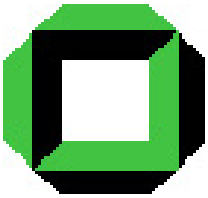

UNIVERSITÄT KARLSRUHE (TH)
Fakultät für Informatik
System Architecture Group
Frank Bellosa, Gerd Liefländer, Philipp Kupferschmied
Dominik Bruhn, Atanas Dimitrov,
Jonathan Dimond, Johannes Weiß

Basispraktikum Systemarchitektur - WS 2008/2009

Ticketverkauf im Reisebüro

Das Szenario

Die Grundlage dieses Versuchs bildet ein vereinfachtes Modell eines Reisebüros.

Am Ende soll eine feste Anzahl von Flügen (mit Datum bzw. Flug-ID, Anzahl der Sitzplätze etc.) als Ticketkontingent vorhanden sein. An einer festen Zahl von Schaltern werden die Kunden bedient. Der Kunde an sich kann 2 Aktivitäten ausführen. Zum einen kann er sich ausschließlich beraten lassen (500ms) oder er kann direkt buchen, d.h. er weiß wo er wann hinfliegen möchte, wie viele Plätze er buchen will, (diese brauchen nicht alle nebeneinander zu liegen etc. (im optionalen Teil können auch Sitzplatz-Präferenzen berücksichtigt werden)). Will der Kunde die Tickets kaufen, so beschreibt er dem Reisebüro-Angestellten zunächst, was er genau möchte (120 ms), danach werden die angeforderten Tickets gebucht (250 ms) und anschließend die Tickets an den Kunden übergeben (80 ms).

Da wir hier nur ein vereinfachtes Modell betrachten wollen, gehen wir davon aus, dass ein Kunde nur eine Strecke buchen möchte und auch nur einen einfachen Flug.

Wie sie die Phasen von Beratung, Anfrage, Buchung und Aushändigung der Tickets gestalten (Text, Begrüßung etc.) bleibt Ihnen überlassen.

Vorüberlegungen:
Überlegen Sie sich ein festes Kontingent an Flugstrecken, die Sie im Reisebüro anbieten wollen. Zu jeder Flugstrecke gehört ein genau ein Flugzeug. Die zur Flugstrecke zugehörigen Flugzeuge selbst haben ein bestimmtes Sitzplatzkontingent und ein Abflugdatum bzw. eine Flug-ID (um den Flug zu identifizieren).

Vorgaben

Jeder Kunde soll durch einen eigenständigen Thread modelliert werden, ebenso wie jeder Reisebüro-Angestellter.

Da am Ende an mehreren Schaltern gleichzeitig gearbeitet wird, ist es wichtig, dass die Reisebüro-Angestellten-Threads nie gleichzeitig beim Buchungsvorgang auf gemeinsame Daten zugreifen. Verwenden Sie dazu die von java bereitgestellten Synchronisationsmechanismen wie synchronized, wait() und notify().

Hinweis: *Entwickeln Sie ihre Klassen und Methoden möglichst so, dass sie gut in den einzelnen Teilaufgaben wieder verwendet werden können.*

Aufgabe 1: Ein einsamer Reisebüro-Angestellter mit 3 Flugstrecken

Schreiben Sie einen Thread, der das Verhalten eines Reisebüro-Angestellten modelliert. Dazu gehört: Kundenwünsche anzunehmen, den Buchungsvorgang durchzuführen und die Tickets an den Kunden zu übergeben oder alternativ den Kunden ausschließlich zu beraten. (*Kann ein Kundenwunsch nicht erfüllt werden (können nicht alle Plätze gebucht werden, dann soll keiner gebucht werden), so wird der Kunde zunächst nach Hause geschickt.*)

Schreiben Sie dann auch eine Menge von Kunden-Threads, die verschiedene Verhaltensmuster aufweisen (Beratungsgespräch oder Ticketkauf (Flug, Anzahl der Sitzplätze) und sich n mal nacheinander (mit Pausen von t msec) anstellen. Die Kunden-Threads sollen sich beim Eintreffen in das Reisebüro in die Warteschlage des Schalters einreihen.

Sind alle Flugzeuge voll besetzt oder alle Kunden-Threads komplett abgearbeitet, so soll das Programm beendet werden.

Aus dem Ganzen soll ein Programm resultieren, das alle Threads startet und zunächst textuell ausgibt, wie sich die aktuelle Flugzeug-Belegung verändert (freie/belegte Plätze) *(da in der abschließenden Teilaufgabe eine visuelle Ausgabe erforderlich ist, sollten Sie bereits jetzt an die dafür notwendigen Komponenten denken)*

Die Parameterübergabe (Kundenanzahl / Anzahl, wie oft jeder Kunde ins Reisebüro kommt / Pause, nach der sich die Kunden erneut ins Reisebüro begeben) soll über die Konsole erfolgen.

Anmerkung: Die Pause und die Anzahl, wie oft ein Kunde ins Reisebüro kommt, kann auch für jeden Thread individuell sein. Z.B. kann ein Minimum übergeben werden und die Threads benutzen jeweils ein Vielfaches dieses Parameters.

Aufgabe 2: neue Mitarbeiter, neue Flugstrecken

Um den einsamen Mitarbeiter nicht weiter zu überlasten werden nun insgesamt k ($2 <= k <= 5$) Mitarbeiter im Reisebüro beschäftigt und insgesamt $m > 5$ Flugstrecken angeboten.

Überarbeiten Sie in diesem Teilversuch nochmals den Buchungsvorgang im Hinblick auf die notwendigen Synchronisationsmechanismen, damit bei der Flugbuchung garantiert nichts schief geht und bauen Sie Ausgaben ein, um dies zu überprüfen. Überlegen Sie sich dazu vor allem, ob der komplette Buchungsvorgang als kritischer Abschnitt zu bezeichnen ist oder ob nur ein Teil daraus als kritisch zu behandeln ist.

Überlegen Sie sich, wie sie bei mehreren Angestellten die Kunden in Warteschlangen einreihen (eine Warteschlange und die Kunden gehen immer zum nächsten freien Schalter oder k Warteschlangen) wollen.

Auch hier soll die Ausgabe wieder textuell erfolgen. Dabei sollte zu erkennen sein, welcher Angestellter welche Plätze gebucht hat und auch, ob die Kundenwünsche mit der endgültigen Belegung übereinstimmen (als Kontrolle dafür, dass alles glatt gelaufen ist).

Die Parameterübergabe (Anzahl der Angestellten / Anzahl der Flüge / Anzahl der Kunden / Anzahl, wie oft jeder Kunde ins Reisebüro kommt / Pause, nach der sich die Kunden erneut ins Reisebüro begeben) soll auch hier über die Konsole erfolgen (die Pause und die Anzahl, wie oft ein Kunde ins Reisebüro kommt soll hier individuell für jeden Thread sein (vgl. Realisierungsvorschlag von Aufgabe 1).

Aufgabe 3: Visualisierung

In dieser Teilaufgabe soll die Funktionalität des Reisebüros an sich nicht mehr erweitert werden, sondern nur noch eine graphische Oberfläche für das ganze Szenario entwickelt werden. Graphisch ausgegeben werden soll dabei:

- die aktuelle Flugzeugbelegung aller Flugstrecken
- eine Übersicht darüber, wie viele Kunden sich an welchem Schalter angestellt haben und wie viele Plätze an jedem Schalter für welche Strecke gebucht werden (insgesamt)
- eine Übersicht über die Warteschlange / Warteschlangen in die sich die Kunden einreihen

Die Parameter sollen diesmal ebenfalls über die graphische Oberfläche übergeben werden.

Aufgabe 4 (optional)

Um die ganze Modellierung etwas realitätsnäher zu gestalten, gibt es jetzt unter anderem noch folgende Möglichkeiten das Ganze zu erweitern.

1. Für jede Strecke werden zusätzlich verschiedene Flugtermine angeboten und der Kunde muss entscheiden, an welchem Tag er fliegen möchte.
2. Ist Punkt 1 realisiert, so kann ein Kunde, der leider kein Ticket für den gewünschten Flug mehr bekommen kann, auf einen anderen Termin verwiesen werden.
3. Der Kunde soll Sitzplatzpräferenzen haben (z.B. Fensterplatz oder mehrere Plätze nebeneinander), die nach Möglichkeit umgesetzt werden sollten. Ist der Kundenwunsch so direkt jedoch nicht erfüllbar, muss eine Alternative gefunden werden, die der Kunde akzeptiert.