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

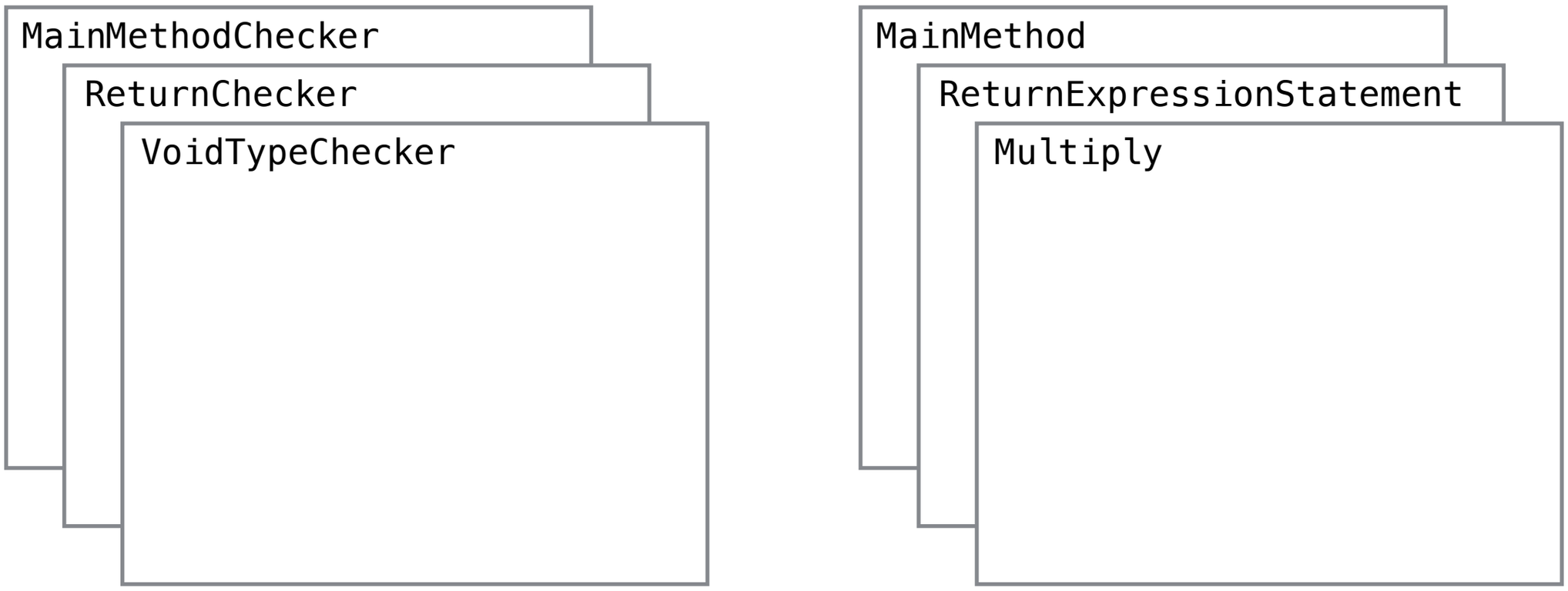
Optimierung

firm-Graph

Codeerzeugung

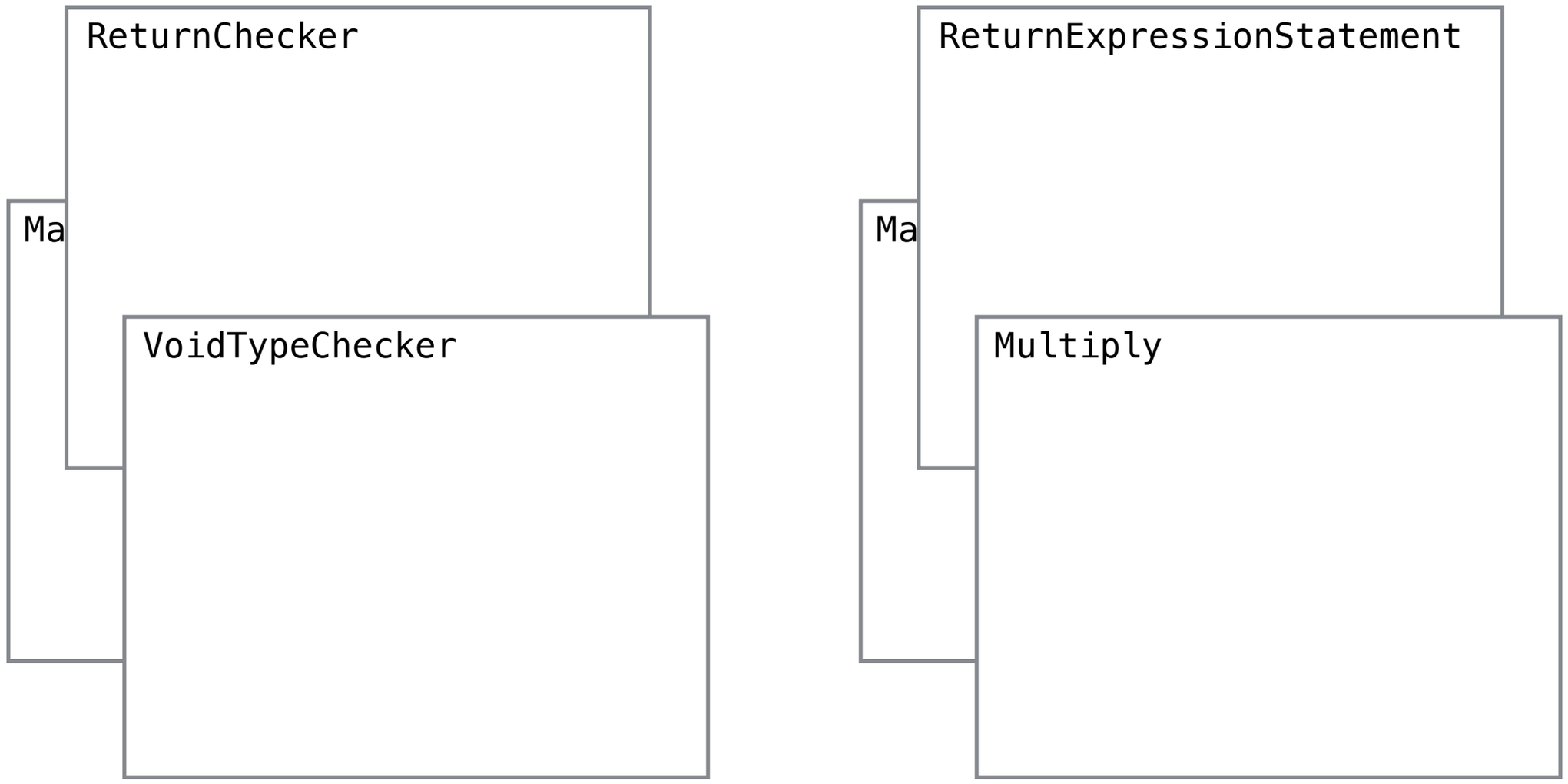
x86_64 assembly

Datenstrukturen



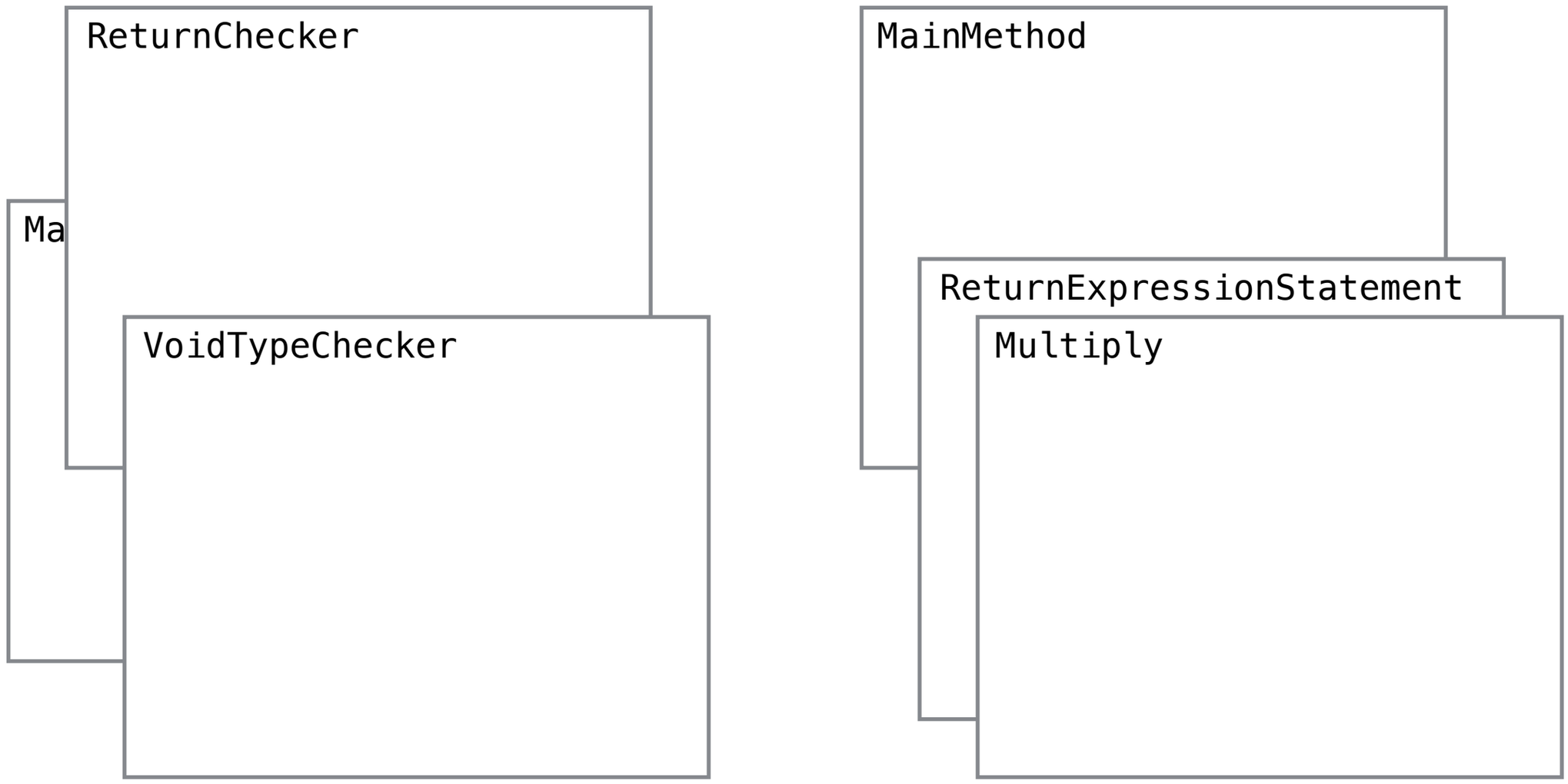
Checker

Knotentypen



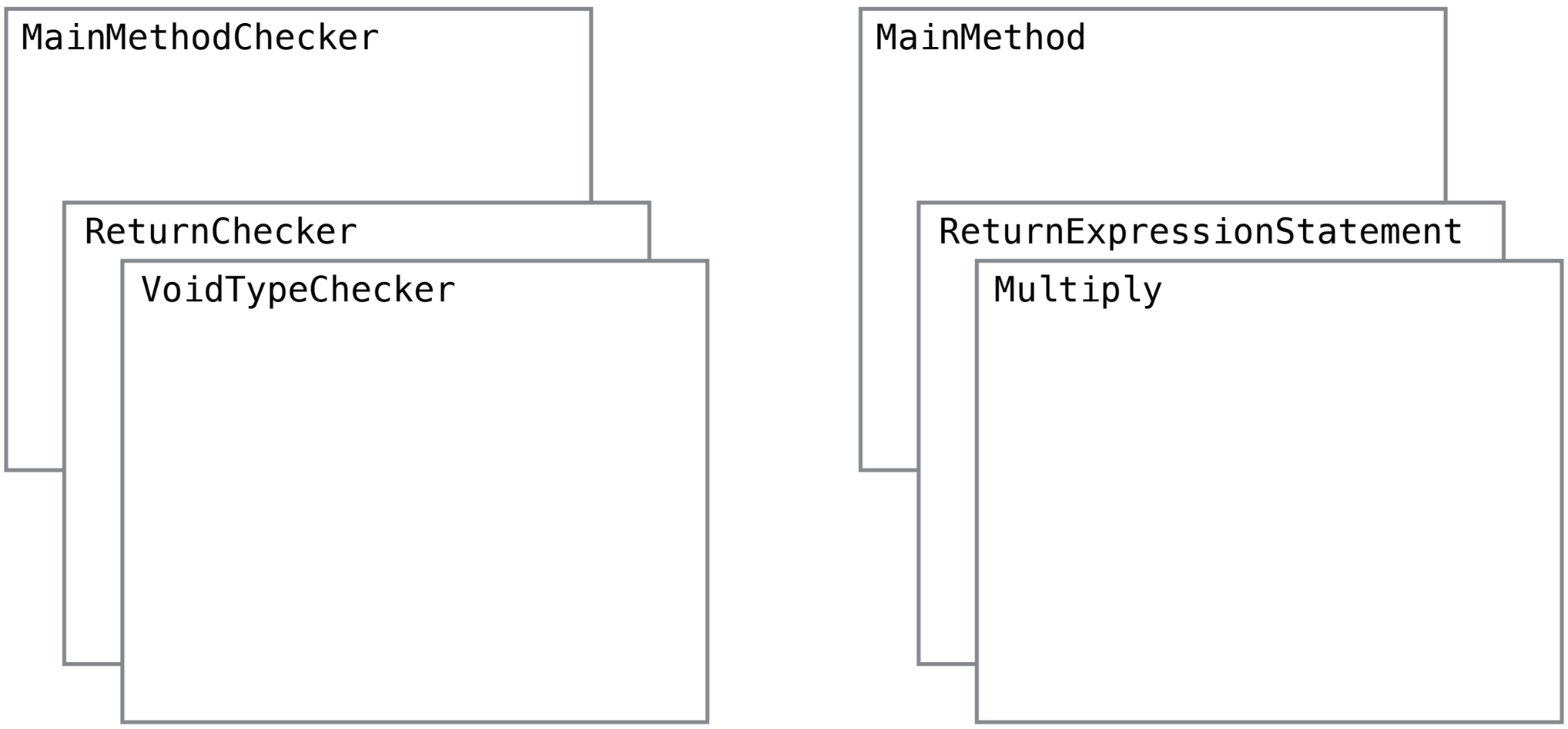
Checker

Knotentypen



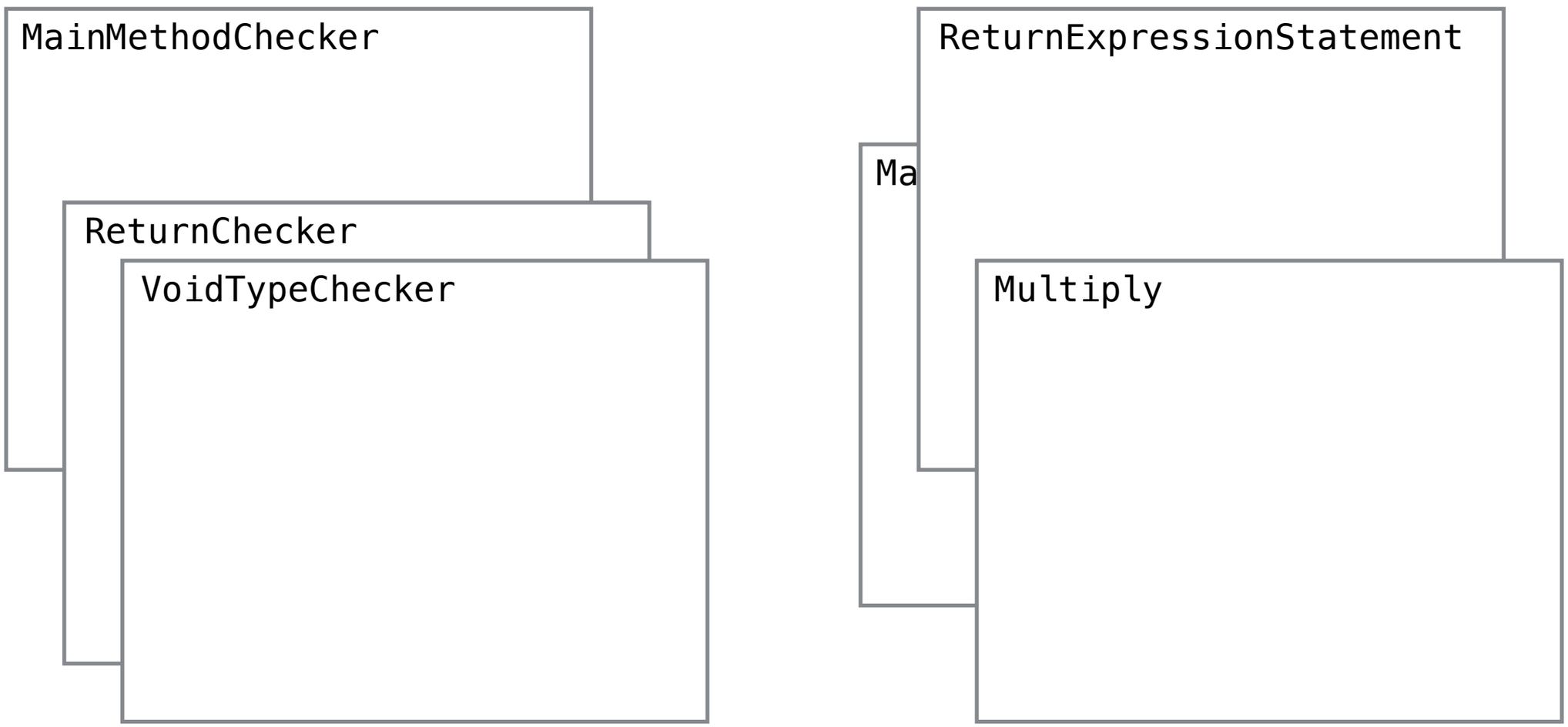
Checker

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. Typanalyse löst Typen von obj (11, 12) zu Typ A auf.
3. Namensanalyse löst Attributnamen x (12) zu Definition (2) auf

Konkret

1. StaticDeclarationsCollector
2. StaticResolver
3. TypeChecker