Semantische Analyse

Roland Osterrieter, Peter Eisenmann, Marcel Kost, Markus Schlegel

ASCII

Lexer

Tokenstrom

Parser

AST

Semantische Analyse → Akzeptanz der Eingabe

1. Namensanalyse
2. Typeanalyse
3. Weitere Prüfungen

Attributierter AST

Zwischencodeerzeugung

firm-Graph

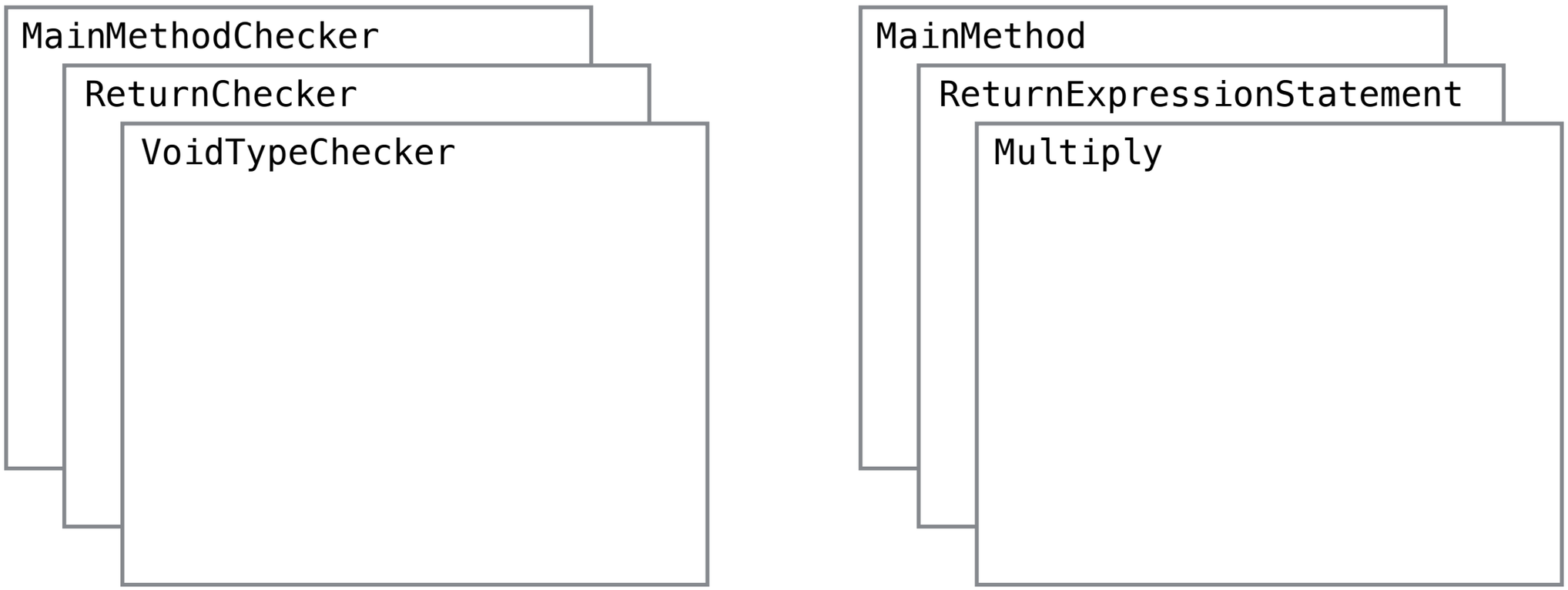
Optimierung

firm-Graph

Codeerzeugung

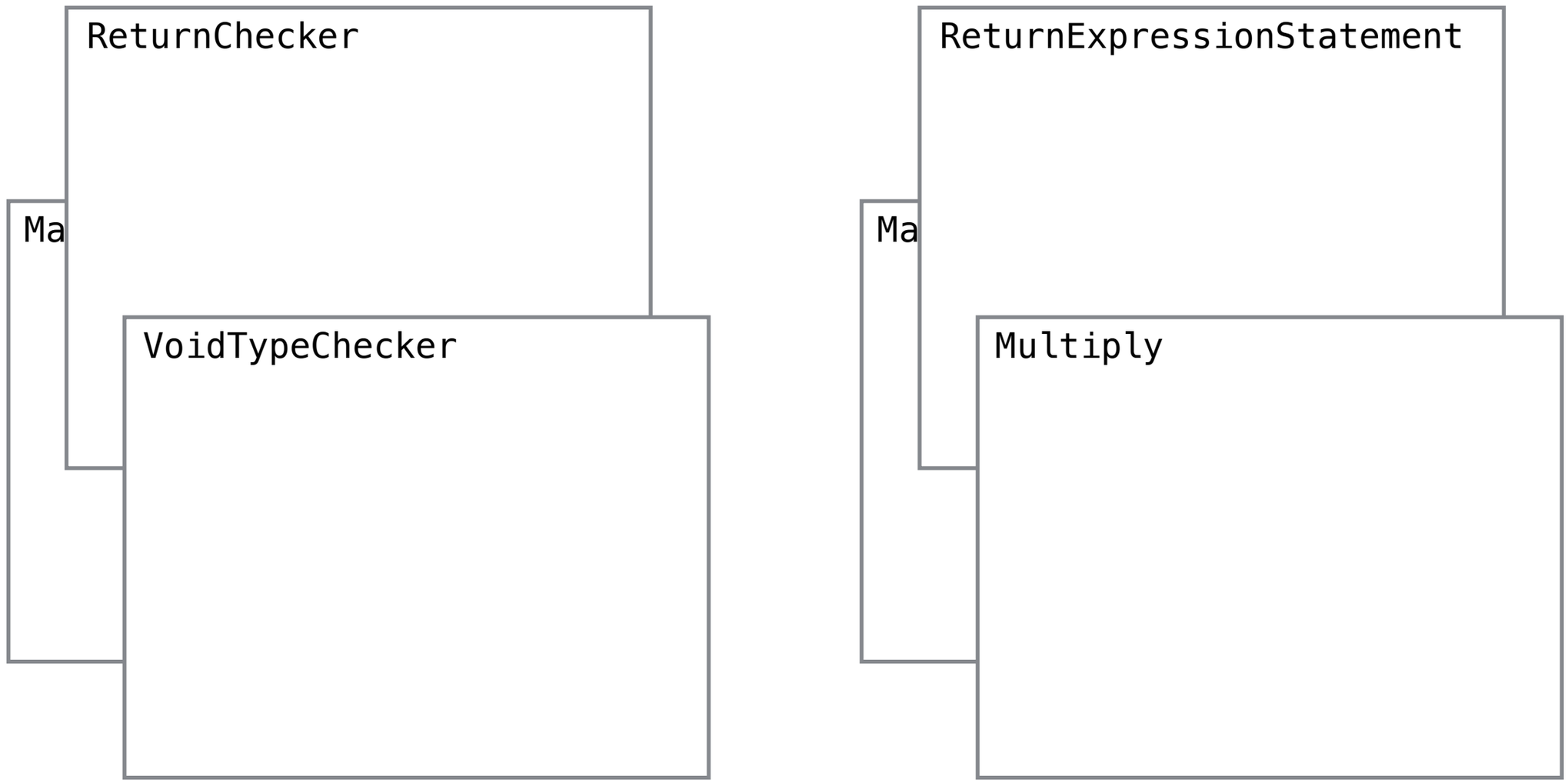
x86_64 assembly

Datenstrukturen



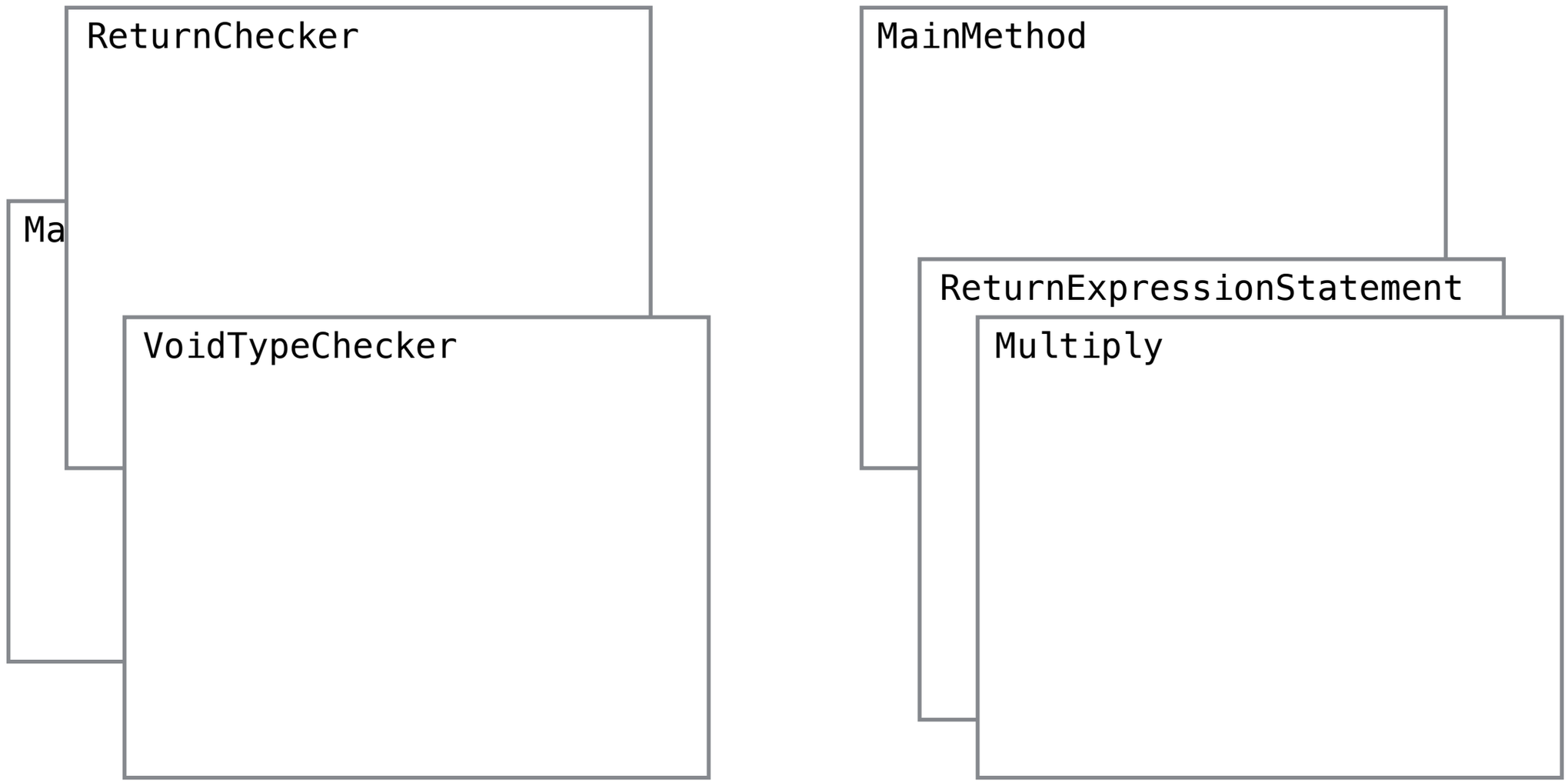
Checker

Knotentypen



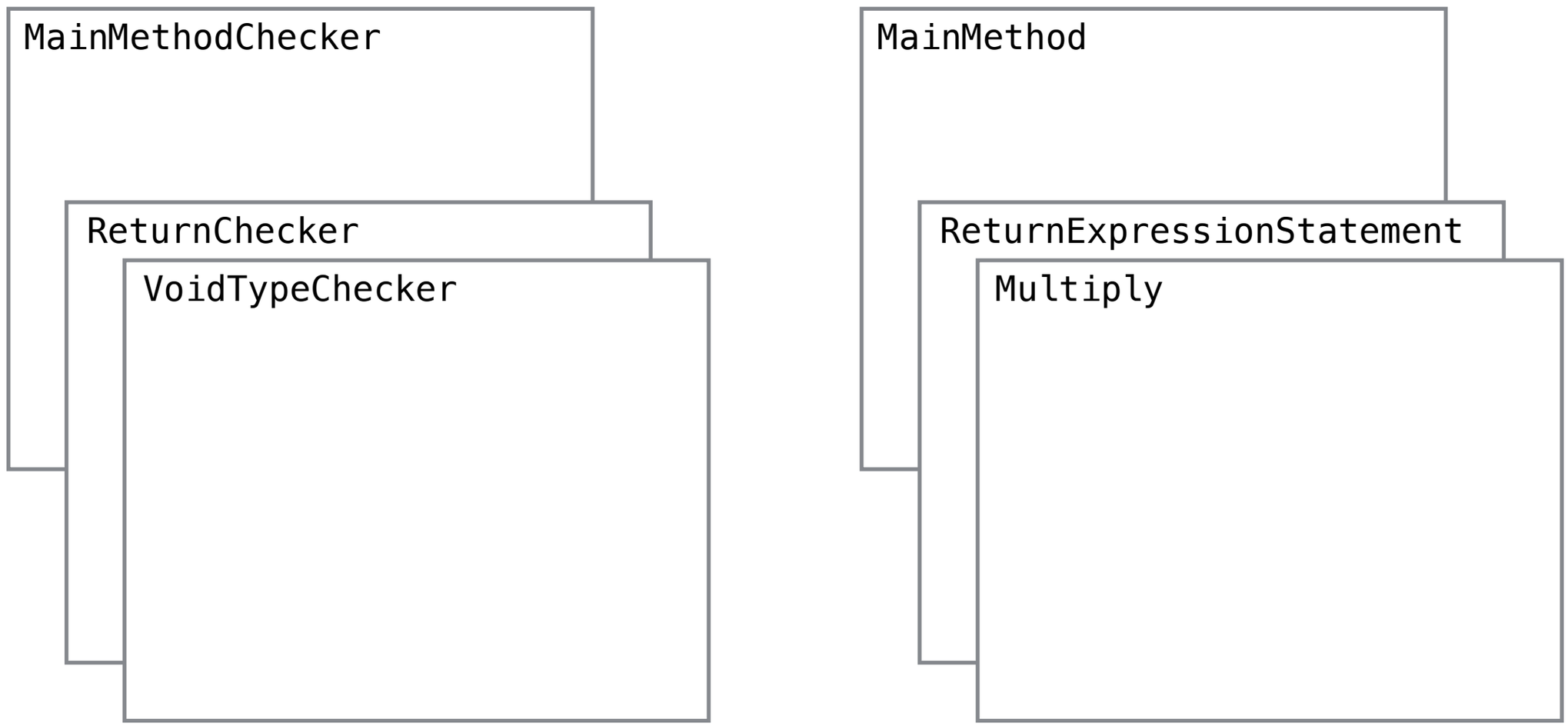
Checker

Knotentypen



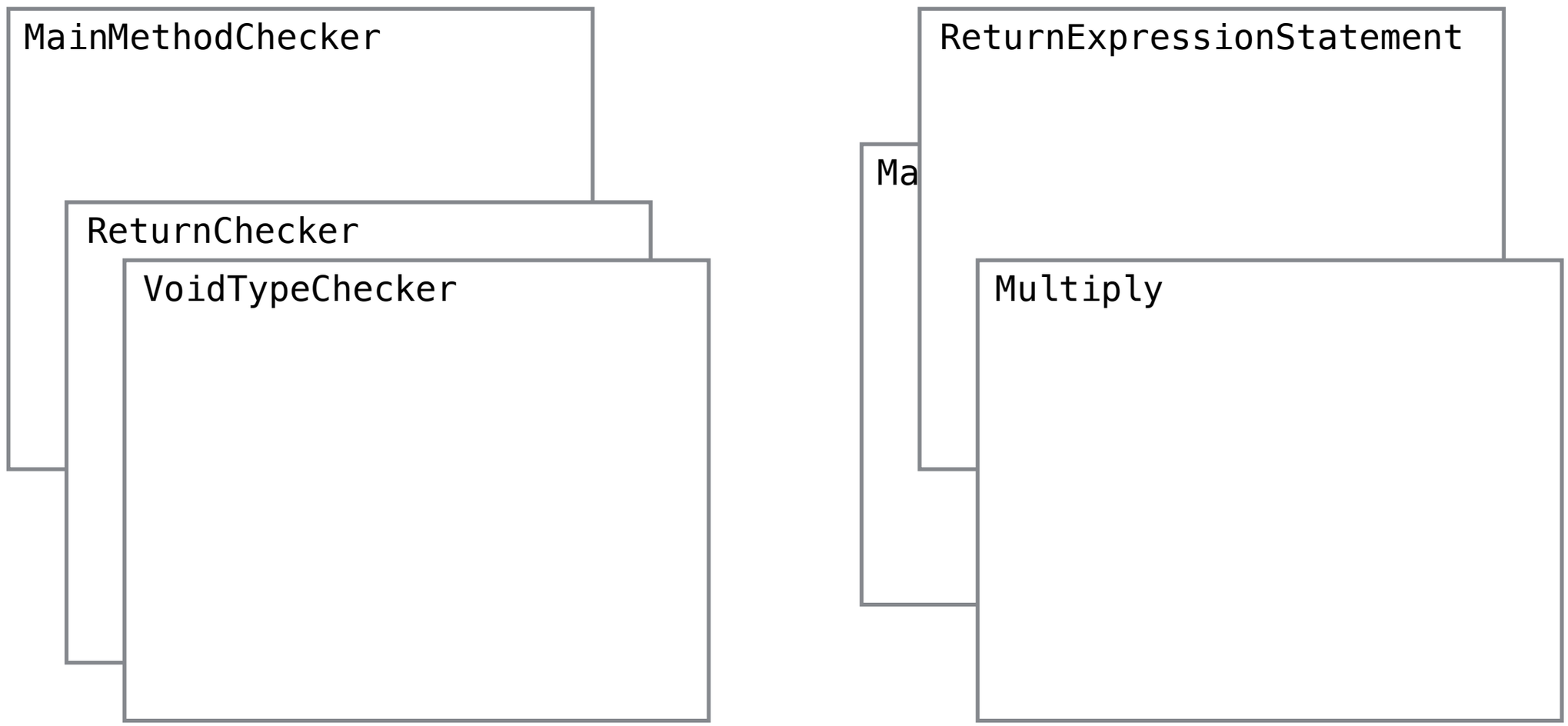
Checker

Knotentypen



Checker

Knotentypen



Checker

Knotentypen

Double Binding
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 };
```

Verschränkung von Namens- und Typanalyse

```
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. 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 Felder und Methoden im Programmtext vor ihrer Deklaration verwendet werden

Umsetzung

1. StaticDeclarationsCollector
    1. Klassennamen
    2. Methodennamen
    3. Feldnamen
    4. Parameter
    5. Lokale Variablen
2. StaticResolver
    1. UserType zu ClassDeclaration
    2. Call zu Method
    3. CRef zu Definition
3. NameTypeChecker
    1. Setzt type-Attribut jedes Expression-Knotens
    2. Löst Feld- und Methodenzugriffe auf

Schwierigkeiten

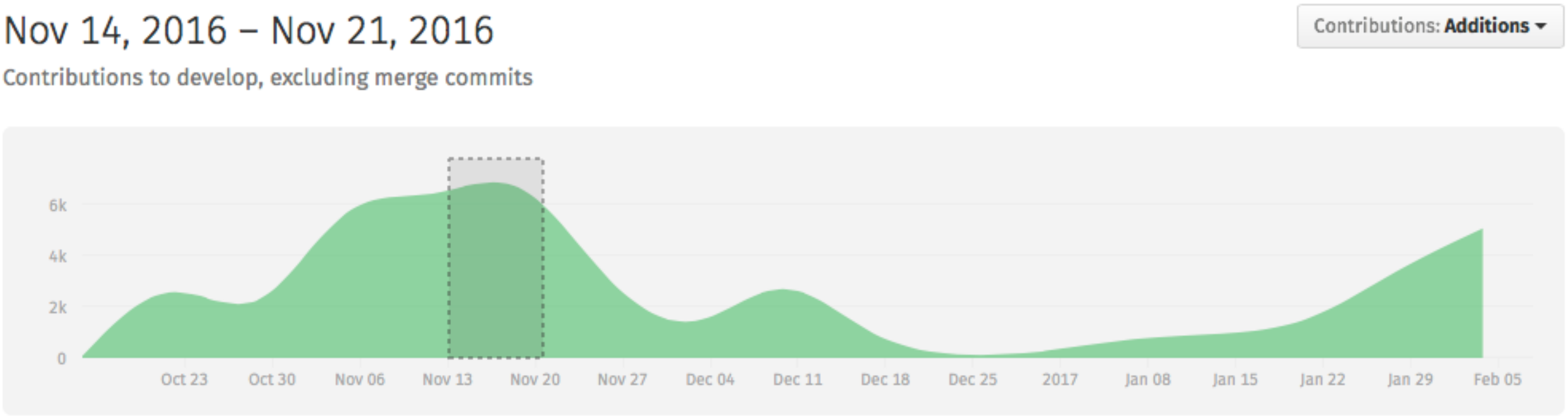
```java
class C {
  public int x;
  public int y;
  public void c(int x) {
    d(x);
    d(y);
    {
      int y = 42;
      d(x);
      d(y);
    }
    d(x);
    d(y);
  }

  public void d(int a) {  }

  public static void main(String[] args) {}
}
```

Verschachtelte Namensräume

Erster Ansatz: komplexe Datenstruktur zur einheitlichen Behandlung von lokalen Variablen, Parameter und Feldern

Später: Getrennte Behandlung von Feldern und Parametern/LocalVars und simple Stack-basierte Symboltabelle

Arbeitsaufwand qualitativ



Semantische Analyse