

Projekt Java Compiler

Spezielle Kapitel der Praktischen Informatik: Compilerbau

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

05.02.2018

Universität Tübingen

2018-02-04

Projekt Java Compiler

Projekt Java Compiler

Spezielle Kapitel der Praktischen Informatik: Compilerbau

Florian Engel, Robin Heinz, Pavel Karasik, Steffen Lindner, Arwed Mett
05.02.2018
Universität Tübingen

2018-02-04

Projekt Java Compiler

└─ Agenda

Agenda

Test-Framework

Parser

Test-Framework

Parser

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

Aufgabenstellung:

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

Allgemein: Ziel

Ziel: Korrektes Übersetzen der folgenden Klasse:

```
class FibonacciLoop {  
    int getFib(int n) {  
        int f1 = 0, f2 = 1;  
  
        for(int i = n; i > 1; i--) {  
            int tmp = f2;  
            f2 += f1;  
            f1 = tmp;  
        }  
        return (n<2) ? n : f2;  
    }  
}
```

2018-02-04

Projekt Java Compiler

└ Allgemein: Ziel

Allgemein: Ziel

Ziel: Korrektes Übersetzen der folgenden Klasse:

```
class FibonacciLoop {  
    int getFib(int n) {  
        int f1 = 0, f2 = 1;  
  
        for(int i = n; i > 1; i--) {  
            int tmp = f2;  
            f2 += f1;  
            f1 = tmp;  
        }  
        return (n<2) ? n : f2;  
    }  
}
```

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.)

2018-02-04

Projekt Java Compiler

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.)

2018-02-04

Projekt Java Compiler

└─Entwicklung

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

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

Als Build-System wird cabal eingesetzt.

2018-02-04

Projekt Java Compiler
└─ Test-Framework

Test-Framework

Test-Framework

2018-02-04

Projekt Java Compiler
└─ Test-Framework

└─ 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.

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

Tests werden in korrekte und falsche Testfälle unterteilt.

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

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

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

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

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

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

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.

2018-02-04

Projekt Java Compiler
└─ Test-Framework

└─ Test-Suite: Testfälle

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" [] []]

2018-02-04

Projekt Java Compiler
└─ Test-Framework

└─ Test-Suite: Beispiel Testfile

```
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" [] [] ]
```

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

- Multiplikation
- Gaußsumme (kleiner Gauß)
- Fakultät
- Fibonacci

2018-02-04

Projekt Java Compiler
└─ Test-Framework

└─ Test-Suite: Beispielprogramme

Test-Suite: Beispielprogramme

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

- Multiplikation
- Gaußsumme (kleiner Gauß)
- Fakultät
- Fibonacci

Parser


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

2018-02-04

Projekt Java Compiler

└ Parser

└ Parser - Operatoren Priorität

Parser - Operatoren Priorität

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

Program

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

Statement

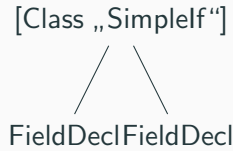
```
: SingleStatement SEMICOLON      { $1 }  
...  
| IF LEFT_PARENTHESSES Expression RIGHT_PARENTHESSES  
  Statement ELSE Statement  
                                     { If $3 $5 (Just $7) }  
  
| IF LEFT_PARENTHESSES Expression  
  RIGHT_PARENTHESSES Statement  
  %prec THEN  
               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

Struktur Happy File

```
Program  
- Class           { [$1] }  
+ Program Class   { $1 ++ [$2] }  
+ Program SEMICOLON { $1 }  
  
Statement  
- SingleStatement SEMICOLON      { $1 }  
  
+ IF LEFT_PARENTHESSES Expression RIGHT_PARENTHESSES  
  Statement ELSE Statement  
                                     { If $3 $5 (Just $7) }  
  
+ IF LEFT_PARENTHESSES Expression  
  RIGHT_PARENTHESSES Statement  
  %prec THEN  
                                     { If $3 $5 Nothing }  
+ Switch                             { $1 }
```



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



2018-02-04



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 └─ Parser

2018-02-04

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