SoSe 2017

Nichtsequentielle und verteilte Programmierung

1. Übungsblatt

Prof. Dr. Margarita Esponda

Ziel: Auseinandersetzung mit grundlegenden Begriffen der Nichtsequentiellen Programmierung.

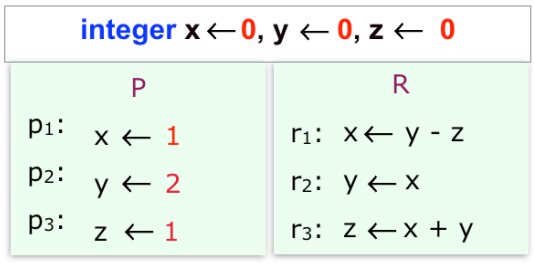
1. **Aufgabe** (2 P.)
   a) Was ist Datenparallelität?
   b) Was ist Funktionsparallelität?

2. **Aufgabe** (2 P.)
   Was ist der Unterschied zwischen Prozessen und Threads (nach der Vorlesungsdefinition)?

3. **Aufgabe** (4 P.)
   a) Wann ist ein Algorithmus determiniert?
   b) Wann ist ein Algorithmus nicht deterministisch?
   c) Ist jeder deterministische Algorithmus determiniert? Begründen Sie Ihre Antwort.
   d) Ist jeder determinierte Algorithmus auch deterministisch? Begründen Sie Ihre Antwort anhand von Beispielen.

4. **Aufgabe** (2 P.)
   Eine nebenläufige, asynchrone Ausführung von Prozessen führt immer zu nicht deterministischen Abläufen? Begründen Sie Ihre Antwort.

5. **Aufgabe** (6 P.)
   Betrachten Sie folgende Prozesse **P** und **R**, die nebenläufig ausgeführt werden sollten:



   a) Wenn Sie annehmen, dass **x**, **y** und **z** gemeinsame Variablen sind, die mit **0** initialisiert werden und dass jede Zuweisung atomar ausgeführt wird.

      Welche Wertkombinationen von **x**, **y** und **z** können am Ende erwartet werden? Begründen Sie Ihr Antwort.

   b) Wie viele verschiedene verzahnte Ausführungen sind möglich? Begründen Sie Ihre Antwort.

6. Aufgabe (4 P.)

Betrachten Sie folgende Java-Threads mit entsprechenden run()-Methoden:

Thread 1	Thread 2

…
public void run(){
  while (Share.COUNTER<8)
  Share.COUNTER++;
}
…
	…
	public void run(){
	  while (Share.COUNTER>-7)
	  Share.COUNTER--;
	}
	…

a) Welches Thread beendet die Schleife zuerst?

b) Können Sie garantieren, dass beide Threads die Ausführung der run()-Methode beenden? Begründen Sie Ihre Antworten.

7. Aufgabe (8 P.)

a) Für die Programmierung dieser Aufgabe definieren Sie zuerst eine Klasse **Nap**, die unter Anwendung der **sleep**-Methode der Thread-Klasse mindestens folgende statische Methoden enthält:

```
public static void nap( int milliSekunden );
public static void randomNap( int minMillisekunden, int maxMillisekunden );
```

b) In dieser Aufgabe sollen Sie eine **Gambler** Klasse definieren, die in der Lage ist, Glücksspieler-Objekte als Threads zu produzieren. Die Glücksspieler wetten parallel und unabhängig von einander. Alle Spieler starten mit 5 Dollar und nach jedem Spiel verlieren oder gewinnen sie einem Dollar. Die Spieler hören auf zu spielen, wenn diese kein Geld mehr haben oder sich ihr Budget verdoppelt haben. Eine Denkpause der Spieler zwischen den Wetten soll mit Hilfe der *randomNap*-Methode simuliert wird.

Die Simulation wird beendet, wenn alle Spieler aufgehört haben zu spielen.

Der gesamte Spielverlauf soll protokolliert und als Ergebnis der Simulation ausgegeben werden.

(2 Bonuspunkte) Für besondere schöne Visualisierungen des Spielverlaufs.

8. Aufgabe (3 P.)

Nehmen Sie an, dass Sie eine unbegrenzte Anzahl von Prozessen haben, die ohne zeitliches Overhead gestartet werden können. Wie würden Sie damit ein Array mit **n** Zahlen zusammen addieren?

a) Wie viele Prozesse können sinnvollerweise maximal verwendet werden?

b) Analysieren Sie die Zeitkomplexität des Algorithmus.

Wichtige Hinweise zur Abgabe:

1. Es muss der Quellcode hochgeladen werden (.java-Dateien), nicht der kompilierte Bytecode (.class-Dateien).
2. Zusätzlich zur Online-Abgabe muss der Quellcode vollständig ausgedruckt werden. Der ausgedruckte Quellcode muss lesbar sein, insbesondere ist auf automatische Zeilenumbrüche zu achten.
3. Für die gute Verständlichkeit des Quellcodes sind sinnvolle Bezeichnernamen zu wählen und der Code ausreichend zu kommentieren.
4. Das Programm muss Testläufe enthalten, die die Aufgabe vollständig abdecken. Das heißt, zum Testen sollen keine Änderungen am Code durch die Tutorin/den Tutor mehr notwendig sein. Die Testläufe müssen in einer eigenen Klasse Main enthalten sein.
5. Das Programm muss mit Java 8 kompatibel sein.
6. Alle Übungsaufgaben, die nicht Programmieraufgaben sind, müssen ebenfalls digital und nicht handschriftlich sowohl online als auch ausgedruckt zu dem im KVV angegebenen Datum (in der Regel Dienstag um 11:55) abgegeben werden. Zu spät abgegebene Lösungen werden mit 0 Punkten bewertet.