## Funktionale Programmierung

# Studienarbeit

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 09.10.2018 06:54

Sie müssen zum Bestehen dieser Lehrveranstaltung eine Studienarbeit erstellen. Diese Studienarbeit wird benotet.

## Rahmenbedingungen

- Die Studienarbeit kann alleine oder in einer Zweiergruppe bearbeiten und eingereicht werden.
- Über den GitHub Classroom Link können Sie ein Team erstellen oder einem bereits erstellten Team beitreten. Das Team bekommt ein gemeinsames, leeres Repository. Auch wenn Sie alleine arbeiten, müssen Sie ein Team erstellen.

## Anforderungen

Gegenstand der Studienarbeit ist ein Haskell-Projekt. Dabei muss folgendes umgesetzt werden:

1. Die Lösung besteht im Wesentlichen aus Haskell Code. Wenn Sie beispielsweise eine Webanwendung in Haskell schreiben, darf die Logik nicht in Javascript umgesetzt sein.
2. Die Lösung ist ein Haskell-Projekt, das mit Stack gebaut werden kann.
3. Die Top-Level-Funktionen und die Module sind vollständig mit Haddock dokumentiert.

4. Der wesentliche Teil der Funktionen ist durch Tests, die sich mit Stack ausführen lassen, getestet.
5. Der Build und die Tests werden mit einem Travis-Job ausgeführt, der bei jedem Push auf GitHub getriggert wird. Eine Beschreibung Ihres Projektes und die Haddock-Dokumentation etc. wird auf einer GitHub-Page zur Verfügung gestellt.
6. Im gemeinsamen Wiki ist Ihr Repository und Ihre Website verlinkt.
7. Aus der Datei `README.md`, die top-level im Repository liegt, geht hervor, wie Ihre Anwendung zu bauen und zu nutzen ist.
8. Zusätzlich für Zweierteams: Aus der Datei `README.md` geht hervor, wer für welche Teile des Projektes **verantwortlich** ist. Sie bekommen individuelle Noten für Ihren jeweiligen Anteil.
9. Sie gehen nach dem Git Flow vor.
10. Zur Vorstellung des Zwischenstandes releasen Sie eine *Prerelease*-Version.
11. Zur Abschlusspräsentation releasen Sie Ihre Version `1`.
12. Sie können bis zum endgültigen Abgabetermin eine weitere Version releasen. Das letzte Release **vor** dem Abgabetermin wird bewertet. Denken Sie insbesondere daran auch bei Bugfixes oder ähnlichem nach einem Release ein weiteres Release zu erzeugen.

# Termine

### ab sofort

Themenfindung / Teambildung

Nutzen Sie gerne die Praktikumstermine um mit mir Ideen durchzusprechen.

### 29.10.2018 (spätestens nach dem Praktikum)

Ihr Thema steht fest, ist mit mir besprochen und im Wiki eingetragen. Im Wiki haben Sie Schreibrechte!

### 26.11.2018

Kurze Vorstellung des Zwischenstandes (== Prerelease) im Rahmen der Lehrveranstaltung.

### 14.01.2019

Abschlusspräsentation der Studienarbeit (== Release 1) im Rahmen der Lehrveranstaltung.

### 28.01.2019, 08:00 Uhr

Deadline für endgültige Abgabe. Das letzte Release **vor** dem 28.01.2019, 08:00 Uhr ist automatisch die Abgabe.

## Support

In den Praktikumsterminen stehe ich Ihnen zur Verfügung. Darüber hinaus steht Ihnen der Gitter-Channel jederzeit für den Austausch mit mir und den anderen Teilnehmern der Lehrveranstaltung zur Verfügung. Wenn Sie konkrete Fragen zu Ihrem Code haben, ist es am Besten, wenn Sie den Code auf GitHub pushen und ein Issue in Ihrem Repository erzeugen und, mit dem Label `help wanted` versehen, mir zuweisen. Mehr Informationen finden Sie auch unter `https://ob.cs.hm.edu/exercises.html`.

## Beispiele für Studienarbeiten aus den letzten Jahren

- Turingmaschine mit Curses-Interface
- Minesweeper mit OpenGL-GUI
- Simple CLI-Filesharing through DNS-SD
- Vier gewinnt mit GUI
- Fractal-Generator mit GTK-GUI
- Dependency Parser für C++
- Verwaltungssystem Campus IT (Threepenny-GUI mit Datenbankanbindung)
- Hangman
- Finanzverwaltungssystem für Verein
- Schiffeversenken-P2P (Peer2Peer)
- LAN-Chat mit Autodiscovery

bzw. unter `https://github.com/ob-fun-ws17/ob-fun-ws17/wiki`.

Anmerkung: Die Anforderungen bei den letzten Malen, waren zum Teil etwas anders, beispielsweise war nicht verlangt nach dem Git Flow vorzugehen.