SoSe 2017

Nichtsequentielle und verteilte Programmierung

5. Übungsblatt

Prof. Dr. Margarita Esponda

Ziel: Auseinandersetzung mit Deadlocks.

1. Aufgabe (6 P.)

Beweisen oder widerlegen Sie folgende Implikationen:

$$\Box A \wedge \Diamond B \implies \Diamond(A \wedge B)$$

$$\Box(A \vee B) \implies (\Box A \vee \Diamond B)$$

$$\Diamond A \wedge \Diamond B \implies \Diamond(A \wedge B)$$

2. Aufgabe (4 P.)

Im folgenden Algorithmus sollen mit Hilfe von binären Semaphoren n Threads synchronisiert werden, sodass sich maximal k Threads gleichzeitig in dem kritischen Abschnitt befinden.

```
binary semaphore  S ← 1
binary semaphore  D ← 0
integer count ← k
```

```
    integer temp
    loop forever
p1:     NCS
p2:     acquire(S)
p3:     count ← count - 1
p4:     temp ← count
p5:     release(S)
p6:     if temp ≤ -1
p7:         acquire(D)
p8:     CS
p9:     acquire(S)
p10:    count ← count + 1
p11:    if count ≤ 0
p12:        release(D)
p13:    release(S)
```

Finden Sie für n=4 und k=2 ein Szenario, bei dem D den Wert 2 hat.

Das widerspricht eigentlich der Behauptung, dass S und D nur binäre Semaphoren sind.

Zeigen Sie das Gleiche für n=3 und k=2.

3. Aufgabe (8 P.)

In dieser Aufgabe sollen Sie ein Java Programm schreiben, dass die Bindung von Wassermolekülen mit Hilfe von Threads simuliert.

Es sollen Wasserstoff und Sauerstoff Threads produziert werden, die mit Hilfe eines Barrier zu einem Molekül synchronisiert werden.

Wenn ein Sauerstoff Thread an die Barrier ankommt und kein Wasserstoff Thread da ist, muss es warten, bis zwei Wasserstoff Threads ankommen.

Wenn ein Wasserstoff Thread an die Barrier ankommt und keine weiteren Threads da sind, muss es auf ein Wasserstoff und ein Sauerstoff Thread warten.

Die Synchronisation besteht daraus, dass nur Thread-Gruppen, die aus einem Sauerstoff und zwei Wasserstoff Threads bestehen, die Barrier verlassen können. Sie müssen dafür eine korrekte Wasserstoff und Sauerstoff Klasse von Thread-Objekten definieren, die mit diesen Einschränkungen korrekt funktionieren.

Argumentieren Sie genau, warum Ihre Lösung funktioniert. Postulieren Sie Invarianten, die für das korrekte Verhalten des System gelten müssen.

4. Aufgabe (3 P.)

Zeichnen Sie die graphischen Darstellungen des folgenden „Resource-Allocation"-Graphen:

$$T = \{T_1, T_2, T_3, T_4, T_5\}$$
$$R = \{ R_1, R_2, R_3, R_4\} \quad R_1=2, \quad R_2=1, \quad R_3=3, \quad R_4=1$$
$$E = \{ R_1 \rightarrow T_1, R_1 \rightarrow T_2, R_2 \rightarrow T_1, R_3 \rightarrow T_2, R_3 \rightarrow T_4, R_3 \rightarrow T_5, R_4 \rightarrow T_3,$$
$$T_1 \rightarrow R_3, T_2 \rightarrow R_2, T_3 \rightarrow R_2, T_3 \rightarrow R_3, T_5 \rightarrow R_4 \}$$

Ist eine Verklemmung aus dem Graph zu erkennen?

Befindet sich der Graph in einem von Deadlock gefährdeten Zustand?

Begründen Sie Ihre Antworten.

5. Aufgabe (5 P.)

Nehmen Sie an, es gibt ein System mit **n** Ressourcen der gleichen Ressourcen-Klasse, die von **m** Threads beansprucht werden können.

a) Die Threads können jedes mal nur einzelne Ressourcen in Anspruch nehmen oder frei geben.

b) Die maximale Anforderung der einzelnen Threads liegt zwischen **1** und **n**.

c) Die Summe der maximalen Anforderungen aller Threads ist kleiner als **n+m**.

Zeigen Sie, dass unter den Bedingungen a), b) und c) das System verklemmungsfrei ist.

Allgemeine Wichtige Hinweise zur Übungsabgaben:

1. Es muss der Quellcode hochgeladen werden (.java-Dateien), nicht der kompilierte Bytecode (.class-Dateien).

2. Zusätzlich zur Online-Abgabe muss der selbstprogrammierten Quellcode 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.