Praktische Informatik in der Wirtschaft
Westfälische Wilhelms-Universität Münster
Prof. Dr. Herbert Kuchen 5. Dezember 2016

---

8. Übung zu Informatik I im WS 16/17
Abgabe: 12. Dezember 2016, 12 $\frac{00}{}$ Uhr.

Aufgabe 20 - Gruppenabgabe (16 Punkte)

Santa braucht Ihre Hilfe! Die Wünschedatenbank, die jährlich zur Verwaltung der Wünsche aller Kinder verwendet wird, ist inzwischen ziemlich in die Jahre gekommen. Die Anwendung erlaubt das Eintragen von Wünschen für jedes einzelne Kind. Außerdem verfügt es über eine Suche, mit welcher nach den Wünschen eines bestimmten Kindes gesucht werden kann. Aber das Eintragen aller einzelnen Wünsche ist sehr aufwändig und auch unter Verwendung von Santas Fluxkompensator kaum zu bewältigen. Daher wünscht sich Santa nichts sehnlicher als die Möglichkeit eines Dateiimports in die Datenbank. Das Interface dafür ist bereits vor Jahren geschaffen worden, wurde aber nie vervollständigt. Und immerhin kommen die Wünsche eh jedes Jahr in Dateiform zu ihm ins Haus geflattert.

Helfen Sie Santa! Laden Sie sich das Projekt aus dem Learnweb herunter. Starten Sie es mit Hilfe der `main` Methode der Klasse `Wuenschedatenbank`. Analysieren Sie den fehlenen Code in der Klasse `DateiHochladenFrame` und implementieren Sie die fehlende Funktionalität. Das `TextField dateiName` soll den voll qualifizierten Pfad zu einer Datei erhalten (bspw. C\:Wunschzettel.txt). Dieser kann und soll direkt und unverändert an die Klasse zum Lesen der Datei übergeben werden! Die zu lesende Datei enthält pro Zeile den Wunsch eines Kindes. Jeder Wunsch hat insgesamt drei Ausprägungen, welche jeweils mit einem ";"getrennt sind. Dabei ist der Teil vor dem ersten ";"der Name des Kindes. Der zweite Teil ist der Wunsch des Kindes und der letzte Teil ist der Typ des Geschenks (Klein, Mittel oder Groß). Dabei soll es möglich sein, Kinder auch doppelt zu führen. Eine Datei könnte beispielsweise so aussehen:

```
Jochen;Ein Malbuch;Klein
Martin;Den Weltfrieden;Gross
Manuela;Einen kleinen Bruder;Mittel
Ricarda;Batman Aktionfigur;Klein
Annemarie;Neue Schuhe;Klein
Jochen;Eine Schaukel;Mittel
```

Verwenden Sie für das tatsächliche Hochladen und Parsen der Datei die vorgegebene Methode `public void dateiHochladen(String datei)`. Verwenden Sie zur Speicherung der Daten aus der Datei dieselben Strukturen, die auch für das manuelle Anlegen von Wünschen (siehe `NeuerEintragFrame`) verwendet werden. Verwenden Sie das Label `warnung`, um den Nutzer über Fehler in der Ausführung des Hochlade-Vorgangs zu informieren. Die Aktualisierung der Warnmeldung soll dabei nicht innerhalb der Methode `dateiHochladen` stattfinden, sondern gebündelt innnerhalb der Methode `actionPerformed()`. Werfen Sie dabei in der Methode `dateiHochladen` geeignete Exceptions, welche Sie in der Methode `actionPerformed()` auffangen und für einer Aktualisierung der Warnmeldung verwenden. Falls notwendig können Sie auf die vorhandenen Exceptions innerhalb des Projekts zurückgreifen. Werfen Sie auf diesem Weg nur Exceptions, welche tatsächlich einen Fehler beim Dateiupload signalisieren. Dabei sollte es anhand der Warnmeldung möglich sein, die Art des Fehlers zu identifizieren. Andere Exceptions, welche nicht einen fehlerhaften Dateiupload signalisieren, sollen Sie bereits in der Methode `dateiHochladen` behandeln!

Hinweis: Zum Einlesen einer Datei kann die Klasse `BufferedReader` verwendet werden. Deren Konstruktor nimmt z.B. einen `InputStreamReader` entgegen, dessen Konstruktor wiederum, wie der `ObjectInputStream`, einen `FileInputStream` entgegen nimmt. Über `readLine()` des `BufferedReader` kann die nächste Zeile aus der Datei gelesen werden. Am Dateiende wird eine `EOFException` geworfen, welche keinen Fehlerzustand signalisiert, sondern als Abbruchkriterium verwendet werden kann.

Hinweis: Die Klasse String stellt verschiedene nützliche Methoden zur Verfügung. Beispiele hierfür sind `split(String trennzeichen)`, welche einen String anhand eines Trennzeichens in mehrere Teile zerteilt, aber auch `substring(int a)`, welche es erlaubt einen Teilstring mit a Zeichen von einem String zu erzeugen!

Aufgabe 21 - Gruppenabgabe (8 + 6 = 14 Punkte)

Auf Vorlesungsfolie 132 wurde eine Funktion `map` zur Verwendung von λ-Ausdrücken definiert. Diese verwendet als Parameter ein Array sowie einen λ-Ausdruck. `map` wendet den λ-Ausdruck auf jedes einzelne Element des Arrays an und gibt diese Werte in einem neuen Array zurück.

a) Implementieren Sie eine statische Methode `zip` innerhalb einer Klasse `Zip`. Diese besitzt 3 Parameter. Die ersten beiden sind Arrays von Typ int[]. Der letzte ist ein λ-Ausdruck, welcher zwei Integer übergeben bekommt und einen zurückgibt. `zip` wendet den λ-Ausdruck sukzessive auf jeweils zwei Einträge aus den beiden Arrays an und schreibt das Ergebnis in ein neues Array vom Typ int[]. Falls die beiden Arrays unterschiedliche Länge haben, ist die Länge des kleineren Arrays ausschlaggebend. Überschüssige Einträge des größeren Arrays werden ignoriert und gehen nicht in das Ergebnis mit ein. In nachfolgendem Beispiel ist das Ergebnis eines zip-Aufrufs mit dem λ-Ausdruck $(a, b) \rightarrow a + b$ und den zwei Arrays a und b

aufgeführt:

$$\underset{\begin{bmatrix}1\\2\\3\end{bmatrix}}{\text{a}} \quad \underset{\begin{bmatrix}3\\5\\2\\7\end{bmatrix}}{\text{b}} \quad \rightsquigarrow \quad \underset{\begin{bmatrix}4\\7\\5\end{bmatrix}}{\text{Ergebnis}}$$

b) Schreiben Sie JUnit-Tests für mindestens 3 unterschiedliche λ-Ausdrücke. Verwenden Sie dazu die Datei ZipSuite.java. Analysieren Sie dabei Ihren Code und wählen Sie Ihre Array-Paare so, dass Sie möglichst viele unterschiedliche Fälle abdecken. Gehen Sie dabei insbesondere die Behandlung von unterschiedlichen Längen der übergebenen Arrays ein.

Anmerkung: *Versehen Sie ihre Abgaben bitte mit* ***Namen***, ***Matrikelnummern***, ***E-Mail-Adressen*** *und* ***Studiengängen*** *der beteiligten Bearbeiter/innen und laden Sie diese in der entsprechenden Aktivität im Learnweb hoch. Aufgaben, die mit dem Hinweis* ***Gruppenabgabe*** *versehen sind, dürfen mit maximal 3 Bearbeiter/innen gelöst werden. Aufgaben mit dem Hinweis* ***Einzelabgabe*** *müssen von jedem Studenten und jeder Studentin eigenständig gelöst und abgegeben werden. Viele der Aufgaben werden über das* ***EASy*** *(E-Assessment System) System im Learnweb eingereicht und automatisch vorausgewertet. Um Probleme bei der Abgabe wegen Überlastung des EASy-Servers zu vermeiden, würde ich Sie bitten, eine Abgabe auf den letzten Drücker nach Möglichkeit zu vermeiden. Bei Fragen zum Übungsbetrieb wenden Sie sich bitte an ihren/ihre Tutor/in oder an Tobias Reischmann.*