



Technische Universität Berlin

Software and Embedded Systems Engineering Group

Prof. Dr. Sabine Glesner

www.sese.tu-berlin.de Sekr. TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin





Softwaretechnik und Programmierparadigmen WiSe 2023/2024



Prof. Dr. Sabine Glesner
Simon Schwan
Julian Klein

Übungsblatt 05



Aufgabe 1: \$ and do

- a) Was macht der \$ Operator: $(\$) :: (a \rightarrow b) \rightarrow a \rightarrow b$? 
- b) Erarbeitet ein Programm, in dem ein Name als Eingabe gefordert wird und anschließend **Hello** {name} auf der Konsole ausgegeben wird mit der **do** Notation. 





Aufgabe 2: Module und Importe

- a) Fasst alle in den letzten Tutorien entwickelten Teile unseres Taschenrechners als Stack-Modul mit dem Namen *Command* zusammen, in dem alles exportiert wird. 
- b) Nun sollen nach außen hin nicht alle Teile des Moduls sichtbar sein: Insbesondere **fee** geht niemanden etwas an. Außerdem dürfen **Commands** keine negativen Werte enthalten. Versteckt daher auch den Konstruktor **ValR** und stellt dafür eine Funktion **toCommand** `:: Int -> Command` zur Verfügung, die nur für positive Integers definiert ist. Wurde eine solche Funktion eigentlich schon implementiert? 

Schlüssel:



-  Ein ergänzendes Video wird zur Vor- oder Nachbereitung veröffentlicht.
-  Wird im Tutorium besprochen.

Aufgabe 3: Testen in Haskell

- a) Was sind Unit Tests? 
- b) Implementiert Unit-Tests für die Funktionen `toCommand` und `eval` mithilfe der *Hspec*-Bibliothek. Die Tests sollen mit dem Befehl `stack test` ausgewertet werden. 
- c) Teilt die Testfälle zur Übersicht in verschiedene Test-Spezifikationen auf. 
- d) Wurden beide Funktionen ausreichend getestet? Überprüft eure Testfälle mithilfe der *Haskell Program Coverage* (HPC) und erweitert ggf. eure Tests, um beide Funktionen vollständig zu überprüfen. 
Hinweis: Nutzt die Befehle `stack clean` und `stack test -coverage`.

Aufgabe 4: Gitlab

In dieser Aufgabe soll ein Gitlab-Projekt auf der Gitlab-Instanz der TU für unser Taschenrechner angelegt werden. Gitlab ist ein sehr verbreitetes Versionskontrollsystem. Die TU bietet einen eigenen Gitlab Service, den wir auch in der Hausaufgabe nutzen werden. Diese Aufgabe soll deshalb als kleine Einführung in die Basis git-Befehle dienen.

- a) Legt zunächst ein Gitlab-Projekt für den Taschenrechner an. Nutzt dafür die folgenden Befehle: `git init`, `git add`, `git commit`, `git remote` und `git push`. 
- b) Führt nun lokale Änderungen durch und ladet diese anschließend auch auf Gitlab hoch. Was müsst ihr tun, wenn ihr neue Dateien hinzufügen wollt? 
- c) Erzeugt nun einen neuen Branch und führt einen Merge durch. Beuntzt dafür `git merge`. 