HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 3

Betriebssysteme

Luis Staudt

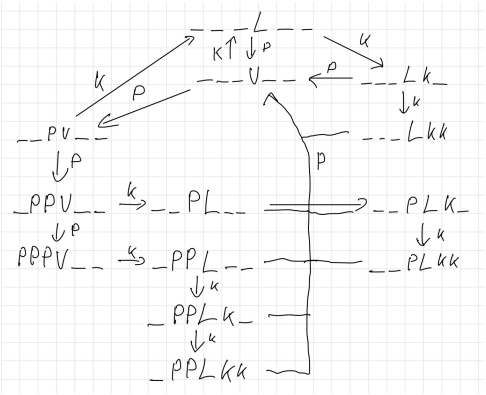
Aufgabe a)

I

Durch das notfiyAll() kann es nicht zu einer Verklemmung kommen, lediglich die Ausführungsdauer könnte länger werden.

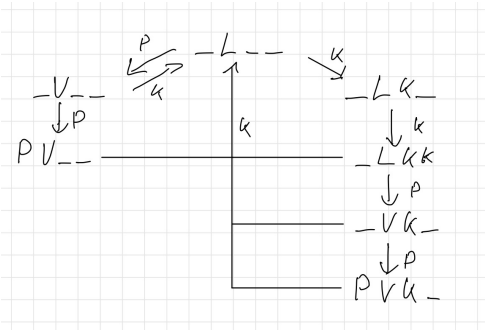
II

Keine Verklemmung möglich.



III

Auch keine Verklemmung möglich. Konsumenten wecken immer alle Threads nach dem leeren. Produzent weckt immer einen Konsumenten.



IV

Ändert nichts, da erst nach Beenden der Methode freigebe geben wird.

Aufgabe b)

II

1. Mehrfachvererbung: Java unterstützt keine Mehrfachvererbung von Klassen. Wenn eine Klasse bereits von einer anderen Klasse erbt, kann sie nicht gleichzeitig von Thread erben, kann aber das Runnable-Interface implementieren.
2. Separation of Concerns: Die Verwendung von Runnable ermöglicht eine bessere Trennung zwischen dem Verhalten des Threads (die run-Methode) und der Thread-Verwaltung selbst.
3. Wiederverwendbarkeit: Ein Runnable-Objekt kann an mehrere Threads übergeben werden, wodurch dieselbe Aufgabe von verschiedenen Threads ausgeführt werden kann.
4. Ressourcenfreundlicher: Die Implementierung von Runnable benötigt weniger Ressourcen, da nicht für jede Task eine vollständige Thread-Unterklasse erstellt werden muss.
5. Thread-Pools: In modernen Java-Anwendungen mit Thread-Pools (z.B. via Executor-Service) werden typischerweise Runnable-Tasks verwendet.

III

Möglicher Programmablauf mit vermischten Druckjobs:

1. Hänsel-Thread startet und beginnt mit dem Drucken von haensel.txt.
2. Hänsel-Thread druckt “haensel 1“.
3. Der Scheduler unterbricht Hänsel und wechselt zum Gretel-Thread.
4. Gretel-Thread beginnt mit dem Drucken von gretel.txt.
5. Gretel-Thread druckt “gretel 1“.
6. Scheduler wechselt zurück zum Hänsel-Thread.
7. Hänsel-Thread druckt “haensel 2“ und “haensel 3“.

8. Scheduler wechselt zum Gretel-Thread.

9. Gretel-Thread druckt “gretel 2“ und “gretel 3“.

Mögliche Ausgabe: haensel 1 gretel 1 haensel 2 haensel 3 gretel 2 gretel 3