Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. Dr. François Bry

Programmierung und Modellierung, SoSe 16
Übungsblatt 12

Abgabe: bis Mo 04.07.2016 10:00 Uhr

Besprechung: ab Di 05.07.2016

Die Semestralklausur findet am Dienstag, 26. Juli 2016 statt.
Bitte die Hinweise auf der Webseite der Vorlesung beachten.

Aufgabe 12-1 Funktionale Fehlerbehandlung und Module

a) Erweitern Sie die Funktionen des Bankablaufes vom Übungsblatt 11 Aufgabe 1 um eine funktionale Fehlerbehandlung mit `Either`. Folgende Fehlerfälle sollen abgefangen werden:

- Einzahlungen sollen nur positive Integer Werte haben.
- Auszahlungen sollen nur negative Integer Werte haben.
- Der Bankkunde darf maximal 1000 € auf einmal abheben.
- Jeder Bankkunde hat einen Dispokredit von 1000 €, der nicht überschritten werden darf.

Definieren Sie eventuell Hilfsfunktionen, um diese Funktionalitäten umzusetzen und geben Sie im Fehlerfall *sinnvolle* Meldungen aus.

b) Definieren Sie für Ihre Funktionen ein Modul Bank. Es sollen nur die Funktionen `deposit`, `withdraw` und `accountState` exportiert werden.

Aufgabe 12-2 Fehlerbehandlung mit Exceptions

Exceptions bieten sich in Haskell an um Ausnahmen in einem nicht rein-funktionalen Kontext abzufangen und zu behandeln, wie beispielsweise bei `IO`. In der folgenden Aufgabe sollen Sie eine Anwendung implementieren, die eine Textdatei einliest und dem Benutzer danach einzelne Zeilen der Textdatei per Index abfragen lässt.

a) Implementieren Sie eine Funktion `readUserFile :: IO String`. Diese Funktion soll den Benutzer nach einem Dateinamen fragen und danach versuchen diese Datei einzulesen. Wenn die Datei nicht existiert, soll der Benutzer nach einem neuen Dateinamen gefragt werden. Ansonsten sollen der Inhalt der Datei als `IO String` zurückgegeben werden.

b) Machen Sie sich mit der Funktion `reads` vertraut. Schreiben Sie darauf aufbauend eine Funktion `readMaybe :: String -> Maybe Int`, welche einen `String` in den entsprechenden Integer-Wert umwandelt. Wenn es sich beim Argument nicht um einen Integer-Wert handelt, soll `Nothing` zurückgegeben werden.

c) Implementieren Sie eine Funktion `returnIndex :: [String] -> IO ()`, welche den Benutzer nach einem Index fragt und daraufhin das entsprechende Element der übergebenen Liste zurückgibt. Falls der Benutzer etwas anderes als einen Integer-Wert eingeben sollte, soll dies abgefangen werden, eine Fehlermeldung ausgegeben werden und nach einer neuen Eingabe gefragt werden.

d) Implementieren Sie aufbauend auf Ihren Ergebnissen aus den Teilaufgaben a) und b) eine Funktion `main :: IO ()`, die das in der Aufgabenstellung beschriebene Programm implementiert.

e) Ergänzen Sie Ihre Lösung aus Teilaufgabe d), so dass überprüft wird, ob der eingegebene Index in der Liste existiert (d.h., ob der eingegebene Index größer als die Länge der Liste ist). Im Fehlerfall soll eine entsprechende Meldung ausgegeben werden und der Benutzer um eine erneute Eingabe gebeten werden.