HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 9

Betriebssysteme

Luis Staudt

Aufgabe 1: Körner essen

Fragen zur Methode `bearbeiteKachel`

Frage 1: Mit welcher Methode von `Territorium` kann man die Zahl der Körner um eins verringern?

Antwort: Mit der Methode `dekrementAndGet()` des `AtomicInteger` Objekts. Diese Methode ermöglicht es mehreren Threads gleichzeitig, den Wert atomisch um eins zu erniedrigen.

Frage 2: Studieren Sie den Code der Methode. Warum müsste man diese Methode `synchronized` machen und warum kann man sich das trotzdem sparen?

Antwort:

- **Warum synchronized nötig wäre:** Mehrere Threads greifen gleichzeitig auf dasselbe `zahlKoerner[i][j]` Attribut zu, was zu Race Conditions führen kann (Register Problem aus der Vorlesung).
- **Warum es trotzdem nicht nötig ist:** `AtomicInteger` ist bereits thread-safe und bietet atomare Operationen ohne explizite Synchronisation.

Frage 3: Wie kann man die `CyclicBarrier` verwenden, damit jeder Esser die ursprüngliche Zahl der Körner sieht?

Antwort:

1. Alle Threads lesen zuerst die ursprüngliche Körnerzahl
2. Dann warten alle Threads an der `CyclicBarrier` mit `await()`
3. Erst wenn alle 3 Threads die ursprüngliche Zahl gelesen haben, werden sie gleichzeitig freigegeben
4. Danach können alle gleichzeitig die Körnerzahl reduzieren

Fragen zur Methode `main`

Frage: Warum muss man den Threadpool in unserer Aufgabe mit genau drei Threads bestücken?

Antwort: Die `CyclicBarrier` wird im Konstruktor auf eine feste Anzahl von Threads eingestellt (hier: 3). Das `await()` blockiert solange, bis genau diese Anzahl erreicht ist:

- Bei weniger als 3 Threads: Die Threads bleiben in `await()` hängen
- Bei mehr als 3 Threads: Überzählige Threads werden nicht synchronisiert

Aufgabe 2: Ressourcen Manager

Banker's Algorithm - Testfälle

Ausgangssituation:

$$\text{Thread A:} \quad \text{Max: } 9 \tag{0.1}$$

$$\text{Thread B:} \quad \text{Max: } 4, \text{ Aktuell: } 2 \tag{0.2}$$

$$\text{Thread C:} \quad \text{Max: } 7, \text{ Aktuell: } 2 \tag{0.3}$$

$$\text{Gesamte Betriebsmittel:} \quad 10 \tag{0.4}$$

Thread	Aktuell	Maximum	Restforderung
A	3	9	6
B	2	4	2
C	2	7	5

Testfall 1: A hat aktuelle Belegung 3 (SICHER) Freie Betriebsmittel: $10-3-2-2=3$
Sichere Ausführungsreihenfolge:

1. B kann fertig werden (Restforderung $2 \leq 3$ frei)
2. Nach B: $3+2=5$ freie Betriebsmittel
3. C kann fertig werden (Restforderung $5 \leq 5$ frei)
4. Nach C: $5+2=7$ freie Betriebsmittel
5. A kann fertig werden (Restforderung $6 \leq 7$ frei)

Thread	Aktuell	Maximum	Restforderung
A	4	9	5
B	2	4	2
C	2	7	5

Testfall 2: A hat aktuelle Belegung 4 (UNSICHER) Freie Betriebsmittel: $10-4-2-2=2$

Deadlock-Szenario:

- Nur B kann fertig werden (Restforderung $2 \leq 2$ frei)
- Nach B: $2 + 2 = 4$ freie Betriebsmittel
- Weder A noch C können fertig werden (beide brauchen 5)
- $\Rightarrow$ Potentieller Deadlock!