

Projekt Java Compiler

Spezielle Kapitel der Praktischen Informatik: Compilerbau

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

05.02.2018

Universität Tübingen

Projekt Java Compiler

2018-02-07

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

Abstrakte Syntax

Test-Framework

Lexer

Parser

Fazit

Projekt Java Compiler

2018-02-07

└ Agenda

Agenda

- Abstrakte Syntax
- Test-Framework
- Lexer
- Parser
- Fazit

Allgemein

Aufgabenstellung:

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

Projekt Java Compiler

Allgemein

Lung-

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

Allgemein: Ziel

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

Ziel

Korrekte Übersetzung der folgenden Klasse:

Projekt Java Compiler

2018-02-07

└ 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);  
    }  
}
```

Featureliste

Umgesetzte Features (Auszug):

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

Projekt Java Compiler

2018-02-07

└ Featureliste

Featureliste

Umgesetzte Features (Auszug):

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

Als Build-System wird cabal eingesetzt.

2018-02-07

└ Entwicklung

Projektmanagement

Jira Software Kanban board titled "COM board". The board has four columns: Backlog, In Development, Testing, and Done. The Backlog column contains 6 items. The In Development column contains 7 items. The Testing column contains 1 item. The Done column contains 9 items.

Column	Items
Backlog	6
In Development	7
Testing	1
Done	9

Items in the Backlog:

- Build a TUI - Terminal User Interface (Status: Max 7, Assigned to COM-16)
- Class variable definition (Status: Max 7, Assigned to COM-20)
- Block scope for loop variables (Status: Max 7, Assigned to COM-21)
- Local vs Class variable (Status: Max 7, Assigned to COM-22)
- Check If Else, shift/reduce conflict (Status: Max 7, Assigned to COM-23)

Items in the In Development column:

- Generate methods (Status: Max 7, Assigned to COM-26)
- State Monad (Status: Max 2, Assigned to COM-27)
- Generate field variable (Status: Max 7, Assigned to COM-25)
- Implement assign (Status: Max 2, Assigned to COM-29)
- Implement MethodCall (Status: Max 7, Assigned to COM-30)
- Implement VarDecl (Status: Max 7, Assigned to COM-31)
- Implement new (Status: Max 7, Assigned to COM-32)
- Implement SwitchCase (Status: Max 7, Assigned to COM-33)

Item in the Testing column:

- Parser (Status: Max 1, Assigned to COM-10)

Items in the Done column:

- Lexer in Alex (Status: Max 1, Assigned to COM-6)
- Abstrakte Syntax (Status: Max 1, Assigned to COM-12)
- Fix for loop (Status: Max 1, Assigned to COM-18)
- CI Travis (Status: Max 1, Assigned to COM-14)
- Protect Master (Status: Max 1, Assigned to COM-16)
- Switch-Case: Default (Status: Max 1, Assigned to COM-17)

Projekt Java Compiler

2018-02-07

Projektmanagement

Project management interface showing multiple boards and dashboards. The top navigation bar includes "Dashboards", "Projects", "Issues", "Boards", and "Create". The main area displays several boards, likely Kanban boards, for different projects or components.

Abstrakte Syntax

ABSTree.hs

Test-Framework

Projekt Java Compiler
└ Test-Framework

2018-02-07

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

2018-02-07

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

Tests werden in korrekte und falsche Testfälle unterteilt.

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.

Projekt Java Compiler └ Test-Framework

└ Test-Suite: Testfälle

2018-02-07

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

2018-02-07

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

2018-02-07

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

2018-02-07

Lexer

Tokens

```
data Token
```

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

Projekt Java Compiler
└ Lexer

└ Tokens

2018-02-07

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 }

    ...
```

Projekt Java Compiler

└ Lexer

└ Struktur Alex File

15

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 }
    ...
```

Parser

Projekt Java Compiler
└ Parser

2018-02-07

Parser

Parser - Allgemein

Verwendete Werkzeuge:
Happy, Alex

- Verwendete Werkzeuge: Happy, Alex
- Input: Tokens
- Output: ABSTree
- Herausforderungen:
 - Integration in Build System
 - Operatoren Priorität
 - Klassen / Konstruktoren
 - If-Else

Projekt Java Compiler

└ Parser

└ Parser - Allgemein

2018-02-07

- Parser - Allgemein
- Verwendete Werkzeuge: Happy, Alex
 - Input: Tokens
 - Output: ABSTree
 - Herausforderungen:
 - Integration in Build System
 - Operatoren Priorität
 - Klassen / Konstruktoren
 - If-Else

Parser - Integration Build System

```
library
exposed-modules:      Lexer , ...
build-depends:        base <= 4.10.1 ...
hs-source-dirs:        src
build-tools:           alex, happy
other-modules:         Lexer.Lexer, Parser.Parser
default-language:      Haskell2010
```

Projekt Java Compiler └ Parser

2018-02-07

└ Parser - Integration Build System

```
library
exposed-modules:      Lexer , ...
build-depends:        base <= 4.10.1 ...
hs-source-dirs:        src
build-tools:           alex, happy
other-modules:         Lexer.Lexer, Parser.Parser
default-language:      Haskell2010
```

- build-tools
- other-modules

Parser - Operatoren Priorität

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

Projekt Java Compiler └ Parser

2018-02-07

└ Parser - Operatoren Priorität

- Was ist Operatoren Priorität
- Wie wird es in Happy implementiert
- Precedence top (low), bottom (high)
- Beschreibe Assoziativität

Parser - Operatoren Priorität
%right in // lowest precedence
%right ASSIGN ADD ...
%right QUESTIONMARK COLON
%left OR
...
%nonassoc LESSER GREATER LESSER_EQUAL ...
...
%nonassoc INCREMENT DECREMENT // highest precedence

Parser - 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 }
```

Projekt Java Compiler └ Parser

2018-02-07

└ Parser - Struktur Happy File

Parser - 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 }
```

Parser - Klassen / Konstruktoren

Problem:

- Methoden, Attribute aufteilen in Listen
- teste: Klassename = Konstruktorname

Projekt Java Compiler └ Parser

└ Parser - Klassen / Konstruktoren

2018-02-07

Parser - Klassen / Konstruktoren
Problem:
• Methoden, Attribute aufteilen in Listen
• teste: Klassename = Konstruktorname

Parser - Klassen / Konstruktoren

```
ClassBody
: FieldDecl      { ClassBodyDecl [$1] [] [] }
| MethodDecl     { ClassBodyDecl [] [$1] [] }
| ConstructorDecl { ClassBodyDecl [] [] [$1] }
| ClassBody FieldDecl { ClassBodyDecl ((fields $1) ++ [$2]) (...) (...) }
| ClassBody MethodDecl { ClassBodyDecl (...) ((methods $1) ++ [$2]) (...) }
| ClassBody ConstructorDecl { ClassBodyDecl (...) (...) ((constructors $1) ++ [$2]) }
```

Problem:

- Methoden, Attribute aufteilen in Listen
- teste: Klassename = Konstruktorname

Projekt Java Compiler └ Parser

2018-02-07

└ Parser - Klassen / Konstruktoren

Parser - Klassen / Konstruktoren

Problem:

- Methoden, Attribute aufteilen in Listen
- teste: Klassename = Konstruktorname

```
ClassBody
: FieldDecl      { ClassBodyDecl [$1] [] [] }
| MethodDecl     { ClassBodyDecl [] [$1] [] }
| ConstructorDecl { ClassBodyDecl [] [] [$1] }
| ClassBody FieldDecl { ClassBodyDecl ((fields $1) ++ [$2]) (...) (...) }
| ClassBody MethodDecl { ClassBodyDecl (...) ((methods $1) ++ [$2]) (...) }
| ClassBody ConstructorDecl { ClassBodyDecl (...) (...) ((constructors $1) ++ [$2]) }
```

Parser - If - Else

Problem:

```
if_stmt
: "if" expr "{" stmt "}"
| "if" expr "{" stmt "}" "else" "{" statement "}"
```

Projekt Java Compiler └ Parser

2018-02-07

└ Parser - If - Else

Parser - If - Else

Problem:

```
if_stmt
: "if" expr "{" stmt "}"
| "if" expr "{" stmt "}" "else" "{" statement "}"
```

Parser - If - Else

Problem:

```
if_stmt
: "if" expr "{" stmt "}"
| "if" expr "{" stmt "}" "else" "{" statement "}"
```

Solution:

```
%nonassoc THEN
```

```
if_stmt
: "if" expr "{" stmt "}" %prec THEN
| "if" expr "{" stmt "}" "else" "{" statement "}"
```

Projekt Java Compiler └ Parser

2018-02-07

└ Parser - If - Else

Parser - If - Else

Problem:

```
if_stmt
: "if" expr "(" stmt ")"
| "if" expr "(" stmt ")" "else" "(" statement ")"
```

Solution:

```
%nonassoc THEN
```

```
if_stmt
: "if" expr "(" stmt ")" %prec THEN
| "if" expr "(" stmt ")" "else" "(" statement ")"
```

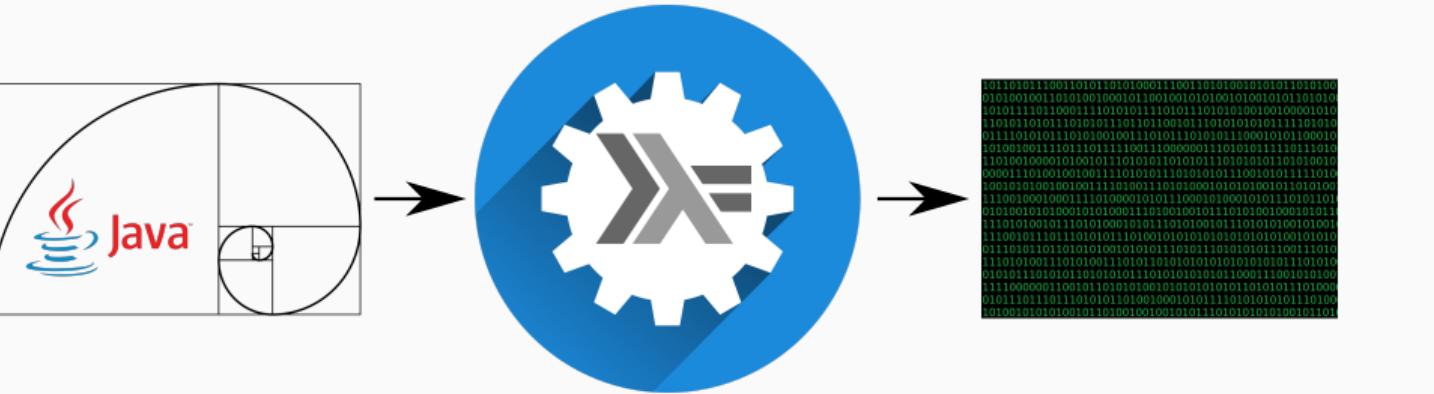
Demo

2018-02-07

Fazit

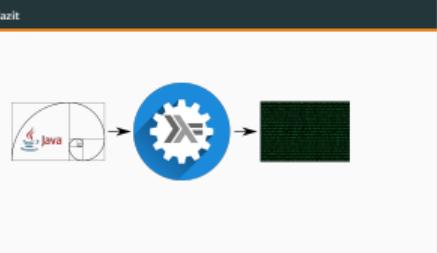
Fazit

Fazit



Projekt Java Compiler

Fazit



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

2018-02-07

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