinP-XXX, that achieves this, albeit there are parts missing - partially indicated by XXXX in the program. Your task is now to write a version of CyclicBarrierJoinP-XXX where you have replaced all the XXXX with some Java code that makes the program work like the original JoinP.java program. You are welcome to copy the CyclicBarrierJoinP-XXX program and then modify it directly.

Either copy from the PDF - or from the text below:

```
import java.util.concurrent.*;
class CyclicBarrierJoinP {
  static CyclicBarrier cb;
  public static void done() {
    XXXX
  }
  public static void main(String[] args) {
    int numberofthreads = 10;
    Thread[] t = new Thread[numberofthreads];
    cb = XXXX;
    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; }
    XXXX;
  }
  static class ExThread implements Runnable {
    public void run() {
```

```
            try {
              TimeUnit.SECONDS.sleep(10);
            } catch (Exception e) { return;};
            done();
          }
        }
      }
```

Fill in your answer here

```
1
```

Maximum marks: 15

Spørsmål 3 Finn Primtalltvillinger (Twin Primes)

Question 3: Finding Twin Prime Pair

This question is based on Erathostenes Sieve that finds prime numbers up to a given number *N*. You should already be quite familiar with the sieve as you have implemented it in Oblig 3 (attached as PDF).

This question is about finding so-called **Twin Primes**. A **twin prime** is a prime number that is either 2 less or 2 more than another prime number—for example, either member of the pair (41, 43) is a twin prime and the two together are called a **twin prime pair**.

You are to write a Java program that given a positive integer, *N,* generates a list of all twin prime pairs where each of the twin primes is less than or equal to *N*. The list should be sorted so that the pairs come in ascending order. The first twin prime pair is (3, 5), the next is (5, 7), and the third is (11, 13).

You need not write the entire program as you can assume that you already have the code that you wrote for Oblig 3.

6

Spørsmål 3.1: Sekvensielt program for å finne Par av Primtalltvillinger (Twin Prime Pairs)

Question 3.1: Sequential Program for finding Twin Prime Pairs

Write a sequential Java program that finds the list of Twin Pair Primes specified in the introduction to Question 3. The program takes an integer parameter, N, which is the largest twin prime to be found.

Fill in your answer here

```
1
```

Maximum marks: 20

7

Spørsmål 3.2: Parallelt program for å finne Par av Primtalltvillinger (Twin Prime Pairs)

Question 3.2: Parallel Program to find Twin Prime Pairs

Write a parallel Java program that finds the list of Prime Deserts specified in the introduction to Question 3. Strive to achieve a speedup over the sequential version. The program takes an integer parameter, N, which is the largest twin prime to be found.

Fill in your answer here

```
1
```

Maximum marks: 40

i

Spørsmål 4: Innstikksortering

Question 4: Insertion Sort

Insertion Sort is a sorting algorithm that sorts, e.g., an integer array A, by dividing an array of elements to be sorted into two parts: a sorted part and an unsorted part. Initially, the sorted part is merely composed of the first element in the array and the remaining part of the array is the unsorted part. The algorithm sorts by repeatedly taking the first element of the unsorted part and then inserting that element into its place in the sorted part - making room for it by shifting the elements larger than or equal to the chosen element.

8

Spørsmål 4.1: Parallellisering av Innstikksortering

Question 4.1: Parallelizing Insertion Sort

How can Insertionsort be parallelized? Describe the design of a solution.

Fill in your answer here

Format

Words: 0

Maximum marks: 20

9

Spørsmål 4.2: Parallell Innstikksortering

Question 4.2: Parallel Insertion Sort

Write a Java program implementing your design of a parallel version of Insertionsort from Question 4.1. Strive to have a speedup over the sequential version. Put any added explanation that you have as comments in the code.

Fill in your answer here

```
1
```

Maximum marks: 40

10

Spørsmål 5.1: Lysets hastighet

Question 5.1: The Speed of Light

The speed of light in vacuum is:

Select one alternative:

- Approximately 300,000,000 km/s
- Approximately 300,000 m/s
- Approximately 300,000 km/s
- Approximately 300,000 km/h

Maximum marks: 2

11

Spørsmål 5.2: Ett nanosekund

Question 5.2: One nanosecond
How far does light travel in vacuum in 1 nanosecond?
Select one alternative:

- 300 millimeters
- 300 meters
- 30 millimeters
- 300 km
- 300 centimeters

Maximum marks: 2

12

Spørsmål 5.3: Relasjonen mellom Caching og Lysets hastighet

Question 5.3: Relationship between Caching and the Speed of Light
Explain what relation there is between the speed of light, the speed of electrical signals in copper and light signals in fiber optics cables, and the use of multi-level caching in modern CPU architectures:
Fill in your answer here

Format

Words: 0

Maximum marks: 15

Question 5
Attached

```
import java.util.concurrent.*;

class CyclicBarrierJoinP {

   static CyclicBarrier cb;

   public static void done() {
      XXXX
   }

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

      Thread[] t = new Thread[numberofthreads];
      cb = XXXX;

      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; }

      XXXX;
   }

   static class ExThread implements Runnable {
      public void run() {
         try {
            TimeUnit.SECONDS.sleep(10);
         } catch (Exception e) { return;};
         done();
      }
   }

}
```

Question 8
Attached

```
public static void insertsort (int a[], int left, int right) {
   int i,k,t;
   for (k = left+1 ; k <= right; k++) {
     t = a[k] ;
     i = k;
     while ( a[i-1] > t ) {
        a[i] = a[i-1];
        if (--i == left) break;
     }
     a[i] = t;
   } // end k
} // end insertsort
```

Question 9
Attached

```
public static void insertsort (int a[], int left, int right) {
   int i,k,t;
   for (k = left+1 ; k <= right; k++) {
     t = a[k] ;
     i = k;
     while ( a[i-1] > t ) {
        a[i] = a[i-1];
        if (--i == left) break;
     }
     a[i] = t;
   } // end k
} // end insertsort
```