



Softwarepraktikum im Grundstudium im Wintersemester 2002/2003

2. Übung, 24. Oktober 2002

In dieser Übung soll eine erste Version der webgestützten Aufgabenverwaltung erstellt werden.

Hinweis:

Die Testate finden Donnerstag von 10.00 Uhr bis 11.45 Uhr statt. Wir erwarten bei allen Gruppen das Vorliegen ihrer Lösung zum **Beginn** des Testats.

1. Aufgabe (Java API-Dokumentation)

Schlagen Sie in der Java-API-Dokumentation die Verwendung des Interfaces `List` und der Klasse `ArrayList` nach. In welcher Beziehung stehen das genannte Interface und die Klasse? Wie werden Elemente in eine Liste (`List`) eingefügt, wie kann man auf den Inhalt einer Liste mit einem `Iterator` zugreifen? Wie kann man in einer Schleife den Inhalt eines `Iterators` durchlaufen?

2. Aufgabe (Klasse Task)

Erstellen Sie eine Java-Klasse `Task`, die eine Aufgabe repräsentiert. Eine Aufgabe besteht aus einem eindeutigen Bezeichner (vom Typ `Long`), einem Namen, einer Beschreibung, einem Anfangsdatum und einem Enddatum. Für die Daten soll der Datentyp `Date` verwendet werden. Alle Instanzvariablen sind `private` und können nur über `get` und `set`-Operationen zugegriffen werden; implementieren Sie diese. Neben diesen Operationen sollen die folgenden Operationen erstellt werden:

- Ein Konstruktor, der Werte für alle Eigenschaften übergeben bekommt und sie entsprechen initialisiert.
- `getStartDateAsString()`: Liefert das Datum als formatierte Zeichenkette. Nutzen Sie hierzu die Klasse `DateFormat`.
- `getEndDateAsString()`: analog
- `setStartDate(String datum)`: Nutzt die Klasse `DateFormat`, um den übergebenen `String` zu parsen, und setzt damit das Anfangsdatum. Die von den Methoden von `DateFormat` ausgelösten Ausnahmen sollen über `throws` in der Methodendeklaration weitergereicht werden.
- `setEndDate(String datum)`: analog
- `toString()`: gibt eine Aufgabe mit allen Daten als Zeichenkette zurück.

3. Aufgabe (Klasse Tasks)

Diese Klasse soll eine Menge von Aufgaben speichern. Hierzu soll Ihre Klasse eine Instanzvariable vom Typ `ArrayList` enthalten und folgende Methoden anbieten:

- Einen Konstruktor, der die `ArrayList` initialisiert.
- Eine Funktion `Iterator getIterator()`, die einen `Iterator` mit allen Aufgaben liefert.
- Eine Funktion `addNewTask(Task task)`, die eine neue Aufgabe in die `ArrayList` einträgt.

4. Aufgabe (Test)

Schreiben Sie ein Kommandozeilenprogramm, das eine Instanz der Klasse `Tasks` erzeugt, zwei `Task`-Objekte füllt und in dieser Instanz abspeichert. Dann soll in einer `while`-Schleife der Inhalt des Iterators über alle Aufgaben durchlaufen und ausgegeben werden.

5. Aufgabe (Aufgabenliste)

a) `index.html`

Erstellen Sie eine HTML-Datei `index.html`, die unter der Überschrift „Menü“ einen Link auf das Servlet `ShowTasksPrepare` enthält.

b) `ShowTasksPrepare.java`

Entwickeln Sie das Servlet `ShowTasksPrepare`. Dieses Servlet enthält die Methoden `init()` und `doGet()`. In `init()` soll überprüft werden, ob im `ServletContext` als Attribut mit dem Namen „Tasks“ eine Instanz der Klasse `Tasks` abgespeichert ist. Wenn nicht, soll diese Instanz erzeugt und dem `ServletContext` hinzugefügt werden.

Die Funktion `doGet()` beantwortet die Anfragen aus dem Link in `index.html`. Diese Funktion holt das Attribut „Tasks“ aus dem `ServletContext` und erhält über einen Aufruf der Funktion `getIterator()` den Iterator mit allen Aufgaben. Dieser Iterator soll als Attribut „Tasks“ dem aktuellen `request` hinzugefügt werden. Schließlich wird die Anfrage an die Java Server Page `ShowTasksView.jsp` weitergeleitet.

c) `ShowTasksView.jsp`

Diese Java Server Page stellt die Liste der Aufgaben in einer Tabelle dar. Dazu wird in einer `while`-Schleife der im `request` übergebene Iterator durchlaufen und für jede enthaltene Aufgabe die entsprechende Tabellenzeile ausgegeben. Unter der Aufgabenliste sollen Links zu `index.html` und `NewTaskView.html` stehen. `NewTaskView.html` wird ein Formular zum Hinzufügen von neuen Aufgaben enthalten.

6. Aufgabe (Aufgaben hinzufügen)

a) `NewTaskView.html`

Diese HTML-Datei soll ein Formular enthalten, in dem alle Daten (außer dem eindeutigen Bezeichner, vgl. Aufgabe 3) für eine Aufgabe erfasst werden können. Ordnen sie die Eingabefelder und die zugehörige Beschriftung in einer Tabelle an, damit diese untereinander stehen. Der Submit-Knopf des Formulars soll mit dem Servlet `NewTaskExecute` verbunden werden (Methode `POST`).

b) `NewTaskExecute.java`

Implementieren Sie in diesem Servlet die Methode `doPost()`. In ihr soll ein neues Objekt vom Typ `Task` erzeugt und mit den Daten aus dem HTML-Formular gefüllt werden. Diese Aufgabe soll dann der im `ServletContext` enthaltenen Instanz der Klasse `Tasks` hinzugefügt werden. Dann soll mit dem Aufruf `response.sendRedirect("ShowTasksPrepare");` die Anfrage auf die Aufgabenliste umgeleitet werden. Im Gegensatz zum Weiterleiten in Aufgabe 5 b) wird hierbei die neue URL im Browser angezeigt.

Testate

Aufgaben 1-4: Donnerstag, 31.10.2002 von 10.00 - 11.45 Uhr

Aufgabe 5 und 6: Donnerstag, 7.11.2002 von 10.00 - 11.45 Uhr