Prof. Dr. Th. Letschert

THM TECHNISCHE HOCHSCHULE MITTELHESSEN | MNI Mathematik, Naturwissenschaften und Informatik

NVP – Nebenläufige und Verteilte Programme

Aufgabenblatt 1

Aufgabe 1

1. Was versteht man unter einem nebenläufigen Programm?
2. Warum werden nebenläufige Programme geschrieben? Gibt es Probleme, die nur mit nebenläufigen Programmen gelöst werden können, oder bieten nebenläufige Programme spezielle Vorteile (schneller, weniger Fehler, ...)?
3. Welche Klasse von Problemen werden von nebenläufigen Programmen besonders gut (einfach / elegant / übersichtlich) behandelt?
4. Wodurch unterscheiden sich nebenläufige von parallelen Programmen?
5. Kann ein Programm, das aus mehreren Threads besteht, auf einer Maschine mit nur einer CPU ausgeführt werden?
6. Hat die JVM eine oder mehrere CPUs?
7. Was ist der Unterschied zwischen Threads und Prozessen?
8. Wer ruft die `run`–Methode eines Threads *normalerweise* auf? Kann man sie auch – beispielsweise von einer `main`–Methode aus – direkt aus eigenem Code heraus aufrufen?
9. Kann ein Programm mit *einer* Thread–Klasse *zwei* Threads starten? Wenn nein: warum nicht? Wenn ja: geben Sie ein einfaches Beispiel an!
10. Kann ein Thread–Objekt mehrmals gestartet werden? Wenn nein: warum nicht (Was passiert warum, wenn man es versucht), wenn ja: wie?
11. Erläutern Sie allgemein in welcher Beziehung Threads, Thread-Klassen und Instanzen einer Thread-Klasse stehen.
12. Wie viele *Stacks* und wie viele *Heaps* und wie viele *JVMs* werden benötigt um eine Java–Programm auszuführen, in dem 3 Threads gestartet werden.

Aufgabe 2

Übersetzen Sie folgende Beispiele von Java nach Scala:

1. Einfache Thread-Erzeugung:

```
package blatt_01.aufgabe_02;

class HelloThread extends Thread {
   public void run() {
      System.out.println("Hello from a thread 1!");
   }
}
```

```
class HelloRunnable implements Runnable {
      public void run() {
      System.out.println("Hello from a thread 2!");
   }
}

public class Beispiel_1 {
   public static void main(String args[]) {
      new HelloThread().start();
      new Thread(new HelloRunnable()).start();
   }
}
```

2. In Java-8 sieht das etwas eleganter aus:

```
package blatt_01.aufgabe_02;

public class Aufgabe_01 {

  public static void main(String[] args) {
    new Thread ( () -> System.out.println("Hello from a thread 3!") ) . start();
  }

}
```

Aufgabe 3

Schreiben sie eine Methode zur Faktorisierung, also zur Berechnung der Primafaktoren einer positiven ganzen Zahl. Die Faktorisierung soll unterbrechbar sein. Ihre Funktion soll ein Argument vom Typ Long annehmen und ein Ergebnis einen Wert vom einem Typ liefern, in dem abgebrochene und nicht abgebrochene Berechnungen unterschieden werden. Das Ergebnis einer abgeschlossen Faktorisierung sollen die Primfaktoren als eine Liste von Long-Werten sein.

```
 case object CanceledException extends Exception
 /**
   * Interruptable thread save computation of the prime factors of a natural number.
   * @param n the number to be factorized
   * @return the List of prime factors of n or reason of failure
   * @throws IllegalArgumentException if n <= 2
   * @interrupt retains interrupted flag
   */
 def factors(n: Long): Try[List[Long]] = ...
```