



Spezielle Kapitel der Praktischen Informatik: Compilerbau, INF3199A

Dokumentation

für die Prüfung zum
Master / Bachelor of Science

des Studiengangs Informatik
an der Universität Tübingen

von

**Arwed Mett, Steffen Lindner, Robin Heinz, Pavel Karasik,
Florian Engel**

Matrikelnummer

4170745, 3912506, 3971071, 3991213, 3860700

Semester

WS17/18

Betreuer

Prof. Dr. Martin Plümicke

Gutachter

Prof. Dr. Martin Plümicke

Inhaltsverzeichnis

1	Einleitung	1
2	Installation	2
2.1	Voraussetzungen	2
2.2	Erstellungsprozess	2
2.3	Bedienung	3
3	Organisation	4
3.1	Teammitglieder	4
3.2	Aufgabenverteilung	4
	Abkürzungsverzeichnis	5
	Literatur	6

1 Einleitung

Im folgenden werden die Ergebnisse des Compilerbau Projektes der Vorlesung „Spezielle Kapitel der Praktischen Informatik: Compilerbau, INF3199A“ dokumentiert. Ziel war es einen Java Compiler zu entwickeln der eine Teilmenge der Programmiersprache Java [3] zu Java Bytecode[2] compiliert. Der Bytecode kann anschließend mit einer Java Virtual Machine (JVM)[1] ausgeführt werden. Des weiteren wurde ein Testframework entwickelt um die Teilfunktionen des Compilers zu validieren.

Das Projekt wurde innerhalb des Wintersemesters 17/18 an der Universität Tübingen in einer Gruppe von 5 Leuten implementiert. Hierfür wurde das Projekt in Parser/Lexer, Typchecker, Codegenerierer und Tester unterteilt.

2 Installation

Im folgenden werden die Schritte zur Installation des Compilers beschrieben.

2.1 Voraussetzungen

Es werden folgende Programme benötigt.

- GHC \geq v8.0.1
- Cabal \geq v1.24.0
- JRE \geq 1.8
- Git \geq 2.14.3

2.2 Erstellungsprozess

Um den Java Compiler zu bauen muss zuerst das Repository heruntergeladen werden. Hierzu kann git verwendet werden. Falls Git nicht auf dem Rechner installiert ist, kann das Repository alternativ auch über das WebUI von Github heruntergeladen werden.

```
1 git clone \  
2   -b 'v1.0.0' --single-branch --depth 1 \  
3   git@github.com:Pfeifenjoy/compilerbau-WS17-18.git
```

Listing 2.1: Download des Projektes

Dadurch wird das Repository in den Ordner „compilerbau-WS17-18“ geladen.

Anschließend muss in den Ordner des Quellcodes gewechselt werden. Dieser kann mit Hilfe von Cabal erstellt und installiert werden. Durch `cabal install` wird das Kommando `jc` installiert. Falls sich `jc` nicht im Pfad befindet kann es alternativ mit `cabal exec jc` ausgeführt werden.

```
1 cd compilerbau-WS17-18/project
2 cabal configure
3 cabal build
4 cabal install
```

Listing 2.2: Bauen des jc Kommandos

2.3 Bedienung

Das Programm `jc` bekommt als Argument eine Liste von Java Quelldateien, und generiert daraus Class-Files. Mithilfe des flag `-l LOG`, e.g. `cabal exec jc -- -l log *.java`, wird eine log-Datei erstellt, welche die abstrakte Syntax etc. beinhaltet. Eine genauere Beschreibung der Abstrakten Syntax ist unter `compilerbau-WS17-18/project/src/ABSTree.hs` zu finden.

Der Hilfetext aus [Listing 2.3](#) kann mittels `cabal exec jc -- -h` generiert werden.

```
1 usage : jc [source-files...] [-l LOG] [-v] [-h] [--version]
2
3 mandatory arguments:
4   source-files           Path to the sources which are
5                           compiled.
6
7 optional arguments:
8   -l, --log LOG          Specify log file.
9   -v, --verbose          Show extra information
10  -h, --help             show this help message and exit
11  --version              print the program version and exit
```

Listing 2.3: Hilfe jc

3 Organisation

3.1 Teammitglieder

3.2 Aufgabenverteilung

Abkürzungsverzeichnis

JVM Java Virtual Machine

Literatur

- [1] *Kostenloser Java-Download*. URL: <https://java.com/de/download/> (besucht am 16.02.2018).
- [2] Tim Lindholm u. a. *The Java Virtual Machine Specification, Java SE 8 Edition*. 1st. Addison-Wesley Professional, 2014. ISBN: 013390590X, 9780133905908.
- [3] Prof. Dr. Martin Plümicke. „Compilerbau Prüfungsleistung“. In: (2017). URL: http://www2.ba-horb.de/~pl/Compilerbau_WS2017_18/Compiler_Aufgabe.pdf.