Vorlesung Fortgeschrittene Softwaretechnik

Wintersemester 2024/25

Prof. Dr. Stephan Diehl

Informatik

Universität Trier

Plan für die nächsten Wochen

Themenblöcke bisher: Testen, CI, Patterns, VR/AR+SE

	Datum	Thema/Inhalt	Übung			Dozent	Block
DO	12.12.2024	Empirische SE + Software Evolution				SD	Quantitative Studien
DI	17.12.2024	MSR, Empfehlungsdienste	Praxis BOA		Ausgabe LIT	SD	
DO	19.12.2024	???					
	FREI						
DI	07.01.2025	RG MSR/BOA	Übung BOA			SD	
DO	09.01.2025	Research Design + Quantitative Analyse				SD	
DI	14.01.2025	Qualitative Analyse	Praxis: QA 1		Ausgabe LIT	SD	Qualitative Studien
DO	16.01.2025	Praxis: QA 2 (gemeinsames Kodieren)				SD	
DI	21.01.2025	RG QA+SE	Übung QA			SD	
DO	23.01.2025	Info zu Portfolio				SD	
DI	28.01.2025	LLM + SE	Übung LLMSE		Ausgabe LIT	SD	LLM+SE
DO	30.01.2025	LLM + SE				SD	
DI	04.02.2025	RG: LLM+SE	Übung LLMSE			SD	
DO	06.02.2025	----------				SD	
DI	11.02.2025	Abgabe Expose				SD	
DO	13.02.2025	----------				SD	

Mining Software Repositories with BOA

Quellen:

- ICSE 2014 Tutorial „**Efficiently Mining Source Code with Boa“,** *Robert Dyer, Haam Nguyen, Hridesh Rajan, and Tien Nguyen*, Iowa Sate University, United States.

  https://boa.cs.iastate.edu/papers/icse14-tutorial-slides.pptx

- "Boa: A Language and Infrastructure for Analyzing Ultra-Large-Scale Software Repositories", *Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan and Tien N. Nguyen*, ICSE 2013

→ http://boa.cs.iastate.edu/

Boa's Architecture

Boa's Data Infrastructure

GitHub

sourceforge

Replicate

and Transform

cache

Stored on cluster

Query result returned via web

User submits query

Compiled into Hadoop program

Deployed and executed on cluster

Boa's Computing Infrastructure

Von BOA gespeicherte Daten von GitHub

2019 October/GitHub (Java)

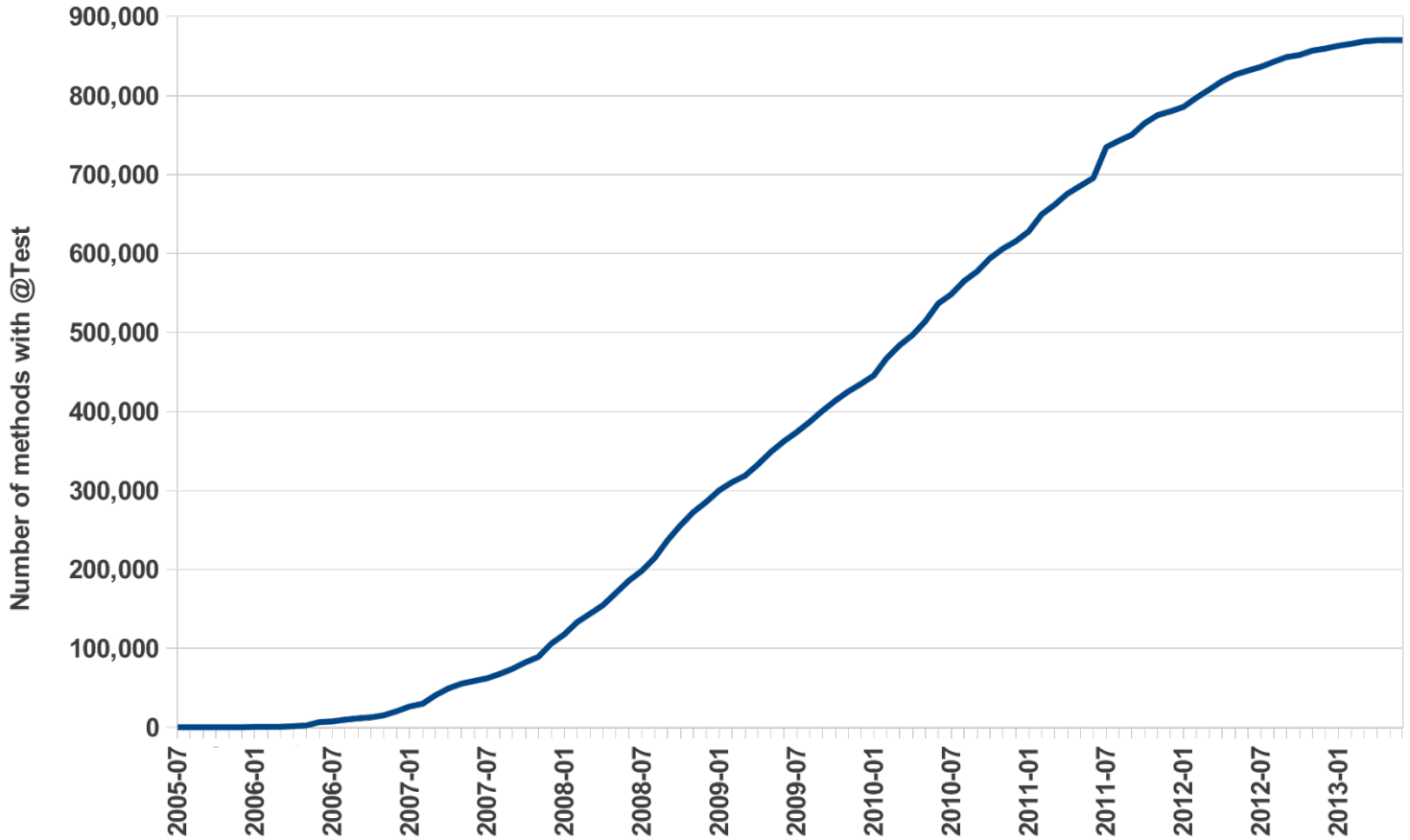
Projects	**7,830,023**
Code Repositories	380,125
Revisions	**23,229,406**
Unique Files	**146,398,339**
File Snapshots	484,947,086
AST Nodes	**71,810,106,868**

2022 February/Python

Projects	**102,424**
Code Repositories	102,424
Revisions	**32,232,939**
Unique Files	**63,681,580**
File Snapshots	293,231,664
AST Nodes	**105,512,426,611**

Over 250GB of *pre-processed* data

How has unit testing evolved over time?



How can we compute these numbers with BOA?

Junit (ab Version 4.x, 2006)

Klasse

```java
package math;

public class Math {

    String name;

    public Math(String name) {
        this.name = name;
    }

    public int add(int x,int y) {
        return x+y;
    }

    public double divide(double x,double y) {
        if (y == 0) {
            throw new ArithmeticException(
                    "Division by zero");
        }
        return x/y;
    }

    @Override
    public String toString() {
        return name;
    }

}
```

Testklasse

```java
import static org.junit.Assert.*;

public class MathTest {

    Math m;

    @Before
    public void setUp() throws Exception {
        m = new Math("My Math Object");
    }

    @Test
    public void testAddPositiveNumbers() {
        assertEquals("add positive numbers", 4, m.add(2,2));
    }

    @Test
    public void testDividePositiveNumbers() {
        assertEquals(0.33333, m.divide(1,3), 0.0001);
    }

    @Test
    public void testDivideByZero1() {
        try {
            m.divide(1,0);
            fail("Should not be possible to divide by zero");
        } catch (ArithmeticException e) {
        }
    }

    @Test (expected = ArithmeticException.class)
    public void testDivideByZero2() {
        m.divide(1,0);
    }

}
```

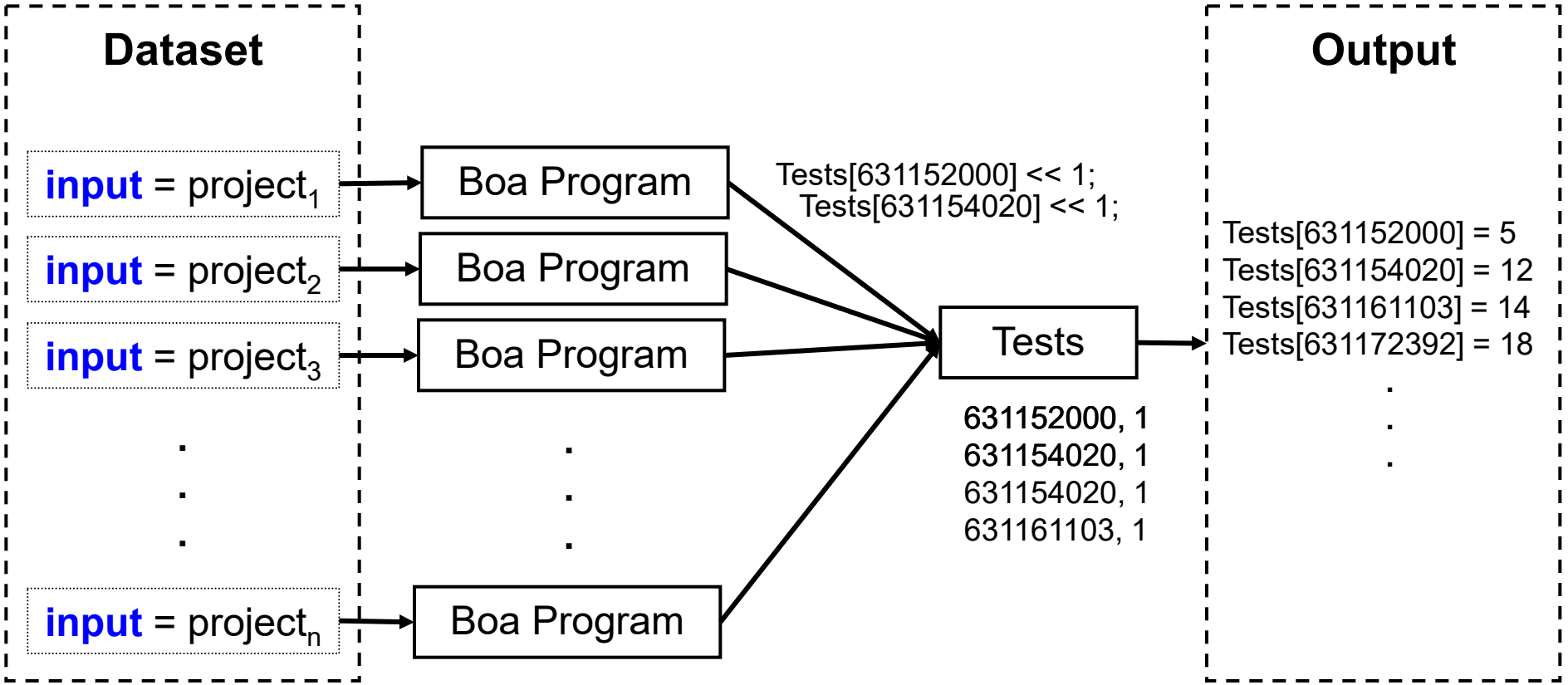
How has unit testing evolved over time?

- Zähle @Test, @junit.Test, @org.junit.Test

```
TestAnnotation: output sum[time] of int;

cur_time:time;

visit(input, visitor {
  before n: Revision -> cur_time = n.commit_date;
  before n: Modifier ->
    if (n.kind == ModifierKind.ANNOTATION &&
        match(`^(Test)`, n.annotation_name))
      TestAnnotation[cur_time] << 1;
});
```




```
Tests: output sum[time] of int;

cur_time:time;

visit(input, visitor {
  before n: Revision -> cur_time = n.commit_date;
  before n: Modifier ->
    if (n.kind == ModifierKind.ANNOTATION &&
        match(`^(org\.junit\.)?Test$`, n.annotation_name))
      Tests[cur_time] << 1;
});
```

Automatic Parallelization

Output variables with built in **aggregator functions**:
sum, mean, top(k), bottom(k), set, collection, etc

```
Tests: output sum[time] of int;

cur_time:time;

visit(input, visitor {
  before n: Revision -> cur_time = n.commit_date;
  before n: Modifier ->
    if (n.kind == ModifierKind.ANNOTATION &&
        match(`^(org\.junit\.)?Test$`, n.annotation_name))
      Tests[cur_time] << 1;
});
```

Compiler generates **Hadoop MapReduce** code

Abstracting MSR with Types

Custom **domain-specific types** for mining software repositories:
base types and **types for source code**

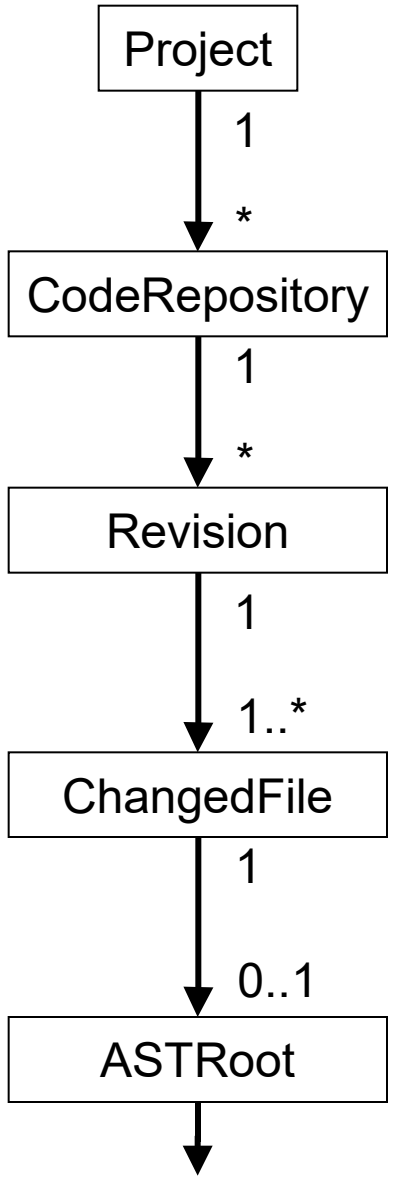
```
Tests: output sum[time] of int;

cur_time:time;

visit(input, visitor {
  before n: Revision -> cur_time = n.commit_date;
  before n: Modifier ->
    if (n.kind == ModifierKind.ANNOTATION &&
        match(`^(org\.junit\.)?Test$`, n.annotation_name))
      Tests[cur_time] << 1;
});
```

No need to understand multiple data formats or APIs

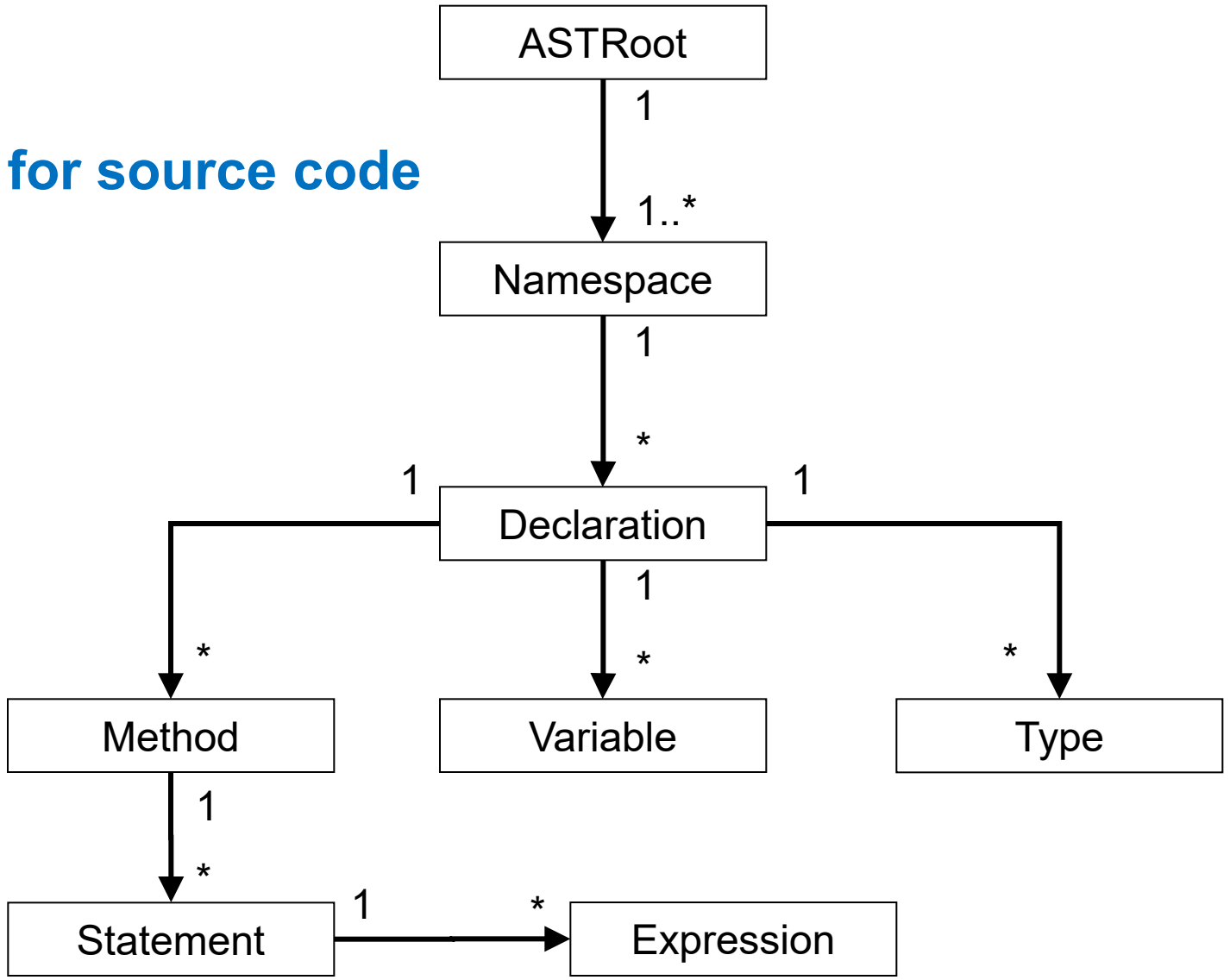
Abstracting MSR with Types



Base types

Abstracting MSR with Types

Types for source code



Beispiel: Anzahl von Klassendeklarationen

```
# How many classes are declared in each project
# incl. all revisions
p: Project = input;
ClassDecs: output sum[string] of int;

visit(p, visitor {
        before node: Declaration -> {
            if (node.kind==TypeKind.CLASS)
                ClassDecs[p.project_url] << 1;

            # also look at nested declarations
            foreach (i: int; node.methods[i])
               visit(node.methods[i]);
            foreach (i: int; node.nested_declarations[i])
               visit(node.nested_declarations[i]);
            stop;
         } } );
```

ClassDecs	http://sourceforge.net/projects/g9json	3

Snapshot

getsnapshot (cr: CodeRepository [, t: time] [, filters: string...]) : array of ChangedFile

- Liefert eine Array mit den letzten Versionen einer Datei vor dem angegeben Zeitpunkt. Wenn eine Datei davor gelöscht wurde, ist sie nicht enthalten.
- Wenn kein Zeitpunkt angegeben wurde, wird now() verwendet, also die letzte Version überhaupt.
- Wenn Filter angegeben wurden, dann werden nur Dateien zurückgeliefert, deren Dateityp (file kind) mit einem Filter beginnt.
- **getsnapshot(cr)** → letzte Versionen aller Dateien
- **getsnapshot("SOURCE_JAVA_JLS")** → letzte Versionen aller Java-Dateien
- **getsnapshot("SOURCE_PY_")** → letzte Versionen aller Python-Dateien

Beispiel: Anzahl von Klassendeklarationen

```
# How many classes are declared in the last „revision“
# of each project?
p: Project = input;
ClassDecs: output sum[string] of int;

myvisitor := visitor {
         before node: Declaration -> {
           if (node.kind==TypeKind.CLASS) ClassDecs[p.project_url] << 1;

            # also look at nested declarations
            foreach (i: int; node.methods[i]) visit(node.methods[i]);
            foreach (i: int; node.nested_declarations[i]) visit(node.nested_declarations[i]);
            stop;
        } } ;

visit(p, visitor { before node: CodeRepository ->
                    { snapshot := getsnapshot(node);
                      foreach(i :int; def(snapshot[i]))
                       visit(snapshot[i], myvisitor);
                      stop; }
                 });
```

```
ClassDecs    http://sourceforge.net/projects/g9json    2
```

Beispiel: Häufigkeit von Klassennamen

```
# Compute the class names
# and how often they occur in all revisions?
p: Project = input;
ClassNames: output sum[string] of int;

visit(p, visitor {
        before node: Declaration -> {
            if (node.kind==TypeKind.CLASS)
                ClassNames[node.name] << 1;

            # also look at nested declarations
            foreach (i: int; node.methods[i])
              visit(node.methods[i]);
            foreach (i: int; node.nested_declarations[i])
              visit(node.nested_declarations[i]);
            stop;
         }

});
```

ClassNames	AccessType	1
ClassNames	Access	886

Beispiel: die 5 häufigsten Klassennamen

```
# What are the top 5 common class names
# in all revisions?
p: Project = input;
ClassNames: output top(5) of string weight int;

visit(p, visitor {
        before node: Declaration -> {
            if (node.kind==TypeKind.CLASS)
                ClassNames << node.name weight 1;

            # also look at nested declarations
            foreach (i: int; node.methods[i])
              visit(node.methods[i]);
            foreach (i: int; node.nested_declarations[i])
              visit(node.nested_declarations[i]);
            stop;
         }

});
```

ClassNames = Access, 886
ClassNames = Bean, 249
ClassNames = User, 239
ClassNames = Main, 208
ClassNames = Render, 203

Übungsaufgaben

Aufgabe 1: Relative Häufigkeit mit BOA (4 Punkte)

a) Implementieren Sie ein BOA-Skript, das für jedes Projekt in temporären Variablen (vom Typ float) die Anzahl der Statements und Try-Statements für den aktuellen Stand jedes Projektes (also alle letzten Dateirevisionen) berechnet und das Verhältnis der beiden Zahlen (also wieviel Prozent der Statements sind Try-Statements) für jedes Projekt ausgibt. Anschließend führen Sie Ihr Skript auf den Datensätzen 2022 Feb/Python und 2022 Jan/Java aus und laden die Ausgabe herunter.

b) Erweitern Sie Ihr Skript so, dass es das Verhältnis der beiden Zahlen für den Stand des Projektes am Ende jeden Jahres (2000 bis 2022) berechnet und ausgibt.

Hinweis: In Python gibt es die Anweisung `open ... as ...`, die syntaktischer Zucker für ein komplexes TRY-Statement ist. Für den Python-Datensatz gibt es daher das zusätzlich Attribut `StatementKind.WITH`.

Aufgabe 2 (→ Übungsblatt)

1. Statistische Auswertung mit Python (optional R) nach Vorlesung am 9.1.2025