

# Projekt Java Compiler

Spezielle Kapitel der Praktischen Informatik: Compilerbau

---

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

05.02.2018

Universität Tübingen

2018-02-01

## 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

2018-02-01

Projekt Java Compiler

└─ Agenda

Agenda

Test-Framework

Parser

Test-Framework

Parser

2018-02-01

Projekt Java Compiler

└ Allgemein

Allgemein

2018-02-01

Projekt Java Compiler  
└─ Test-Framework

Test-Framework

## Test-Framework

2018-02-01

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 unterschieden.

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

Tests werden in Korrekte und Falsche Testfälle 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.

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.

2018-02-01

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" [] [] ]`

2018-02-01

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" [] [] ]
```



2018-02-01

Projekt Java Compiler  
└─ Parser

Parser

Parser



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

```
%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 }
```

2018-02-01

Projekt Java Compiler  
└─Parser

└─Struktur Happy File

Struktur Happy File

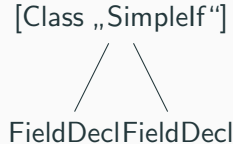
```
Program
- Class           { [$1] }
+ Program Class   { $1 ++ [$2] }
+ Program SEMICOLON { $1 }

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



2018-02-01

Projekt Java Compiler  
└─Parser  
  
└─Beispiel

