Praktikum 4 Nebenläufigkeit Frederic Dlugi & Maximilian Mang

4.1.2) Durch das erstellen von zwei Kassenwerkzeugen, die beide auf das selbe Kino zugreifen, entstehen folgende Probleme:

* Es könnten Plätze doppelt verkauft werden, wenn der einer den Platz ausgewählt hat (und später kauft), während der andere die Bezahlung abschließt.
  + Das gleiche gilt für stornieren
* Ein augenscheinlich verkaufte Platz kann neu verkauft werden, wenn er im anderen Tool storniert wurde.
* Ein augenscheinlich verfügbarer Platz kann nur storniert werden, wenn er im anderen Tool verkauft wurde.
* Während ein Verkaufswerkzeug Änderungen am Kino macht, wird bei dem anderen Werkzeug der Platzplan nicht aktualisiert. ( Mit aktualisierePlatzplan() )

4.1.3) Probleme die durch echte Nebenläufigkeit entstehen wären:

* Bei gleichzeitigem zugriff auf das Platz-Array in der Vorstellung, würden Lost-Update-Probleme auftreten, die manche Änderungen von den Kassenwerkzeugen fallen lassen.

4.1.4.) Um die Probleme zu beheben, die nicht durch echte Nebenläufigkeit ausgelöst werden, möchten wir Plätze sperren, die durch ein anderes Tool ausgewählt sind. Um dies zu realisieren, planen wir die Klasse Vorstellung hinter einen Vorstellungsservice zu packen, wobei der Vorstellungsservice die gleichen Methoden anbieten soll wie die alte Vorstellung. Die Vorstellung soll außerdem ein weites Platz-Array besitzen, das gesperrte Plätze beinhalten soll (Genauso wie Getter und Setter).

Der Vorstellungsservice ist wichtig, damit die Kinokassen über Änderungen informiert werden (wenn der andere etwas ändert). Der Vorstellungsservice wird deshalb ein Observable sein.

Die durch echte Nebenläufigkeit verursachten Probleme, werden wir beheben in dem wir die Methoden, die einen kritischen Abschnitt haben, Synchronized machen. Dadurch wird das Lost-Update-Problem verhindert.