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, wählt er zunächst die Kasse aus, an der die wenigsten Studenten warten, und stellt sich dort an. 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, isst er (Dauer: Zufallszeit) und kommt irgendwann wieder (ebenfalls nach einer Zufallszeit).

1.1. Schreiben Sie ein JAVA-Programm, welches eine Mensa mit beliebig vielen Kassen und Studenten sowie den oben beschriebenen Bezahlvorgang der Studenten simuliert. Benutzen Sie hierfür die Klasse **`ReentrantLock`** aus dem Package **`java.util.concurrent.locks`**. Jeder Student soll dabei als eigener Thread modelliert werden. Nach einer vorgegebenen Zeit sollen alle Threads durch den Aufruf der Methode **`interrupt()`** beendet werden.

1.2. Testen Sie Ihr Programm mit 10 Studenten-Threads und 1, 2 und 3 Kassen!

Tipps:

- *Geben Sie jedem Studenten-Thread bei der Erzeugung Informationen über alle Kassen (z.B. in Form einer Kassenliste), damit er seine Auswahlentscheidung treffen kann.*
- *Die Methode `getQueueLength()` der Klasse `ReentrantLock` liefert nur einen Schätzwert (siehe API-Doku), außerdem ist nicht erkennbar, ob bereits jemand an der Kasse bezahlt. Verwalten Sie daher stattdessen in jeder Kasse selbst einen einfachen Zähler!*
- *Denken Sie daran, dass alle Zugriffe auf Objekte, die von mehreren Threads verändert werden können, synchronisiert werden müssen!*

2. Smoker-Problem (Synchronisation über JAVA-Objektmonitor)

Ein Problem aus einer langsam aussterbenden Welt: Wir betrachten ein System mit drei Rauchern („smoker“) und zwei Agenten. Jeder Raucher rollt sich so oft wie möglich eine Zigarette und raucht diese anschließend. Um eine Zigarette rollen und rauchen zu können benötigt ein Raucher drei Zutaten: *Tabak*, *Papier* und *Streichhölzer*. Einer der drei Raucher hat Tabak, ein anderer hat Papier und der Dritte hat Streichhölzer. Die beiden Agenten haben eine unendliche Menge von allen drei Dingen. Alle sitzen um einen Tisch versammelt.
Das Verhalten in diesem System ist folgendermaßen:

1. Einer der Agenten legt zwei der drei Zutaten auf den Tisch (zufällig ausgewählt).
2. Der Raucher, der die fehlende dritte Zutat besitzt, nimmt die beiden Zutaten vom Tisch, rollt sich eine Zigarette und raucht sie anschließend (das kostet Zeit!). Wenn er mit dem Rauchen fertig ist, signalisiert er dies den Agenten.
3. Ein Agent beginnt wieder einen neuen Zyklus → Schritt 1.

Schreiben Sie ein JAVA-Programm, welches das o.g. Systemverhalten mit zwei Agenten-Threads und drei Smoker-Threads simuliert!

- Benutzen Sie die JAVA-Synchronisationsmechanismen (Eintritt in einen Objekt-Monitorbereich mittels **`synchronize`**) und die Methoden zur Threadsynchronisation (**`wait()`, `notify()`, `notifyAll()`**).
- Geben Sie jedem Thread einen aussagekräftigen Namen und lassen Sie eine Ausgabe erzeugen, durch die der jeweilige Zustand der Threads und des Tisches dokumentiert wird.
- Nach einer vorgegebenen Zeit sollen alle Threads durch den Aufruf der Methode **`interrupt()`** beendet werden.

Tipp: *Führen Sie die Problemstellung auf das aus der Vorlesung bekannte Synchronisationsproblem „Erzeuger/Verbraucher“ zurück und orientieren Sie sich am Beispiel-Code aus der Vorlesung!*