Universität Koblenz-Landau WS '04/05
Institut für Informatik
Bernhard Beckert · www.uni-koblenz.de/~beckert
Vladimir Klebanov · www.uni-koblenz.de/~vladimir

Praktikum

Entwicklung objektorientierter Software mit formalen Methoden

Aufgabenblatt 4

Aufgabe 9

Verbessern und Sie das Klassendiagramm und passen Sie es an das in der Sitzung am 07.12. besprochene an. Dazu einige Bemerkungen zur Beziehung zwischen Events:

Es wurde besprochen, dass es zwischen Events eine hierarchischen Aggregationsbeziehung gibt (eine Session besteht aus Talks, Konferenz besteht aus Workshops, Tutorials und dem Technischen Programm, usw.). Dabei ist zu beachten, dass diese Beziehung nur bedeutet, dass ein Event als Teil (am Ort und zur Zeit) eines anderen Events stattfindet.

Daneben gibt es eine weitere wichtige Beziehung zwischen Events, nennen wir sie *usesServiceOf*. Diese bedeutet: Ist jemand registrierter (bezahlender) Teilnehmer von Event A, dann darf er auch an Event B teilnehmen (man sollte also zwischen den Rollen *participant* und *registeredParticipant* unterscheiden). Beispiele: Workshopteilnehmer dürfen an den Kaffeepausen während des Workshops teilnehmen, sowie ebenfalls Leistungen des Konferenzsekretariats (das man auch als Event auffassen kann) in Anspruch nehmen.

Aufgabe 10

Im folgenden werden die drei zu bearbeitenden Use Cases informell beschrieben. Präzisieren Sie diese Beschreibungen, so dass sie sich anschlissend ohne weiteres in formale Spezifikationen umgesetzt werden können. Wenn dazu Fragen offen sind, fomulieren Sie diese Fragen schriftlich.

Use Case 1: Rechnung Drucken Dieser Use Case besteht darin, die Rechnungseinträge für eine Rechnung zu bestimmen.

Jede Person kann für verschiedene Events registiert sein, die ihr in Rechnung gestellt werden. Dazu gehören das Technische Programm der Konferenz, die Workshops, die Tutorials und die sozialen Programmpunkte.

Der Preis eines Events ist abhängig von der buchenden Person (Student oder nicht) bzw. dem Zeitpunkt der Buchung (Early Registration, Late Registration, On-Site Registration).

Außerdem gibt es eine Rabattfunktion, die aus den nach den Registrierungen bestimmten Rechnungs-Items zusätzliche Rechnungs-Items, nämlich Ermässigungen berechnen kann. Die Ergebnisse dieser Funktion sind zu berücksichigen. Sie ist aber hier nicht weiter zu spezifizieren.

Schließlich steht in der Datenbank, welche Zahlungen schon geleistet wurden. Auch diese sind auf der Rechnung aufzuführen.

Die Rechnung für eine Person soll auch alle Gebühren für Event-Registrierungen, die im Namen einer begleitenden Person gebucht worden sind, enthalten.

Use Case 2: Gutscheinliste drucken Ähnlich wie bei der Erstellung der Rechnung, soll für jede Person (nicht für solche, die nur begleiten) ein Ausdruck mit Gutscheinen erstellt werden. Diese müssen nicht nur für die Events erstellt werden, für die man registriert ist, sondern auch für die anderen, für die man teilnahmeberechtigt ist.

Bei jedem Event muss gespeichert sein, ob und was für einen Gutschein er erfordert. Wie bei der Rechnung sollen alle Gutscheine einer Begleitperson auf die Liste der begleiteten Person kommen.

Use Case 3: Doppelte Personen-Einträge finden Es soll eine Funktion geben, die Personen-Objekte findet, die die gleiche Person repräsentieren. Die Entscheidung, ob Objekte wirklich die gleiche Person darstellen, wird dem Benutzer überlassen. Die Funktion sollte lieber zuviele mögliche Duplikate finden, als welche zu übersehen.

Die Funktion sollte sich nur auf Namen und Affiliation stützen.

Man sollte berücksichtigen, dass der Name Tippfehler enthalten kann, Vor- und Nachname vertauscht sein können. Die Affiliation kann sich beispielsweise so verändern: University of Munich Universität München, U. München, Munich Univ., usw.

Ein Auschlusskriterium für die Gleichheit ist, dass der Benutzer schon zuvor entschieden hat, dass zwei Objekte nicht gleich sind (ein entsprechender Eintrag ist im Datenmodell vorzusehen).

Es soll auch eine Funktion beschrieben werden, die zwei Objekte zu einem macht.

Abgabe bis 13.12.

Es muss pro Gruppe nur *eine* Lösung abgegeben werden.
Die Abgabe der Übungsblätter erfolgt mit dem CVS System. Dazu legen Sie die abzugebenden Dateien im CVS ab und markieren die abzugebende Version der Dateien mit "`LoesungsBlatt`<*nr*>" wie in Aufgabe 1 beschrieben. Die Lösungen sollten vorzugsweise im dazugehörigen Unterverzeichnis `uebeungsblaetter/nr/` vorzufinden sein, zumindest aber ein Hinweis auf den Ort der Lösungen.
Einige Aufgaben verlangen eine schriftliche Bearbeitung, diese ist dann je nach Komplexität als ASCII, html, ps- oder pdf-Dokument abzugeben. Auf *keinen* Fall im MS Word `doc`-Format.

Materialien

`http://www.uni-koblenz.de/~beckert/Lehre/Praktikum-Formale-Entwicklung/`

Bernhard Beckert: Zi. MB 218, Tel. 287-2775, Email: `beckert@uni-koblenz.de`
Vladimir Klebanov: Zi. MB 224, Tel. 287-2781, Email: `vladimir@uni-koblenz.de`