

Projekt Java Compiler

Spezielle Kapitel der Praktischen Informatik: Compilerbau

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

05.02.2018

Universität Tübingen

Agenda

- Test-Framework
- Lexer
- Parser
- Fazit

Projekt Java Compiler

2018-02-06

Agenda

- Test-Framework
- Lexer
- Parser
- Fazit

Allgemein

Aufgabenstellung:
Entwickeln eines Mini-Java Compilers mit den zugehörigen Schritten: Lexer, Parser, TypChecker und Codegenerierung.

Projekt Java Compiler

2018-02-06

└ Allgemein

Allgemein

Aufgabenstellung:
Entwickeln eines Mini-Java Compilers mit den zugehörigen Schritten: Lexer, Parser, TypChecker und Codegenerierung.

Allgemein: Ziel

Korrekte Übersetzung der folgenden Klasse:

```
class Fibonacci {  
    int getFib(int n) {  
        return (n < 2) ? n : getFib(n-1) + getFib(n-2);  
    }  
}
```

Projekt Java Compiler

2018-02-06

└ Allgemein: Ziel

Allgemein: Ziel

Ziel

Korrekte Übersetzung der folgenden Klasse:

```
class Fibonacci {  
    int getFib(int n) {  
        return (n < 2) ? n : getFib(n-1) + getFib(n-2);  
    }  
}
```

Umgesezte Features (Auszug):

- Ternary Operator
- For / While / DoWhile
- If / If-Else / Switch-Case
- Pre- bzw. Post Inkrement/Dekrement
- Arithmetische Operatoren (+, -, /, div, mod, *) inklusive Zuweisung (+=, etc.)

2018-02-06

└ Featureliste

Featureliste

Umgesezte Features (Auszug):

- Ternary Operator
- For / While / DoWhile
- If / If-Else / Switch-Case
- Pre- bzw. Post Inkrement/Dekrement
- Arithmetische Operatoren (+, -, /, div, mod, *) inklusive Zuweisung (+=, etc.)

Entwicklung

Code-Sharing über GitHub (<https://github.com/Pfeifenjoy/compilerbau-WS17-18>) mit continuous integration (travis).

Als Build-System wird cabal eingesetzt.

Projekt Java Compiler

2018-02-06

└ Entwicklung

Code-Sharing über GitHub (<https://github.com/Pfeifenjoy/compilerbau-WS17-18>) mit continuous integration (travis).
Als Build-System wird cabal eingesetzt.

Projektmanagement

Jira Software - Kanban board

QUICK FILTERS: Only My Issues Recently Updated

Release: ...

Backlog	In Development	Testing	Done
Build a TUI - Terminal User Interface None COM-16	Generate methods Codeerzeugung COM-26	Parser Lexer COM-10	Lexer in Alex Lexer GOM-6
Class variable definition Semantische Analyse COM-20	State Monad Codeerzeugung COM-27	Implement assign Codeerzeugung COM-29	Abstrakte Syntax None GOM-12
Block scope for loop variables Semantische Analyse COM-21	Generate field variable Codeerzeugung COM-25	Implement MethodCall Codeerzeugung COM-30	Fix for loop Parser COM-13
Local vs Class variable Semantische Analyse COM-22	Implement VarDecl Codeerzeugung COM-31	CI Travis None COM-14	Protect Master None GOM-16
Check If Else, shift/reduce conflict Parser COM-23	Implement new Codeerzeugung COM-32	Switch-Case: Default Lexer, Parser COM-17	
	Implement SwitchCase Codeerzeugung COM-33		

7

Projekt Java Compiler

Projektmanagement

2018-02-06

Projektmanagement

Projekt Java Compiler

Boards

Issues

Reports

Confluence

Cloud

Help

Test-Framework

Projekt Java Compiler
└ Test-Framework

2018-02-06

Test-Framework

Test-Framework

Das Test-Framework wurde selbst implementiert. Es enthält diverse Funktionen zum automatisierten Überprüfen der Testfälle.

Tests werden in korrekte und falsche Testfälle unterteilt.

Projekt Java Compiler └ Test-Framework

└ Test-Framework

Test-Suite: Token-Coverage & Testfälle

Die Test-Suite umfasst eine Token-Coverage von 100%.

Zusätzlich umfasst die Test-Suite insgesamt 21 gültige und 12 ungültige Testfälle.

Ungültige Testfälle können in Syntaxfehler (Parser) und Typfehler (Typchecker) eingeteilt werden.

Projekt Java Compiler └ Test-Framework

└ Test-Suite: Token-Coverage & Testfälle

2018-02-06

Test-Suite: Token-Coverage & Testfälle
Die Test-Suite umfasst eine Token-Coverage von 100%.
Zusätzlich umfasst die Test-Suite insgesamt 21 gültige und 12 ungültige Testfälle.
Ungültige Testfälle können in Syntaxfehler (Parser) und Typfehler (Typchecker) eingeteilt werden.

Test-Suite: Testfälle

Jedes Testfile liegt in einem Ordner (Correct bzw. Wrong) mit zugehöriger .java-Datei.

Ein Testfile besteht aus:

- Erwarteten Tokens
- Erwarteter abstrakter Syntax
- Erwarteter getypter abstrakter Syntax

Zusätzlich zum eigentlichen Testfile enthält der Ordner ein ClassFile in Haskell, mit der zu erwartenden Struktur des erzeugten Classfiles.

10

Projekt Java Compiler └ Test-Framework

└ Test-Suite: Testfälle

2018-02-06

Test-Suite: Testfälle

Jedes Testfile liegt in einem Ordner (Correct bzw. Wrong) mit zugehöriger .java-Datei.
Ein Testfile besteht aus:

- Erwarteten Tokens
- Erwarteter abstrakter Syntax
- Erwarteter getypter abstrakter Syntax

Zusätzlich zum eigentlichen Testfile enthält der Ordner ein ClassFile in Haskell, mit der zu erwartenden Struktur des erzeugten Classfiles.

Test-Suite: Beispiel Testfile

```
module Correct.EmptyClass.Steps where

import ABSTree
import Lexer.Token

emptyTokens = [Lexer.Token.CLASS,
              Lexer.Token.IDENTIFIER "Test",
              Lexer.Token.LEFT_BRACE,
              Lexer.Token.RIGHT_BRACE
            ]

emptyABS = [Class "Test" [] []]
emptyTypedABS = [Class "Test" [] []]
```

Projekt Java Compiler └ Test-Framework

└ Test-Suite: Beispiel Testfile

2018-02-06

```
Test-Suite: Beispiel Testfile
module Correct.EmptyClass.Steps where
import ABSTree
import Lexer.Token

emptyTokens = [Lexer.Token.CLASS,
              Lexer.Token.IDENTIFIER "Test",
              Lexer.Token.LEFT_BRACE,
              Lexer.Token.RIGHT_BRACE
            ]

emptyABS = [Class "Test" [] []]
emptyTypedABS = [Class "Test" [] []]
```

Test-Suite: Beispielprogramme

Die Testsuite enthält neben den Testfällen auch eine Reihe von (realistischeren) Anwendungsprogrammen. Diese wurden mit 'normalen' Java programmen getestet.

- Multiplikation
- Gaußsumme (kleiner Gauß)
- Fakultät
- Fibonacci
- Potenz a^b
- $\lfloor \sqrt{x} \rfloor$

Projekt Java Compiler └ Test-Framework

└ Test-Suite: Beispielprogramme

2018-02-06

Test-Suite: Beispielprogramme

- Die Testsuite enthält neben den Testfällen auch eine Reihe von (realistischeren) Anwendungsprogrammen. Diese wurden mit 'normalen' Java programmen getestet.
- Multiplikation
 - Gaußsumme (kleiner Gauß)
 - Fakultät
 - Fibonacci
 - Potenz a^b
 - $\lfloor \sqrt{x} \rfloor$

Lexer

Projekt Java Compiler
└ Lexer

2018-02-06

Lexer

Tokens

```
data Token
```

```
    — Arithmentics
    = ADD
    | SUBTRACT
    | MULTIPLY
    | DIVIDE
    | MODULO
    | INCREMENT
    | DECREMENT
    — Logical
    | NOT
```

```
...
```

Projekt Java Compiler

- Lexer

- Tokens

2018-02-06

Tokens

```
data Token
    — Arithmentics
    = ADD
    | SUBTRACT
    | MULTIPLY
    | DIVIDE
    | MODULO
    | INCREMENT
    | DECREMENT
    — Logical
    | NOT
    ...
```

Struktur Alex File

```
%wrapper " basic"

$digit = 0-9
$alpha = [a-zA-Z]

tokens :-

— Ignore
$white+ ;
"//" .* ;

— Operators
— Arithmetics
\+          { \s -> ADD }
\-          { \s -> SUBTRACT }
\*          { \s -> MULTIPLY }
\/          { \s -> DIVIDE }

...
```

14

Projekt Java Compiler

- Lexer

- Struktur Alex File

Struktur Alex File

```
Wrapper " basic"
$digit = 0-9
$alpha = [a-zA-Z]
tokens :-
— Ignore
$white+ ;
"//" .* ;

— Operators
— Arithmetics
\+          { \s -> ADD }
\-          { \s -> SUBTRACT }
\*          { \s -> MULTIPLY }
\/          { \s -> DIVIDE }

...
```

2018-02-06

Parser

Projekt Java Compiler
└ Parser

2018-02-06

Parser

Parser - Operatoren Priorität

```
%right in
%right ASSIGN ADD ...
%right QUESTIONMARK COLON
%left OR
...
%nonassoc LESSER GREATER LESSER_EQUAL ...
...
%nonassoc INCREMENT DECREMENT
```

Projekt Java Compiler └ Parser

└ Parser - Operatoren Priorität

```
%right in
%right ASSIGN ADD ...
%right QUESTIONMARK COLON
%left OR
...
%nonassoc LESSER GREATER LESSER,EQUAL ...
...
%nonassoc INCREMENT DECREMENT
```

Struktur Happy File

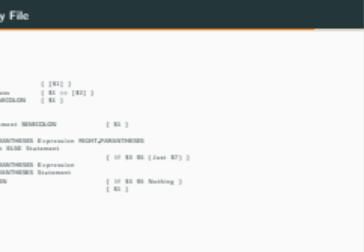
```
Program
: Class { [$1] }
| Program Class { $1 ++ [$2] }
| Program SEMICOLON { $1 }
```

```
Statement
: SingleStatement SEMICOLON { $1 }
...
| IF LEFT_PARANTHESES Expression RIGHT_PARANTHESES
  Statement ELSE Statement
    { If $3 $5 (Just $7) }
| IF LEFT_PARANTHESES Expression
  RIGHT_PARANTHESES Statement
    %prec THEN
      { If $3 $5 Nothing }
| Switch
  { $1 }
```

16

Projekt Java Compiler └ Parser

└ Struktur Happy File



Beispiel

```
class Simpleif {
    int i;
    void dolf() {
        int a;
        a = 5;
        i = 0;
        if(a < 5) {
            i = a;
        }
        else {
            i = 2;
        }
    }
}
```

[Class „Simpleif“]



Projekt Java Compiler └ Parser

2018-02-06

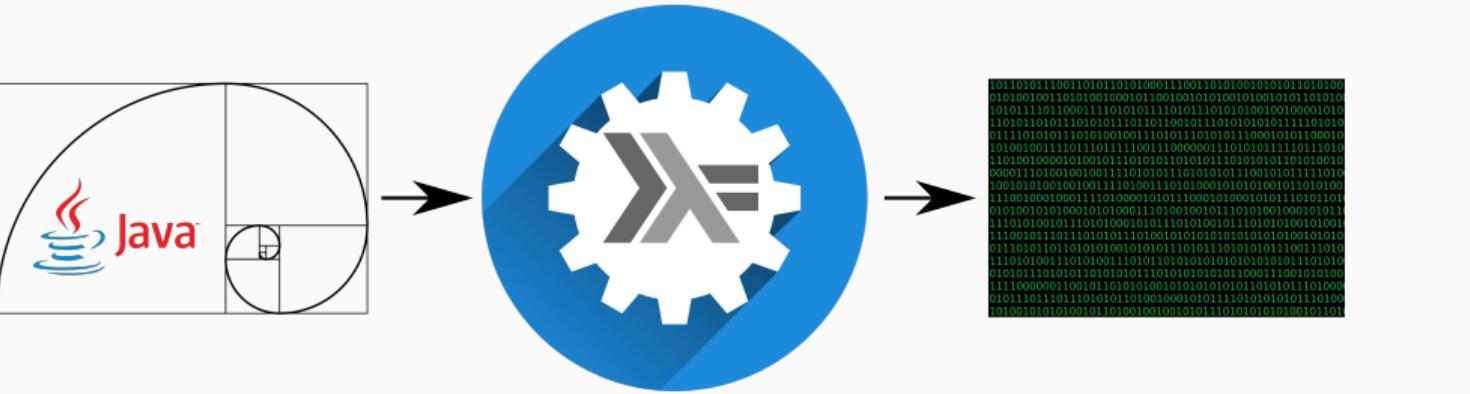
└ Beispiel

```
Beispiel
class Simpleif {
    int i;
    void dolf() {
        int a;
        a = 5;
        i = 0;
        if(a < 5) {
            i = a;
        }
        else {
            i = 2;
        }
    }
}

[Class „Simpleif“]
└
  / \
  FieldDecl FieldDecl
```

Fazit

Fazit



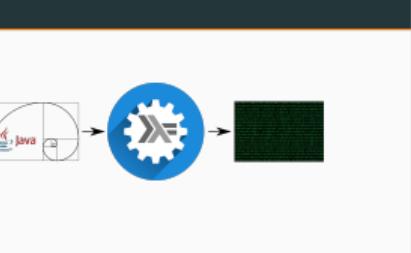
18

Projekt Java Compiler

- └ Fazit

- └ Fazit

2018-02-06



Projekt Java Compiler

Spezielle Kapitel der Praktischen Informatik: Compilerbau

Github: <https://github.com/Pfeifenjoy/compilerbau-WS17-18>

Florian Engel, florian.engel@student.uni-tuebingen.de

Robin Heinz, robin.heinz@student.uni-tuebingen.de

Pavel Karasik, pavel.karasik@student.uni-tuebingen.de

Steffen Lindner, steffen.lindner@student.uni-tuebingen.de

Arwed Mett, arwed.mett@student.uni-tuebingen.de

Projekt Java Compiler

└ Fazit

2018-02-06

Projekt Java Compiler
Spezielle Kapitel der Praktischen Informatik: Compilerbau
Github: <https://github.com/Pfeifenjoy/compilerbau-WS17-18>

Florian Engel, florian.engel@student.uni-tuebingen.de
Robin Heinz, robin.heinz@student.uni-tuebingen.de
Pavel Karasik, pavel.karasik@student.uni-tuebingen.de
Steffen Lindner, steffen.lindner@student.uni-tuebingen.de
Arwed Mett, arwed.mett@student.uni-tuebingen.de