

PM2, Aufgabenblatt 4

Programmieren 2 – Sommersemester 2017

Nebenläufigkeit und Synchronisation

Ausgabedatum: 12. Juni 2017





Kernbegriffe

Nebenläufigkeit ist bei grafischen Benutzungsoberflächen der Normalfall. Wenn mehrere parallele Prozesse auf denselben Daten arbeiten, müssen sie *synchronisiert* werden. In Java ist das *Monitor-Konzept* als Standardmechanismus innerhalb der Sprache umgesetzt: Jedes Objekt hat einen eigenen Lock und kann als Monitor benutzt werden; dazu muss das Schlüsselwort `synchronized` verwendet werden, entweder für ganze Methoden oder für einzelne Blöcke. In der Java-API gibt es seit Version 1.5 weitere Mechanismen zur Synchronisation, unter anderem auch *Semaphore*.

Aufgabe 4.1 Kinoticketverkauf an mehreren Kassen

In jedem größeren Kino gibt es mehrere Kassen, an denen die Angestellten gleichzeitig Tickets verkaufen können. Die Daten, die dabei bearbeitet werden, liegen auf einem gemeinsamen Datenspeicher. In einer realistischen Umgebung hat jede Kasse einen eigenen Rechner, so dass es sich üblicherweise um ein verteiltes System handelt.

Wir wollen Nebenläufigkeit in unserem Kinoticketssystem nur simulieren, indem wir für jede Kasse in einem separaten Fenster ein eigenes Kassensystem zeigen. Alle Fenster sollen aber auf denselben Materialien arbeiten.

- 4.1.1 Baut das Kinoticketverkaufssystem (samt der von euch implementierten Barzahlung) so um, dass beim Starten mehrere Fenster mit dem Kassensystem angezeigt werden (mindestens zwei). In jedem Fenster soll der Verkauf von Tickets **desselben Kinos** möglich sein. Kümmert euch noch nicht um die Synchronisation!
-  4.1.2 Versucht in diesem System, fehlerhafte Situationen zu erzeugen. Welche Probleme treten auf? Überlegt euch **Benutzungsszenarien** und haltet diese **schriftlich** fest, damit ihr sie bei Bedarf für weitere Tests wiederholen könnt. Eure Beschreibung sollte so klar sein, dass ein anderes Paar im Praktikum diese leicht durchspielen kann.
-  4.1.3 Überlegt euch weiterhin, welche fehlerhaften Situationen entstehen könnten, wenn die Fenster wirklich nebenläufig wären (sind sie nicht, ihr könnt ja nur in ein Fenster zur gleichen Zeit tippen). Überlegt euch auch hier Benutzungsszenarien. **Schriftlich!**
-  4.1.4 Überlegt euch, wie ihr die auftretenden Probleme beheben könnt. Müsst ihr synchronisieren? Wenn ja, mit welchen Mechanismen? Sollte ein Service eingeführt werden? Haltet eure Entwurfsüberlegungen zuerst **schriftlich** fest. Bei der Abnahme solltet ihr anfangs euren Entwurf zur Diskussion stellen. Einige Hinweise:
 - In nebenläufigen Systemen ist das Prüfen der Vorbedingung einer Methode mit anschließendem Aufruf der Methode nicht unteilbar.
 - Ihr benötigt keine explizit erzeugten Threads für eure Lösung.
 - Falls ihr einen Benachrichtigungsmechanismus plant: beachtet, dass Materialien nicht beobachtet werden dürfen (siehe Entwurfsregeln).
-  4.1.5 Implementiert eure Lösung. Alle Klassen des Systems können dazu bearbeitet werden. **Dokumentiert geeignet**, welche Klassen geändert wurden, etwa mit Hilfe eines Klassendiagramms. Zeigt anhand eurer Benutzungsszenarien, dass das System auch mit mehreren Kassensystemen korrekt funktioniert.