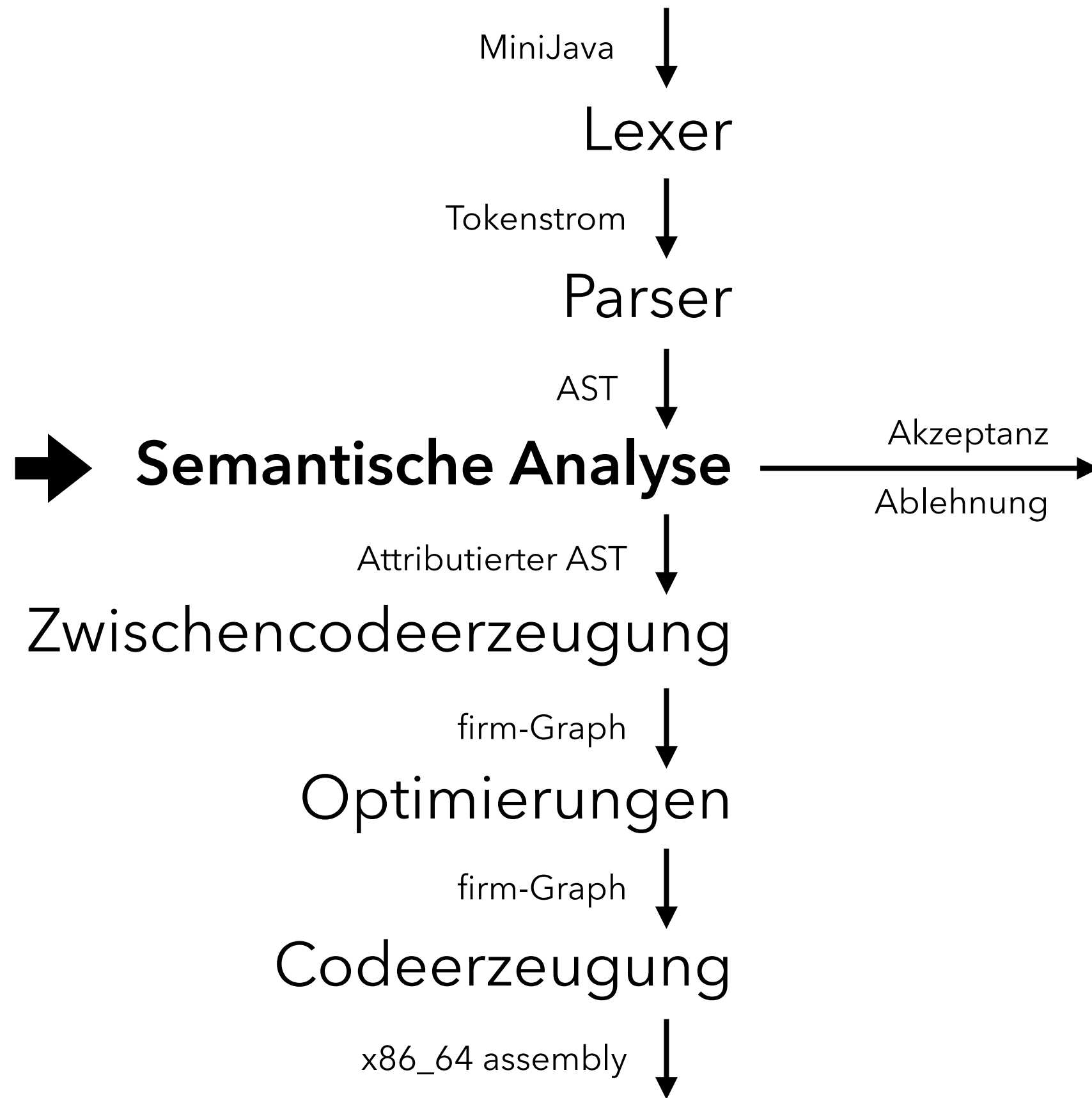


➔ **Semantische Analyse**

Roland Osterrieter, Peter Eisenmann, Marcel Kost, Markus Schlegel



1. Namensanalyse

- Zuordnung von Verwendung zu Deklarationsstelle
- Schwierigkeit: Geschachtelte Namensräume

2. Typanalyse

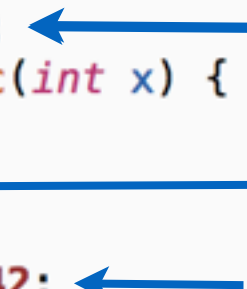
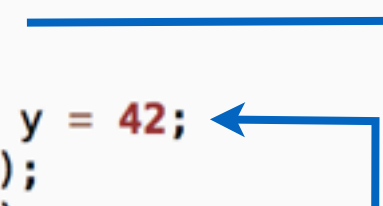

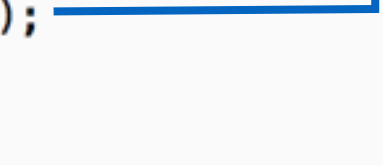
- Typisierung sämtlicher Ausdrücke
- Passen verlangte Typen zu übergebenen Typen?

3. Weitere Prüfungen

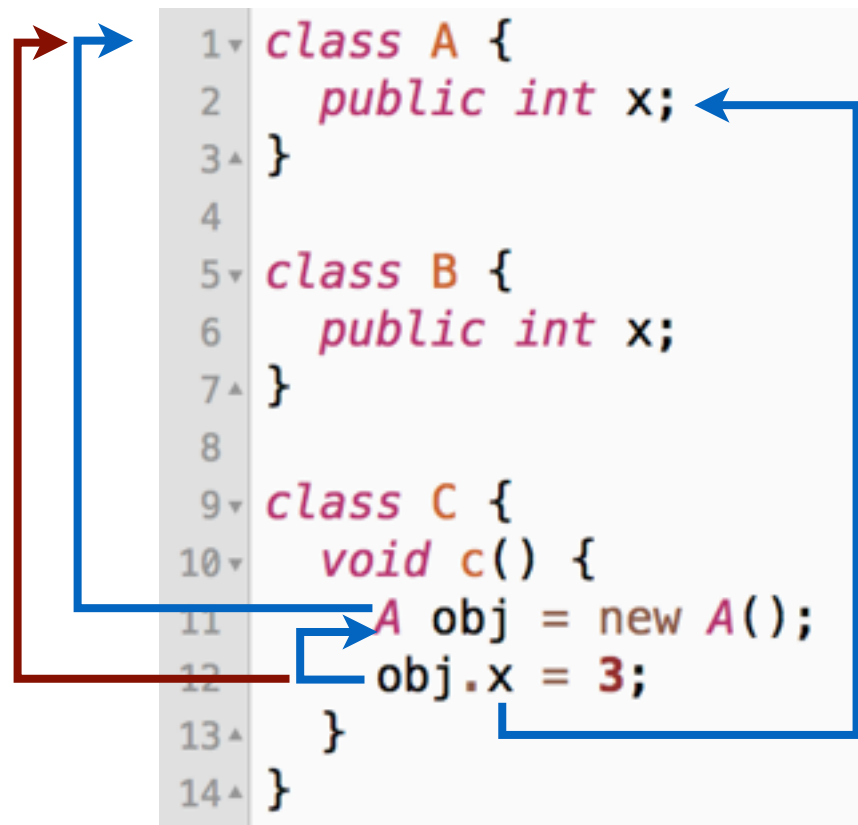
- Return
- MainMethod
- Void-Typ

```
3 public int a(boolean x) {  
4     if (x) {  
5         return 42;  
6     }  
7 }
```

Verschachtelte Namensräume

```
1 class C {  
2     public int x;  
3     public int y;   
4     public void c(int x) {  
5         d(x);  
6         d(y);   
7     {  
8         int y = 42;   
9         d(x);  
10        d(y);   
11    }  
12    d(x);  
13    d(y);   
14 }  
15  
16 public void d(int a) { }  
17  
18 public static void main(String[] args) {}  
19 }
```

Verschränkung von Namens- und Typanalyse



1. **Namensanalyse** löst Typnamen A (11) zu Definition (1) auf
2. **Namensanalyse** löst Bezeichner obj (12) zu Definition (11) auf
3. **Typanalyse** löst Typen von Expression obj (12) zu Typ A auf.
4. **Namensanalyse** löst Attributnamen x (12) zu Definition (2) auf

Außerdem: In Java können Typen, Felder und Methoden im Programmtext vor ihrer Deklaration verwendet werden

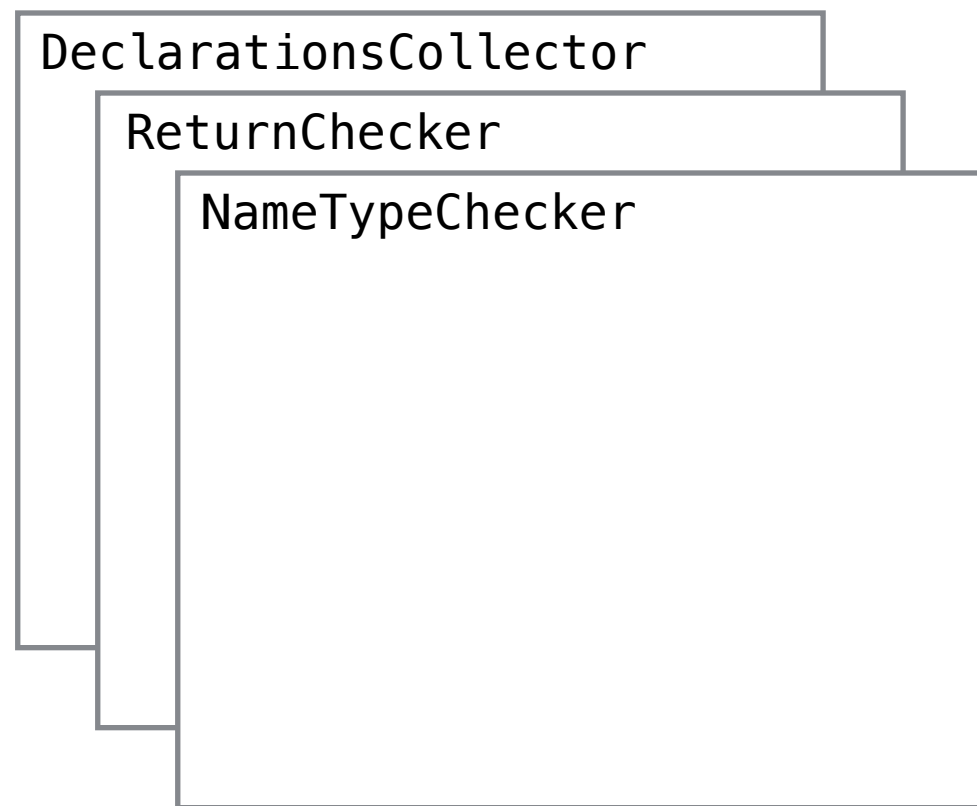
Umsetzung

```
1 class A {  
2     public int x;  
3 }  
4  
5 class B {  
6     public int x;  
7 }  
8  
9 class C {  
10    void c() {  
11        A obj = new A();  
12        obj.x = 3;  
13    }  
14 }
```

1. DeclarationsCollector
2. StaticResolver
3. NameTypeChecker
 1. Setzt type-Attribut jedes Expression-Knotens und überprüft Verwendung
 2. Löst Feld- und Methodenzugriffe auf

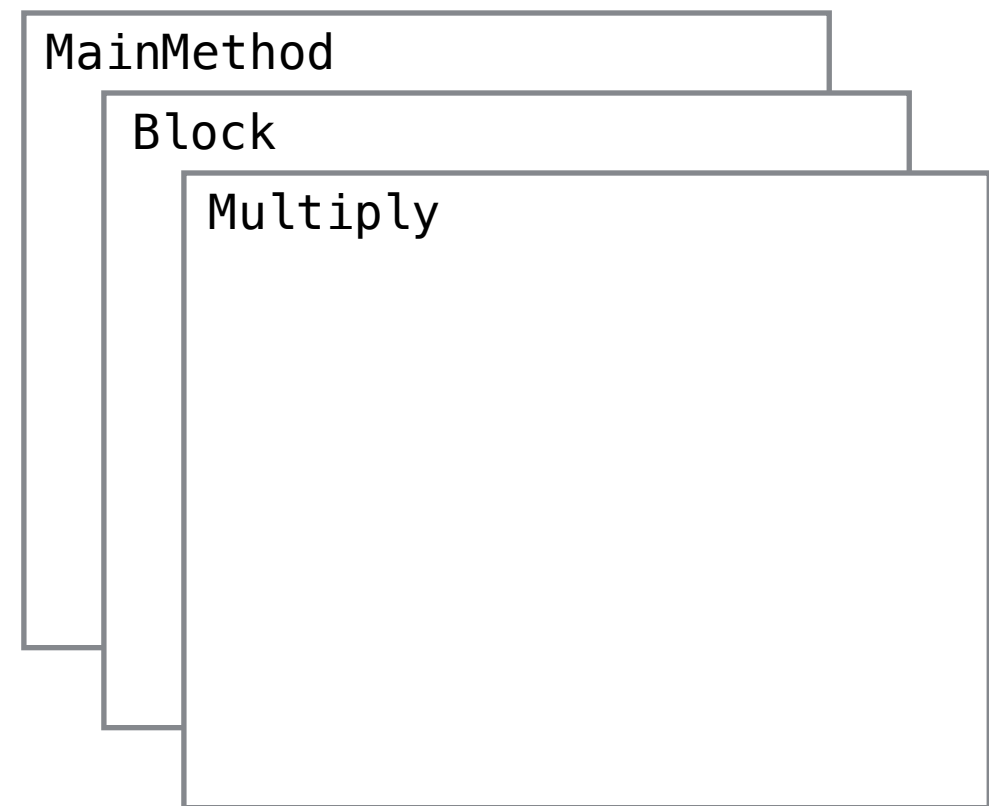
Datenstrukturen

...

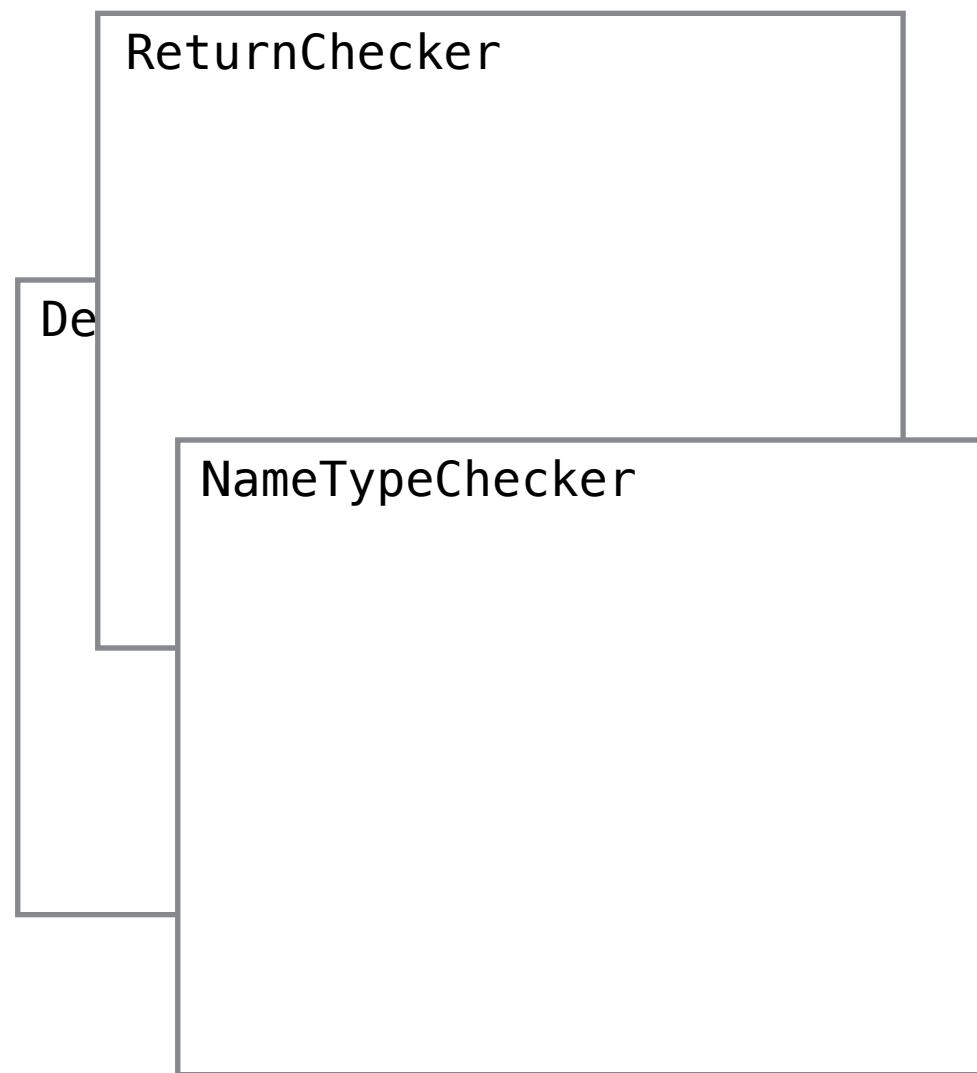


Operationen

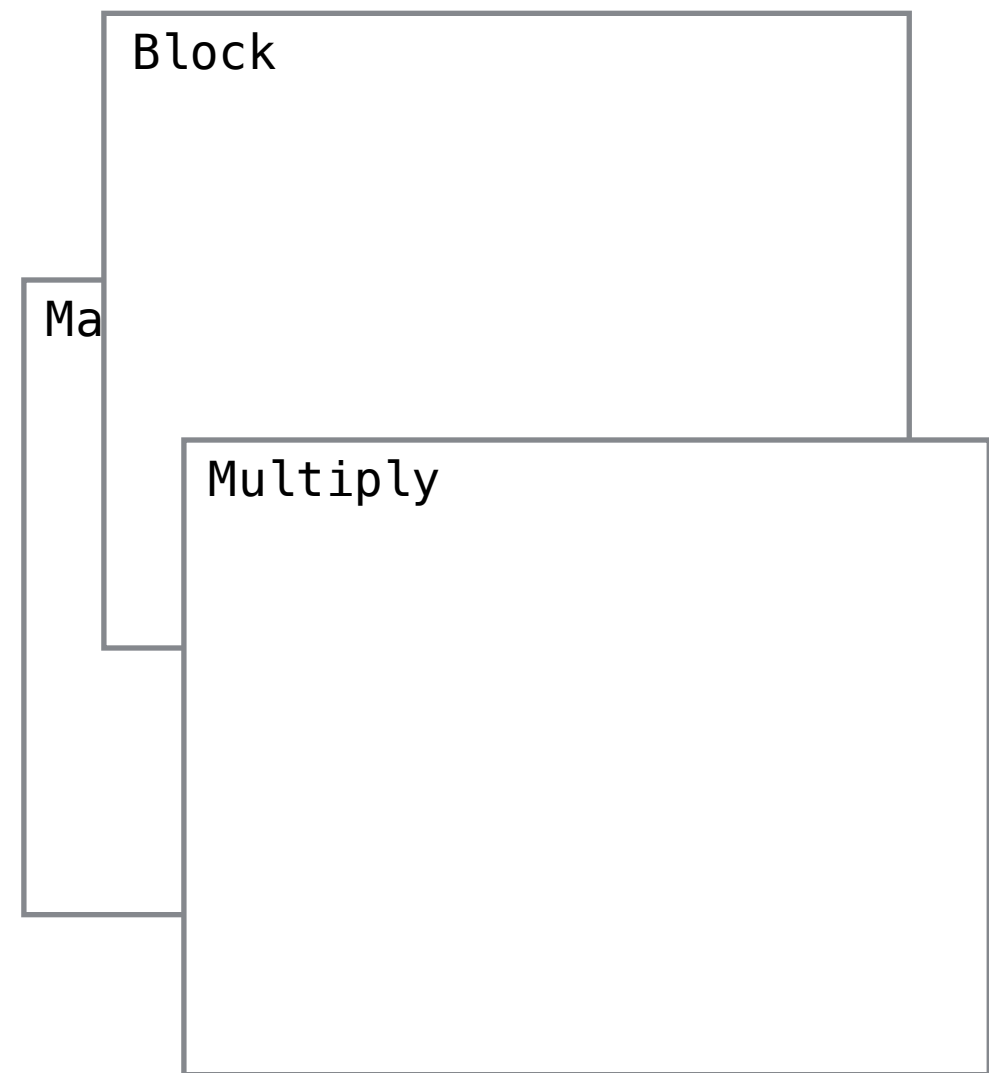
...



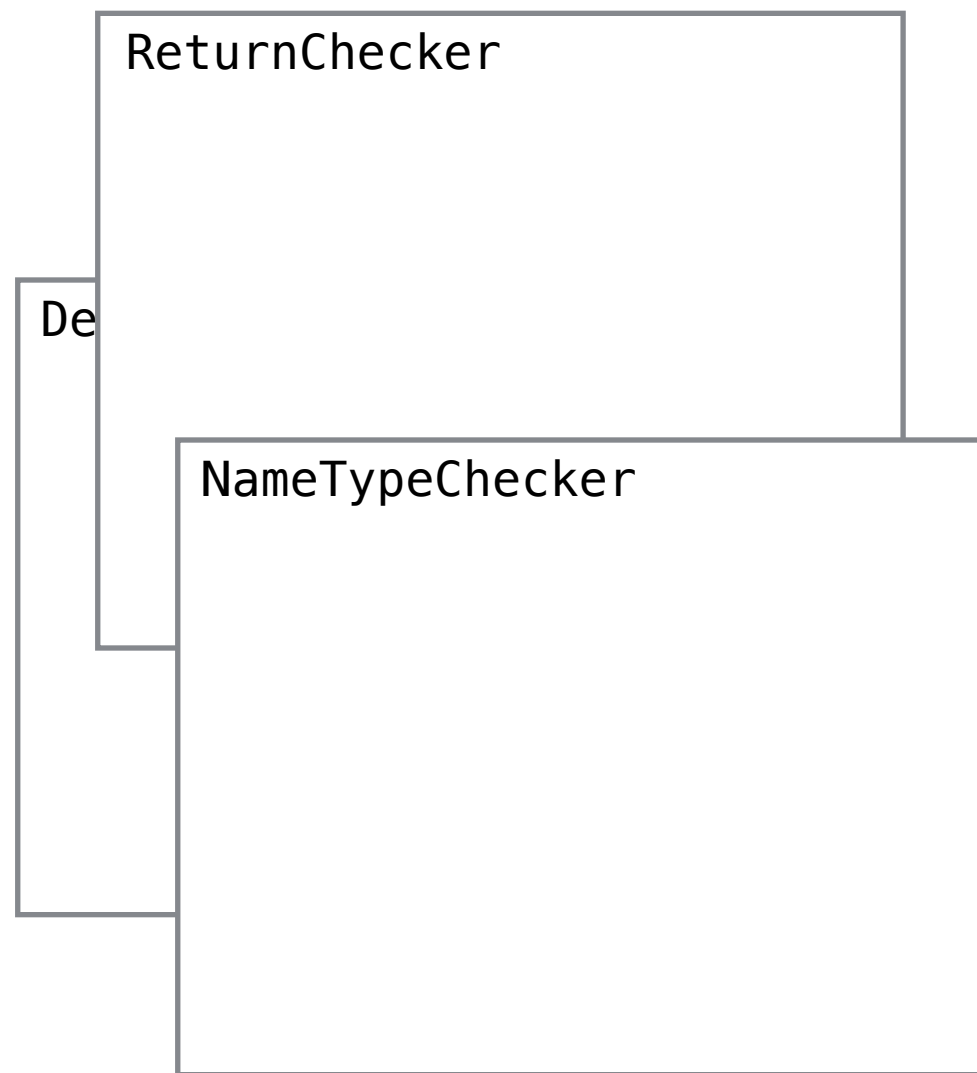
Knotentypen



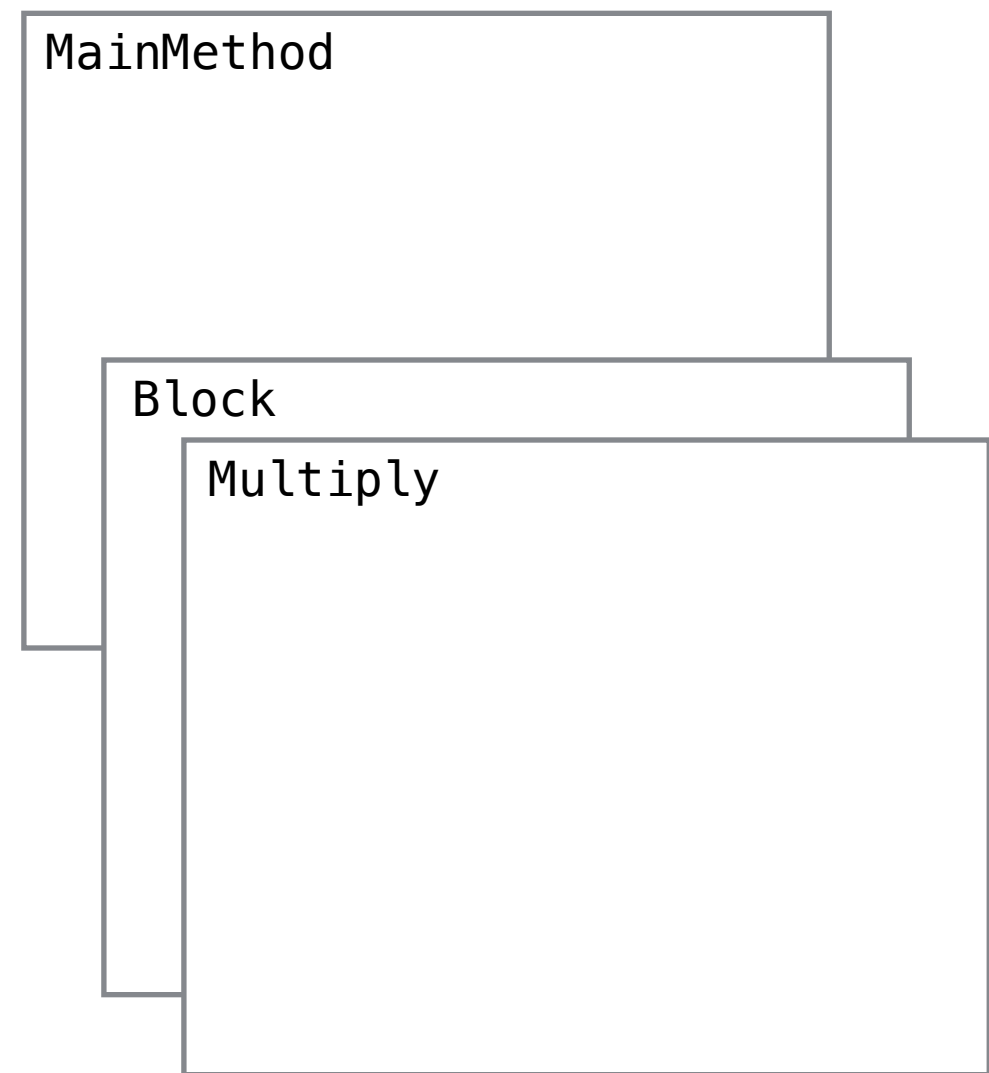
Operationen



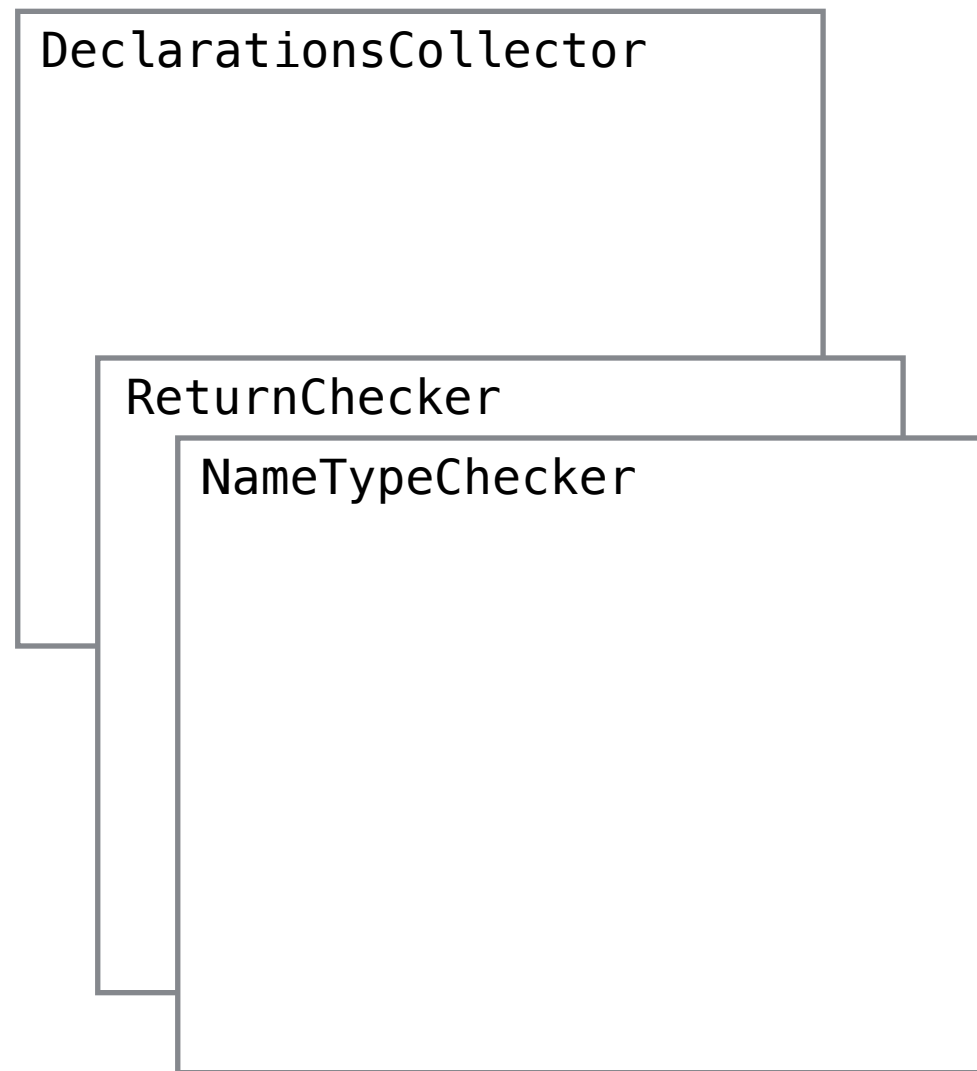
Knotentypen



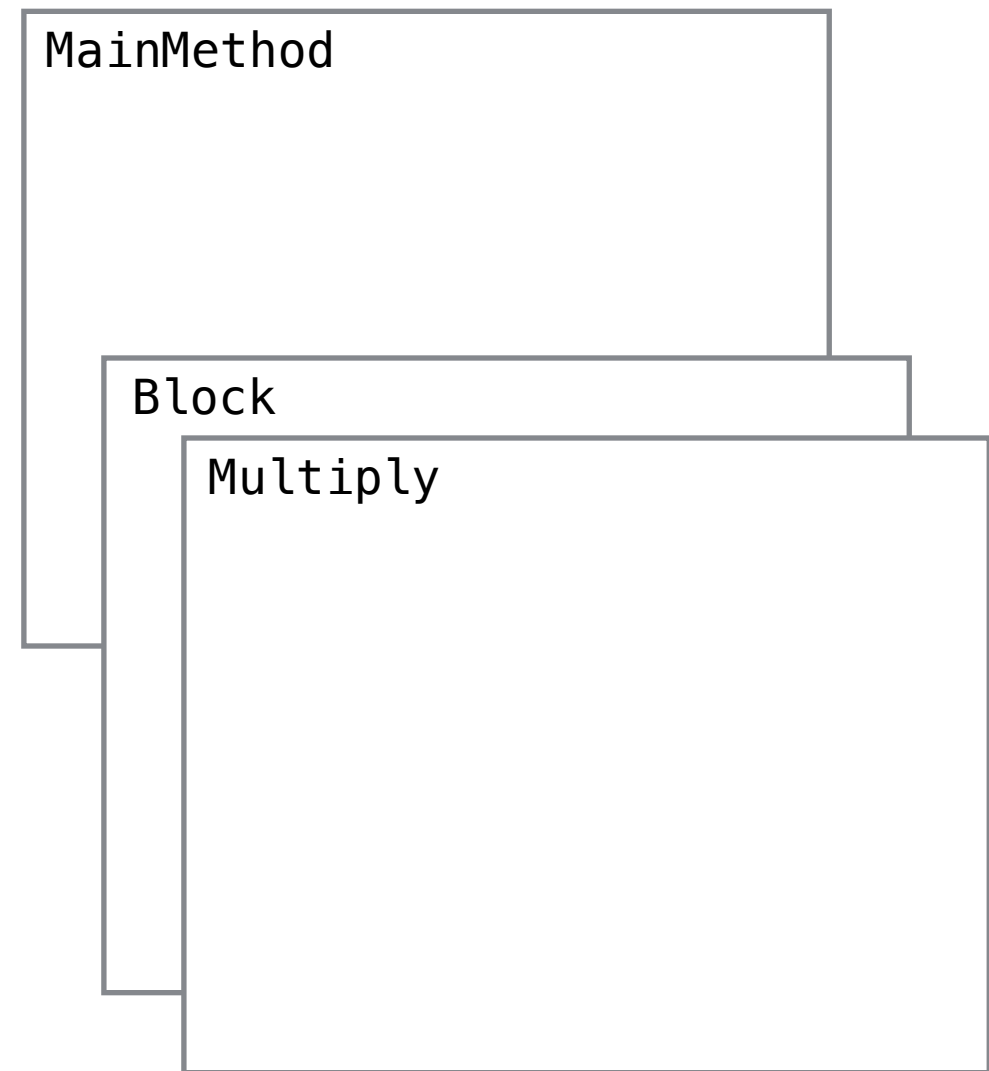
Operationen



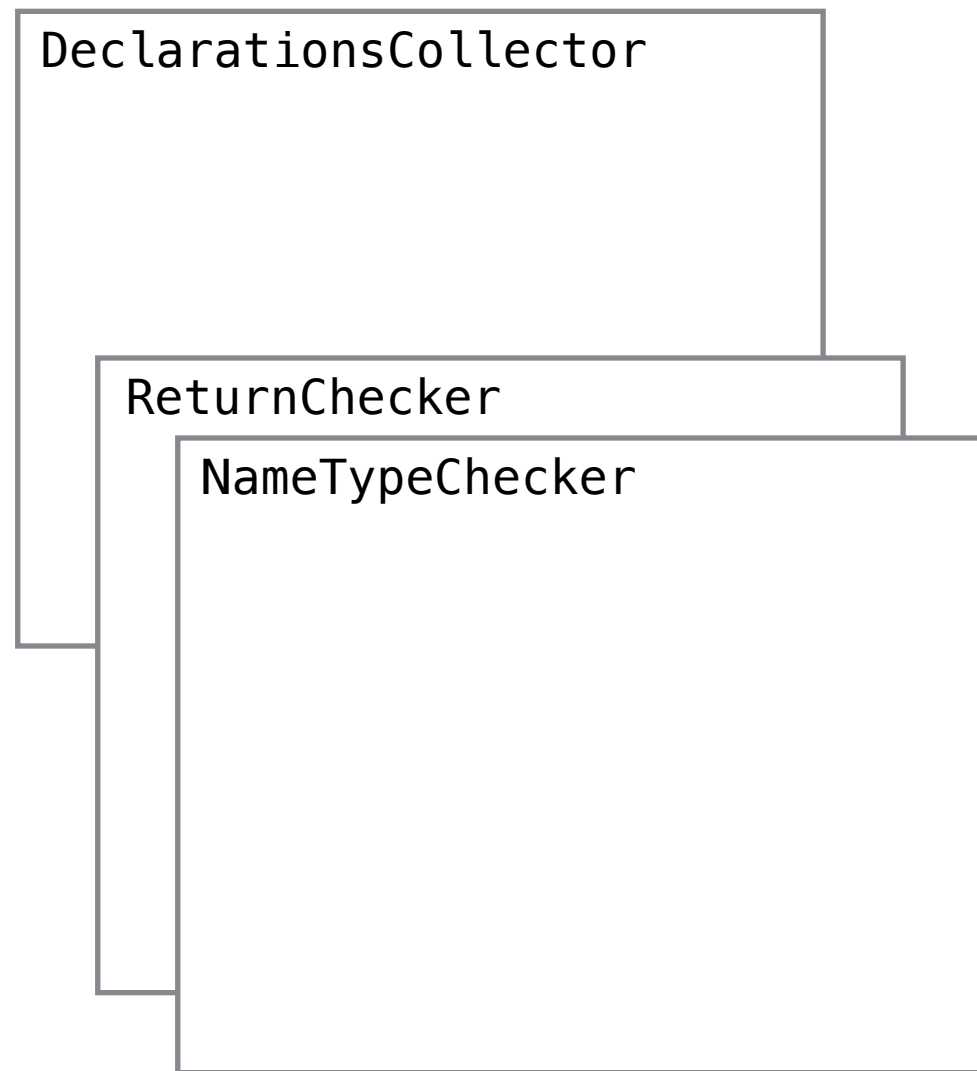
Knotentypen



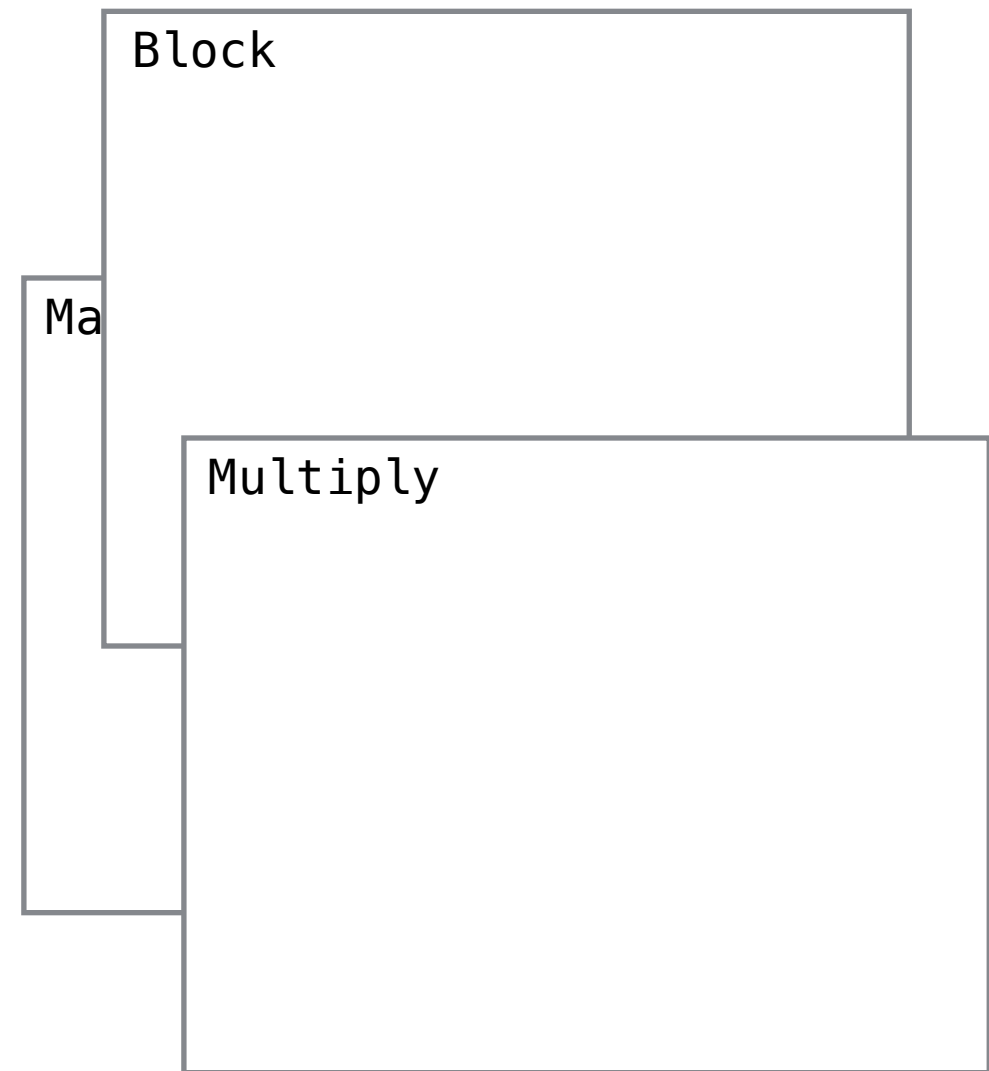
Operationen



Knotentypen



Operationen



Knotentypen

Double Dispatch

Visitor Pattern

```
53 // Returns if one of the statements returns
54 void ReturnChecker::dispatch(std::shared_ptr<Block> n) {
55     for (auto const& s: n->statements) {
56         s->accept(shared_from_this());
57
58         // One of the statements has to return
59         if (s->returns) {
60             n->returns = true;
61             break;
62         }
63     } // dead code recognition could be done here
64 };
```

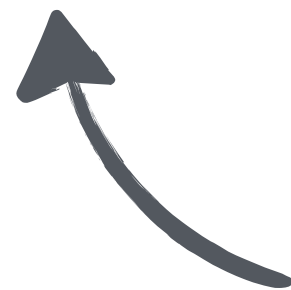
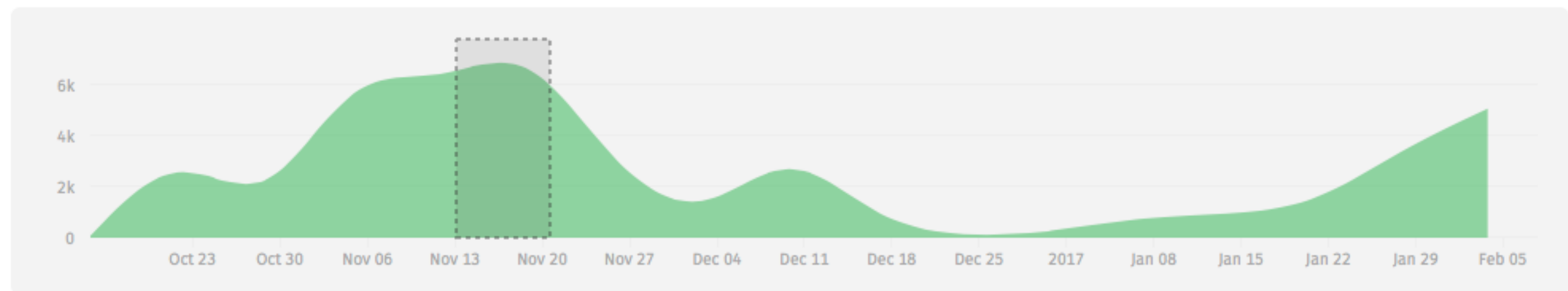
```
78 // Returns if both paths return
79 void ReturnChecker::dispatch(std::shared_ptr<IfElseStatement> n) {
80     n->ifStatement->accept(shared_from_this());
81     n->elseStatement->accept(shared_from_this());
82
83     n->returns = n->ifStatement->returns && n->elseStatement->returns;
84 };
```

Arbeitsaufwand qualitativ

Nov 14, 2016 – Nov 21, 2016

Contributions: **Additions** ▾

Contributions to develop, excluding merge commits



Semantische Analyse

Lines of Code

SL0C	Directory	SL0C-by-Language (Sorted)
2106	semantics	cpp=2106
1699	backend	cpp=1699
1648	structures	cpp=1648
1197	graphcreator	cpp=1197
1109	parser	cpp=1109
670	lexer	cpp=670
418	optimizer	cpp=418
281	compiler	cpp=281
111	main	cpp=111
38	tests	cpp=38
Totals grouped by language (dominant language first):		
cpp:	9277 (100.00%)	
Total Physical Source Lines of Code (SL0C)		= 9,277

Tooling

Generell

C++ 11
CMake
TravisCI
git & GitHub
libfirm

Markus (macOS)

Xcode
ycomp
lldb

Marcel (Linux)

CodeLite
ycomp
gdb

Peter (Linux)

gedit
GnomeBuilder

Roland (Windows + VM)

CodeLite
ycomp