Threads (18 points)

5. Die SOLA-Stafette findet jedes Jahr in und um Zürich statt. Jedes Team besteht aus 14 "Runners" (nummeriert von 0 bis 13). Ein Runner darf erst starten, nachdem alle vorherigen Runners seines/ihres Teams den Lauf beendet haben. Die einzige Ausnahme ist der erste Runner, welcher beim Start der Stafette startet. In dieser Aufgabe modellieren wir jeden Runner als einen separaten Thread mit einer *id*.

   SOLA-Stafette is an annual running race relay in and around Zurich. Each team consists of 14 runners (numbered from 0 to 13) where each runner can start after the previous runner from his/her team finished. The only exception is the first runner who starts at the beginning of the race. In this task we model each runner as a separate thread with an associated id.

   (a) Betrachten Sie die folgende Implementierung solcher Threads, wobei das Objekt `y` unter allen Threads geteilt wird.

   Consider the following implementation of such threads shown below where the object `y` is shared amongst all the threads.

```
1  public class RunnerThread extends Thread {
2
3      private AtomicInteger y;
4      private int id;
5
6      public RunnerThread(int id, AtomicInteger y) {
7        this.id = id;
8        this.y = y;
9      }
10
11     public void run(){
12       while (y.get() != this.id) {
13         // warten bis ich an der Reihe bin / wait until it is my turn
14       }
15
16       // laufen / do running
17       y.incrementAndGet();
18     }
19 }
```

   i. Welchen Wert muss `y` ganz am Anfang haben, damit das Programm oben korrekt funktioniert? (1)

   What is the initial value of `y` required for the above program to function correctly?

   Solution: $y = 0$

(b) Wir sind an einer alternativen Implementierung vom `RunnerThread` aus Aufgabe a) interessiert, wo die korrekte Ausführungsreihenfolge mittels den Methoden wait/notify/notifyAll sichergestellt werden soll.

Consider an alternative implementation of the `RunnerThread` from task a) where the correct order of thread execution is ensured using the methods wait/notify/notifyAll.

i. Vervollständigen Sie die Implementierung im Skelett unten, wobei das Objekt `x` unter allen Threads geteilt wird. Sie können exception handling ignorieren.

Complete the implementation in the skeleton below where the object `x` is shared among all the threads. In your solution you can ignore exception handling. (9)

```
1  public class WaitNotifyRunnerThread extends Thread {
2
3          private ..................... x;
4          private int id;
5
6          public WaitNotifyRunnerThread(int id, ..................... x) {
7                  this.id = id;
8                  this.x = x;
9          }
10
11         public ..................... void run(){
12                 ..................................................
13                 ..................................................
14                 ..................................................
15                 ..................................................
16                 ..................................................
17                 ..................................................
18                 ..................................................
19                 ..................................................
20                 ..................................................
21                 ..................................................
22                 ..................................................
23                 ..................................................
24         }
25 }
```

Solution:

```
1  public class WaitNotifyRunnerThread extends Thread {
2
3      private AtomicInteger x;
4      private int id;
5
6      public WaitNotifyRunnerThread(int id, AtomicInteger x) {
7        this.id = id;
8        this.x = x;
```

```
9       }
10
11      public void run(){
12        synchronized (x) {
13          while (x != this.id) {
14            x.wait();
15          }
16
17          x.incrementAndGet();
18          x.notifyAll();
19        }
20      }
21  }
```

ii. Muss die `run` Methode als `synchronized` deklariert werden, damit Ihre Lösung korrekt funktioniert? Erklären Sie kurz.

Is it necessary for the `run` method to be declared as `synchronized` in order for your solution to work correctly? Briefly explain why. (3)

Solution: No. Synchronizing the thread object make no difference as the run method is executed on separate instances of the thread object.

iii. Welche Implementierung ist effizienter? a) `RunnerThread`, b) `WaitNotifyRunnerThread`, c) keine der Beiden ist signifikant effizienter. Erklären Sie warum.

Which implementation is more efficient? a) `RunnerThread`, b) `WaitNotifyRunnerThread`, c) neither is significantly more efficient. Explain why. (4)

Solution: b) as we avoid the overhead of threads spinning on the while loop. Using wait/notify the threads are woken up only when each thread finishes.