

BAI3 BSP	Praktikum Betriebssysteme	Tdm/Slz
SS 2017	Aufgabe 3 – Thread-Synchronisation in Java	Seite 1 von 2

1. Simulation von Mensakassen (Synchronisation über Semaphore)

Es soll ein Programm zur Simulation von mehreren Mensakassen entwickelt werden. Wenn ein Student (Kunden sind zur Vereinfachung ausschließlich Studenten) bezahlen möchte, geht er zu der Kasse, an der die wenigsten Studenten warten (falls es Warteschlangen gibt). Wenn er an der Reihe ist, bezahlt er. Während des Bezahlens ist die Kasse für ihn in exklusivem Zugriff, d.h. für andere Studenten gesperrt. Nachdem er bezahlt hat, ist er („warten“ des Threads: wie zu implementieren? Wie nicht zu implementieren?, Dauer: Zufallszeit) und kommt irgendwann wieder (ebenfalls „warten“ für eine Zufallszeit).

1.1. Schreiben Sie ein JAVA-Programm, welches eine Mensa mit beliebig vielen Kassen (konfigurierbar, s.u.) sowie den Zugang der Studenten mit Hilfe der Klassen `Semaphore` bzw. `ReentrantLock` simuliert. Benutzen Sie hierfür **unbedingt** die passenden Standard-Klassen aus dem Package `java.util.concurrent`. Jeder Student soll dabei als eigener Thread modelliert werden. Weitere Threads müssen Sie nicht erzeugen, können dies aber. Die Simulation soll nach einer vorgegebenen Zeit durch den Aufruf der Methode `interrupt()` beendet werden.

1.2. Testen Sie Ihr Programm mit 10 Studenten-Threads und 1, 2 und 3 Kassen! Testen Sie jeweils **mehrmals** und suchen und beheben Sie auch solche Fehler, die nur manchmal, sporadisch auftreten!

Tipps:

- Erzeugen Sie einen Thread je Student und geben Sie jedem Studenten-Thread bei der Erzeugung Informationen über alle Kassen (z.B. in Form einer Kassenliste), damit er seine Auswahlentscheidung treffen kann.
- Denken Sie daran, dass **alle** Zugriffe auf Objekte, die von mehreren Threads verändert werden können, synchronisiert werden müssen!

2 Schere, Stein, Papier (Synchronisation über JAVA-Objektmonitor / ConditionQueues)

Zwei Threads sollen das bekannte Spiel „Schere, Stein, Papier“ (auch *Schnick, Schnack, Schnuck* oder *Ching, Chang, Chong* genannt) gegeneinander spielen (siehe https://de.wikipedia.org/wiki/Schere,_Stein,_Papier). Da Threads keine Hände besitzen, muss jeder Thread in jeder Runde ein Spielobjekt (entweder ein Schere-, Stein oder Papierobjekt) auf einen virtuellen Tisch legen. Anschließend nimmt ein Schiedsrichter-Thread die Auswertung vor und leert den Tisch wieder. Es soll solange eine Runde nach der anderen gespielt werden, bis nach einer vorgegebenen Zeit alle Threads mittels des Interrupt-Mechanismus beendet worden sind. Danach soll eine Gesamtauswertung (Gesamtanzahl gespielter Runden, Anzahl Unentschieden, Anzahl Gewinne Thread 0 / Thread 1) ausgegeben werden.

2.1 Schreiben Sie ein JAVA-Programm, welches das o.g. Spiel mittels JAVA-Threads realisiert. Benutzen Sie die JAVA-Synchronisationsmechanismen (Eintritt in einen Objekt-Monitor mittels „synchronize“) und die Methoden zur Threadsynchronisation (`wait()`, `notify()`, `notifyAll()`).

Testen Sie Ihr Programm mehrfach und dokumentieren Sie die erfolgreiche Spieldurchführung durch geeignete Testausgaben! Beheben Sie auch solche Fehler, die nur manchmal, sporadisch auftreten!

BAI3 BSP	Praktikum Betriebssysteme	Tdm/Slz
SS 2017	Aufgabe 3 – Thread-Synchronisation in Java	Seite 2 von 2

2.2. Erzeugen Sie eine weitere Programmversion, die für die Synchronisation ausschließlich die im JAVA-Package `java.util.concurrent.locks` zur Verfügung gestellten Mechanismen (`locks` und `conditions`) mit mehreren `ConditionQueues` benutzt.

Testen Sie Ihr Programm mehrfach und dokumentieren Sie die erfolgreiche Spieldurchführung durch geeignete Testausgaben! Beheben Sie auch solche Fehler, die nur manchmal, sporadisch auftreten!

Tipps:

- **Führen Sie die Problemstellung auf das aus der Vorlesung bekannte Synchronisationsproblem „Erzeuger/Verbraucher“ zurück und orientieren Sie sich am Beispiel-Code aus der Vorlesung!**
- Für die Repräsentation der Spielobjekte (Schere, Stein, Papier) bietet sich das `enum`-Konstrukt in Java an, da `enum`-Konstanten Java-Objekte sind, aber auch über einen Index (Methode `ordinal()`) verfügen und daher für Array-Zugriffe verwendbar sind.
- Stellen Sie die Spielregeln bzgl. der Gewinnermittlung (siehe https://de.wikipedia.org/wiki/Schere,_Stein,_Papier#Grundregeln) als Matrix dar!