# Ein erweiterbarer Compiler zum feature-orientierten Programmieren in Java

Sergiy Kolesnikov

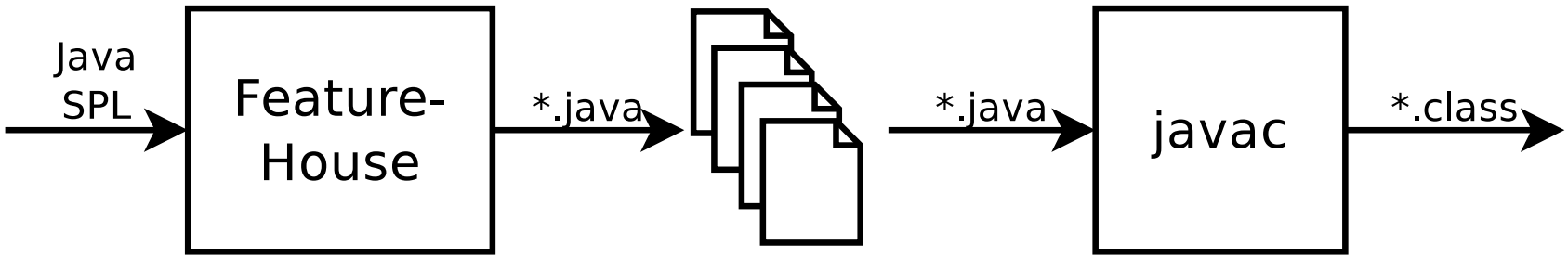
SPL Group, Universität Passau

FOSD Treffen Dresden
10.03.2011


# Outline

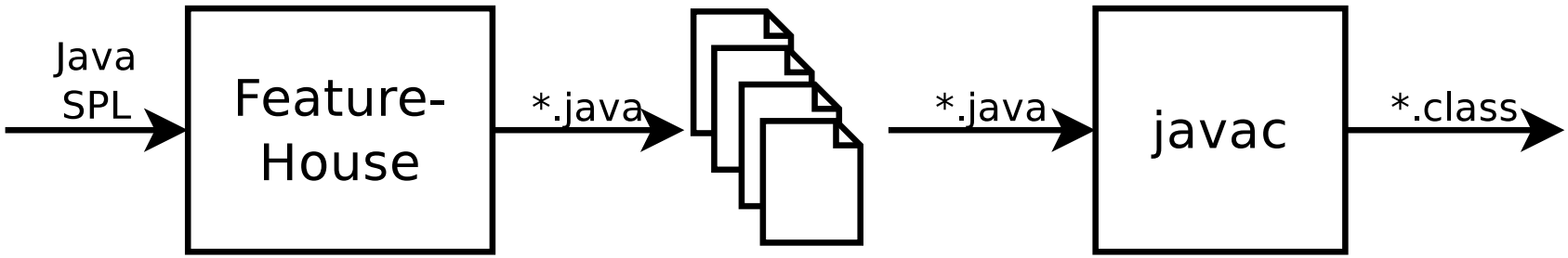
# Motivation

- Bis jetzt: Source-to-Source-Compiler wie AHEAD oder FeatureHouse



# Motivation

- Bis jetzt: Source-to-Source-Compiler wie AHEAD oder FeatureHouse



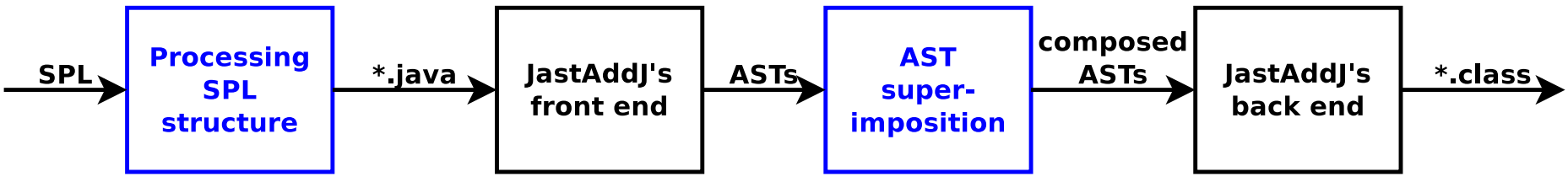

- Nachteile:
  - Kein entwickeltes Typsystem
  - FOP-spezifische Typchecks sind nicht möglich
  - Verschiedene Analysen nur beschränkt möglich
  - Fehlermeldungen beziehen sich auf den Zwischencode

# Fuji Architektur

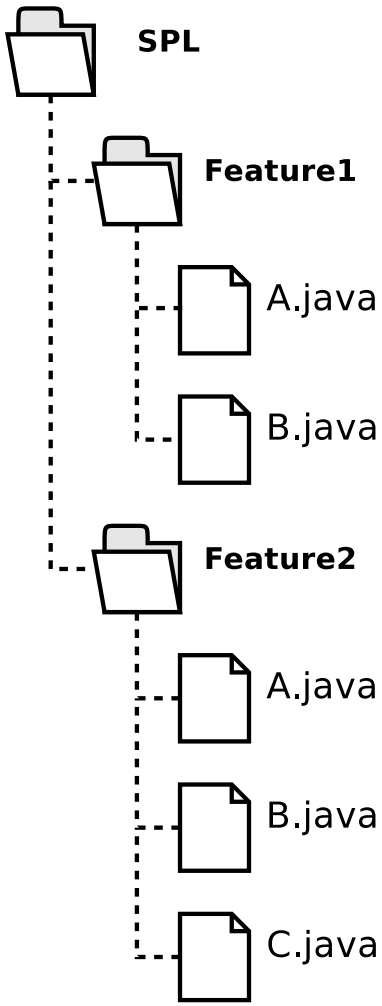
- Fuji basiert auf JastAddJ
- Der Hauptakzent bei JastAddJ liegt auf der Erweiterbarkeit
- Fuji ist eine Erweiterung des JastAddJ-Frontends

# Fuji Architektur

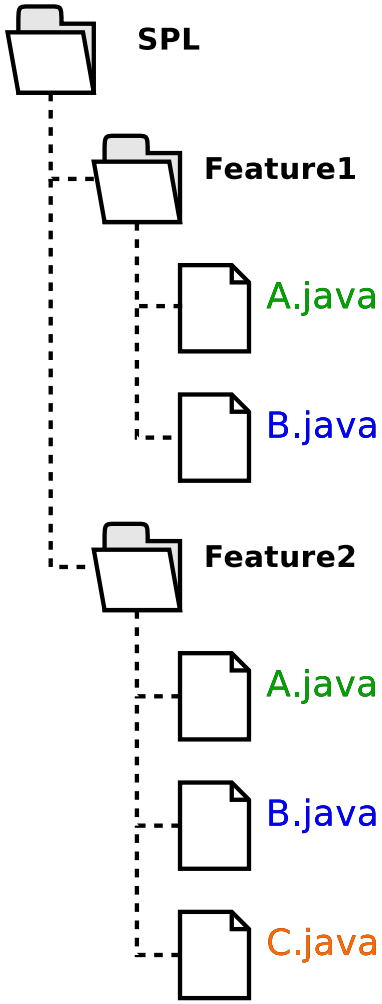
- Fuji basiert auf JastAddJ
- Der Hauptakzent bei JastAddJ liegt auf der Erweiterbarkeit
- Fuji ist eine Erweiterung des JastAddJ-Frontends



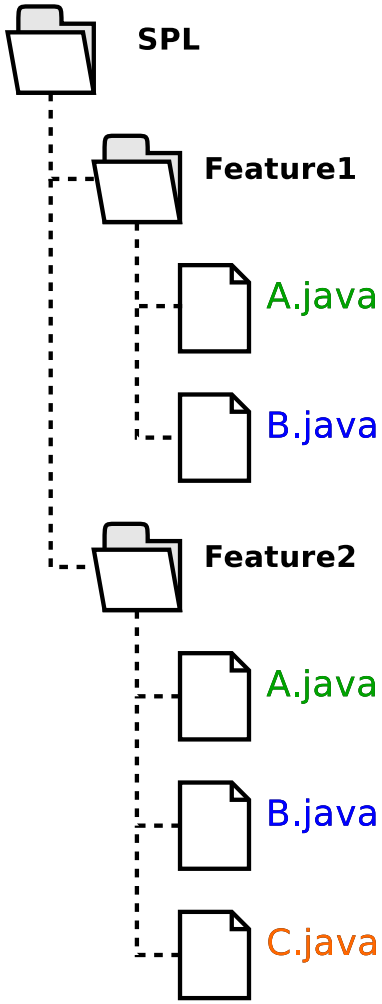
# Kompilierungsschritt: Repräsentation von SPL Struktur



# Kompilierungsschritt: Repräsentation von SPL Struktur



# Kompilierungsschritt: Repräsentation von SPL Struktur

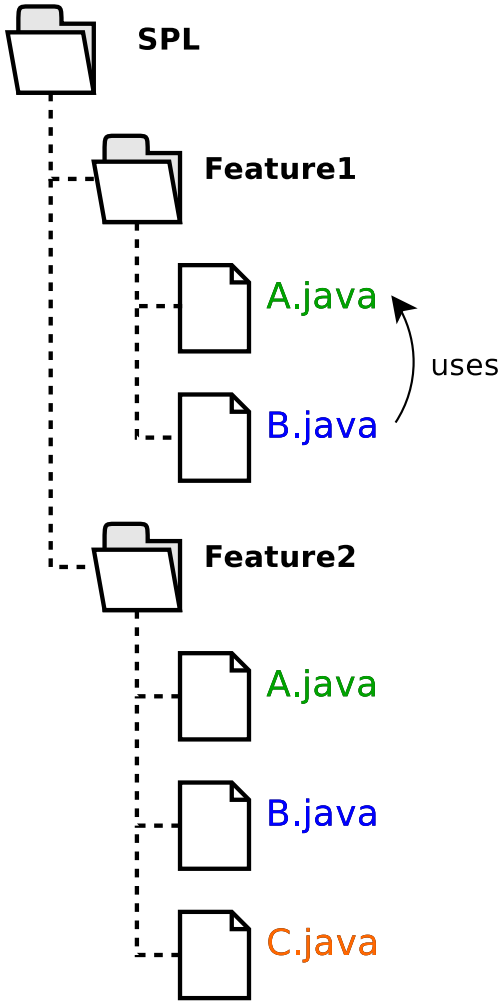



**A : Feature1/A.java Feature2/A.java**

**B : Feature1/B.java Feature2/B.java**

**C : Feature2/C.java**

# Kompilierungsschritt: Repräsentation von SPL Struktur

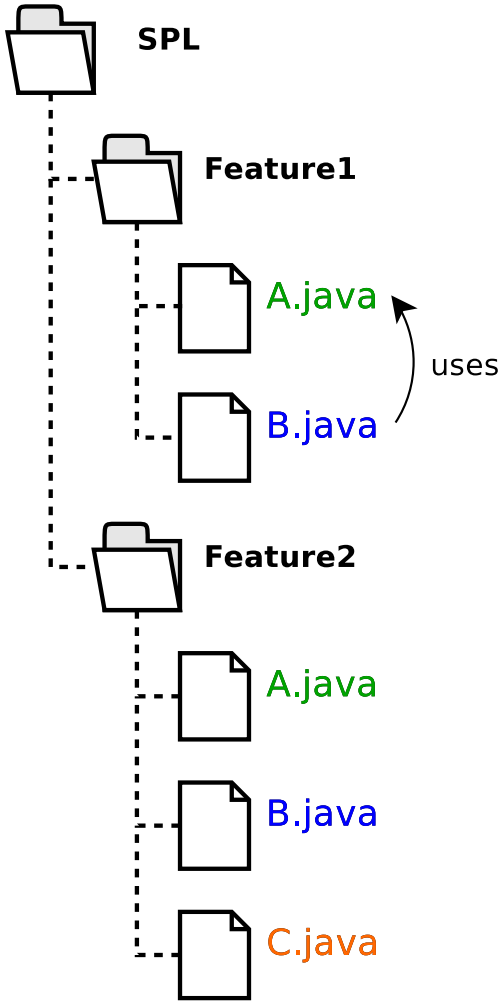



**A : Feature1/A.java Feature2/A.java**

**B : Feature1/B.java Feature2/B.java**

**C : Feature2/C.java**

# Kompilierungsschritt: Repräsentation von SPL Struktur

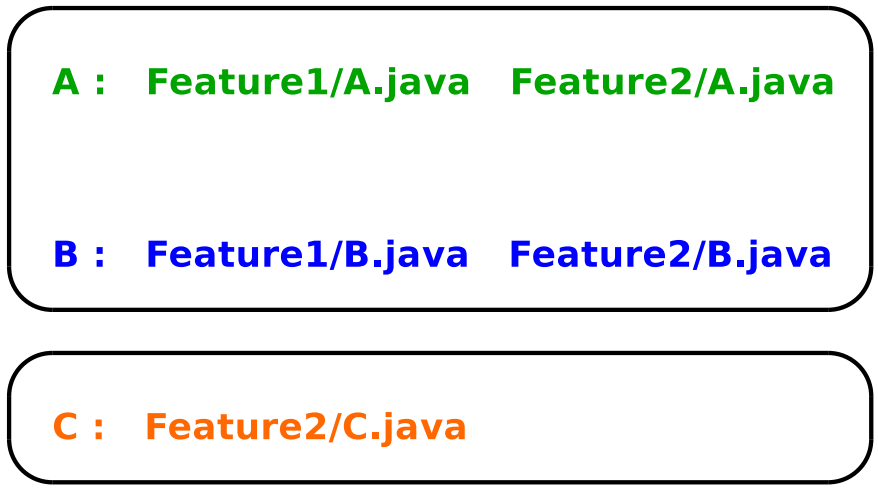



**A :** **Feature1/A.java** **Feature2/A.java**

**B :** **Feature1/B.java** **Feature2/B.java**

**C :** **Feature2/C.java**

# Kompilierungsschritt: Repräsentation von SPL Struktur

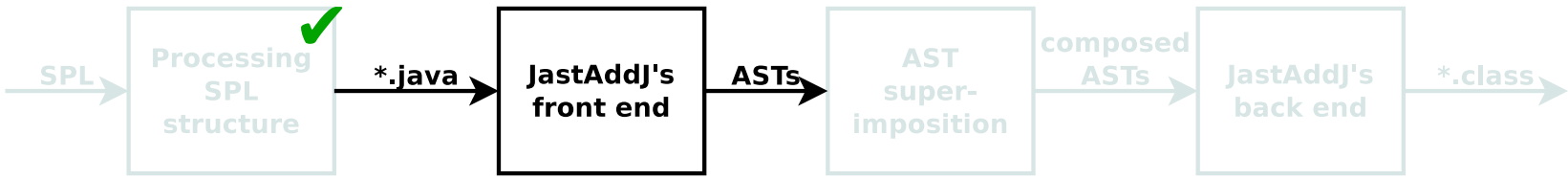


# Kompilierungsschritt: Repräsentation von SPL Struktur

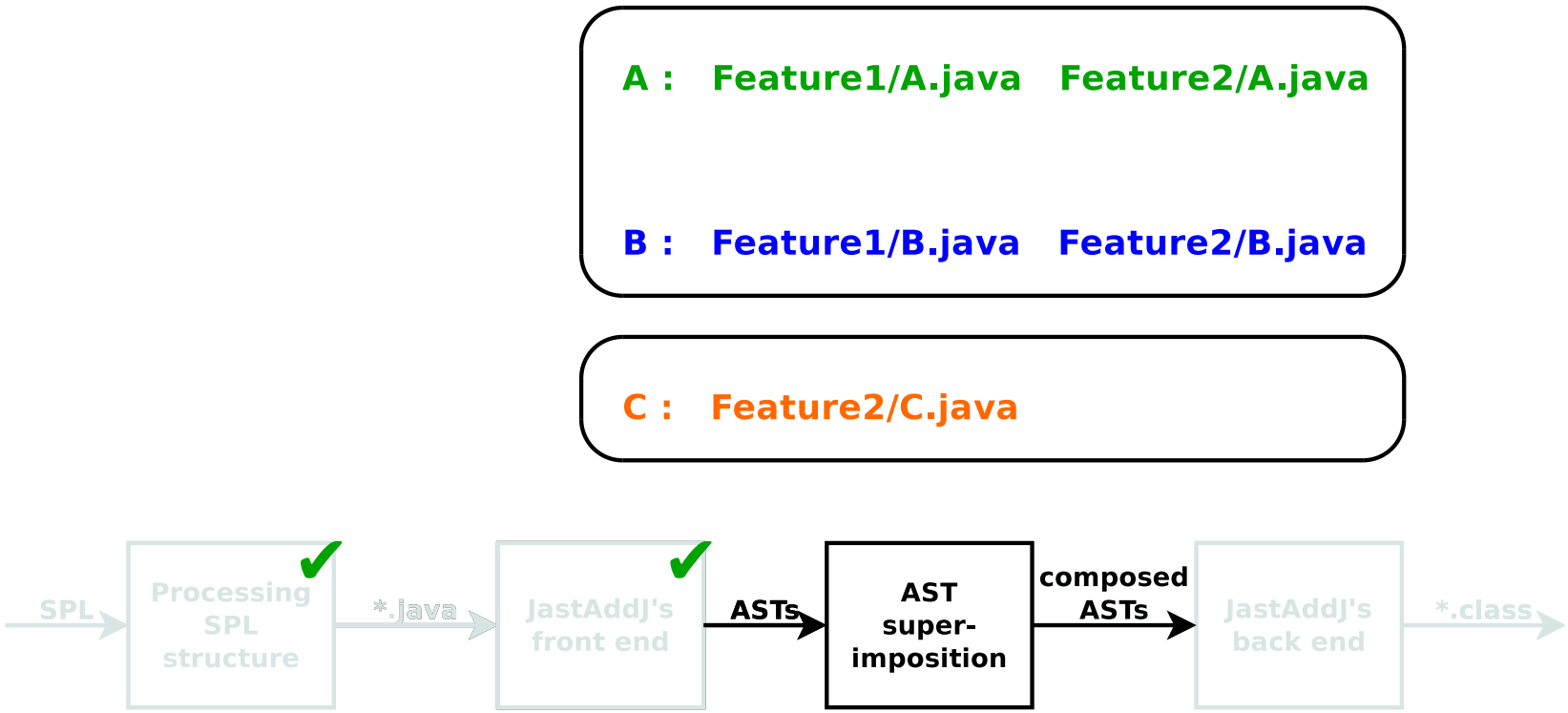
**A :** Feature1/A.java Feature2/A.java

**B :** Feature1/B.java Feature2/B.java

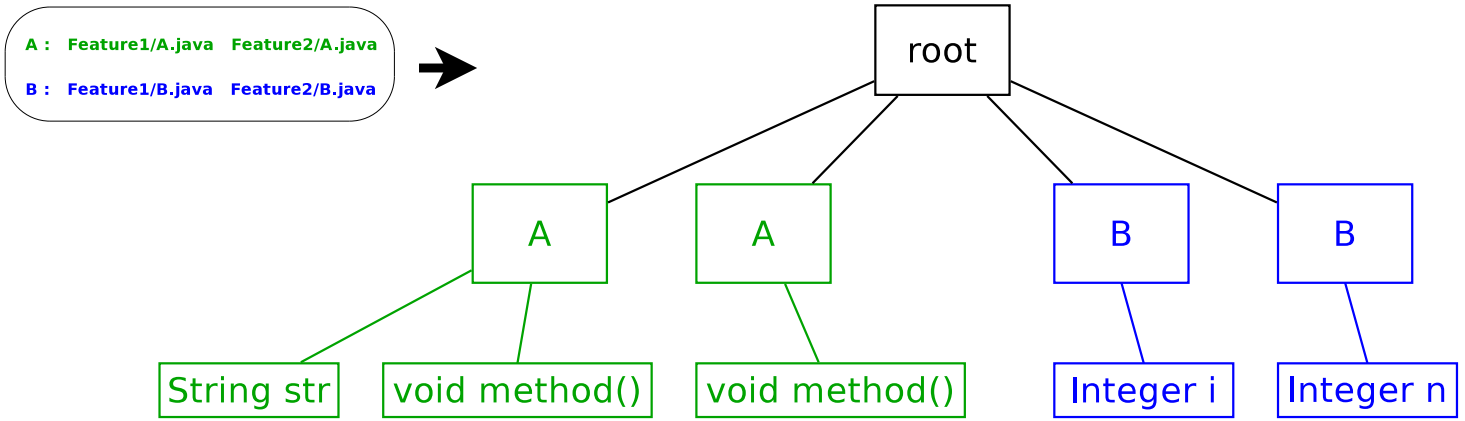
**C :** Feature2/C.java



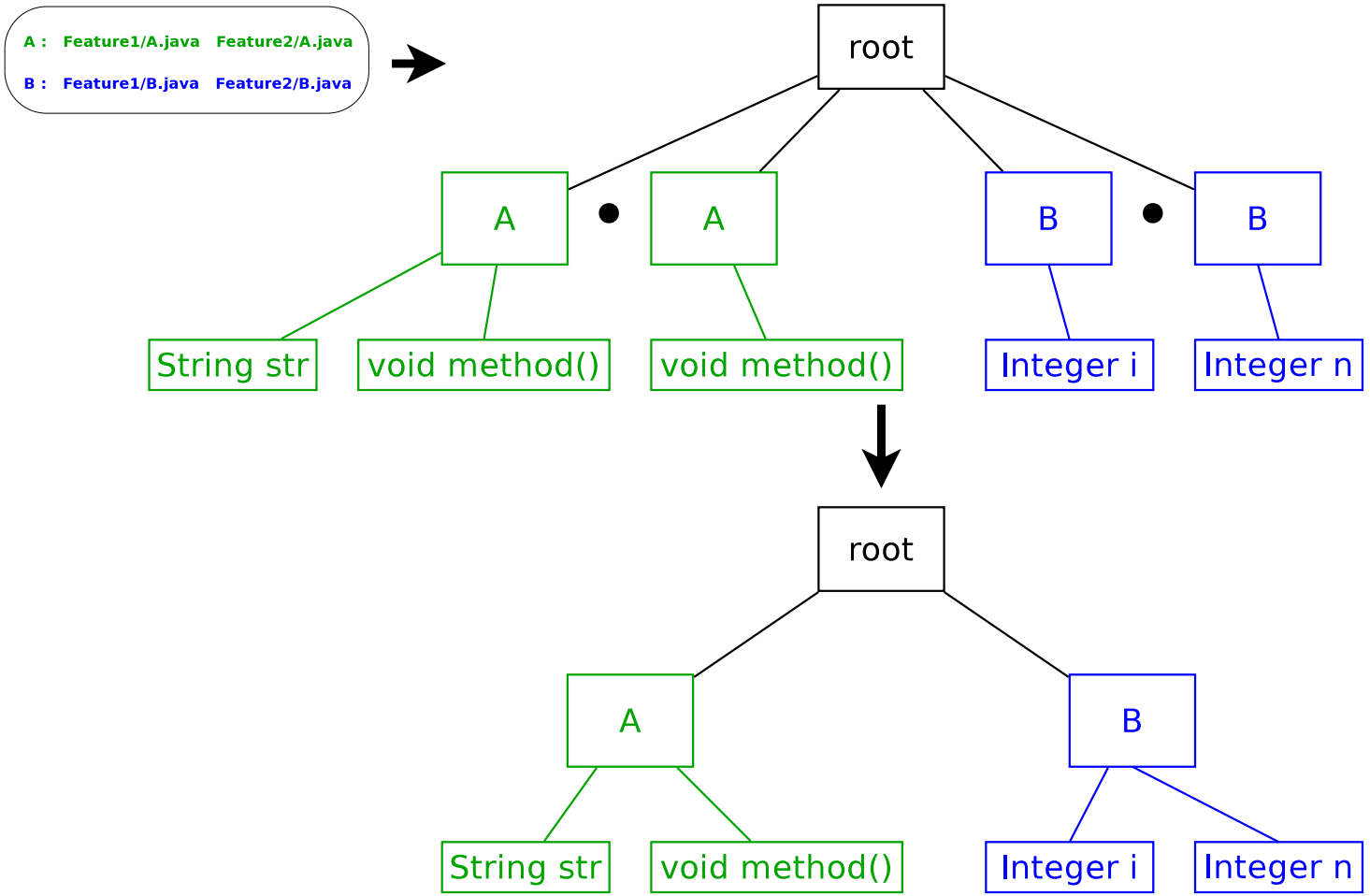
# Kompilierungsschritt: Repräsentation von SPL Struktur



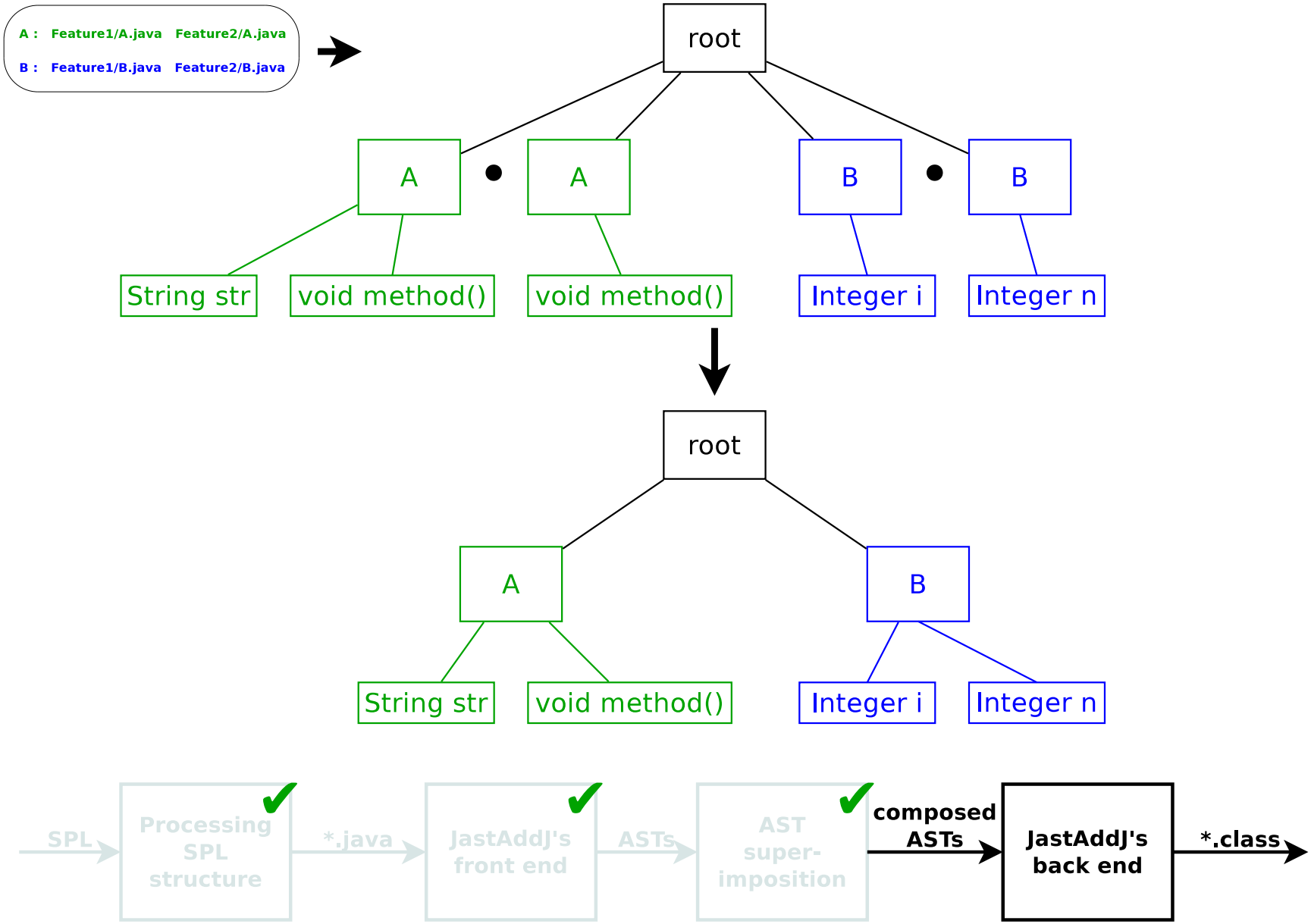
# Kompilierungsschritt: Superimposition von ASTs



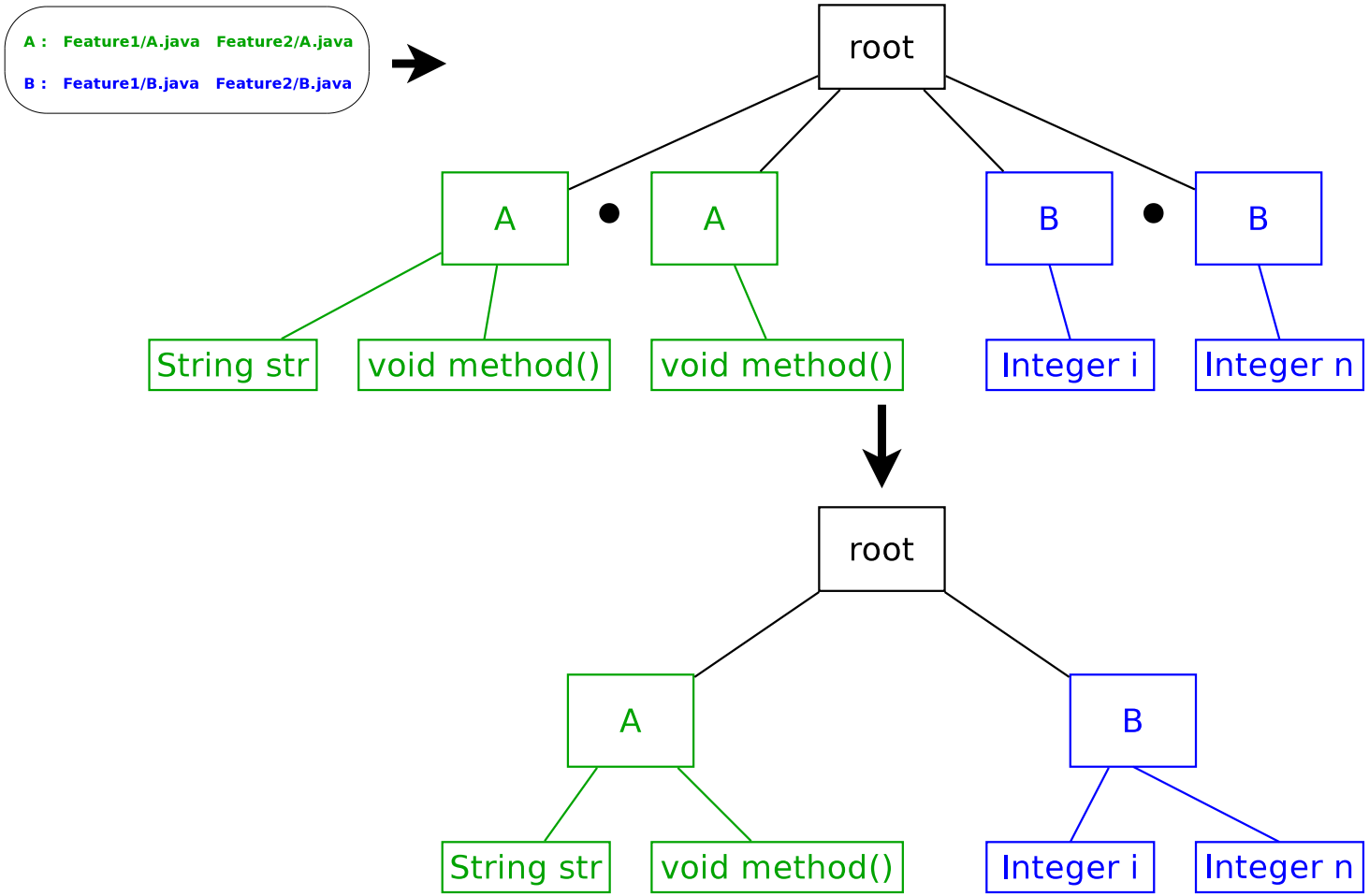
# Kompilierungsschritt: Superimposition von ASTs



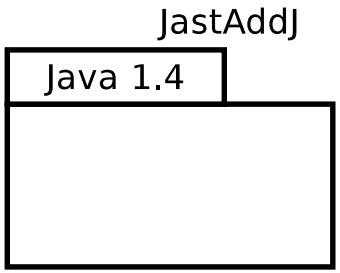
# Kompilierungsschritt: Superimposition von ASTs



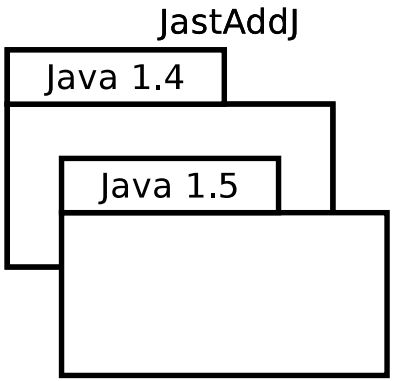
# Kompilierungsschritt: Superimposition von ASTs



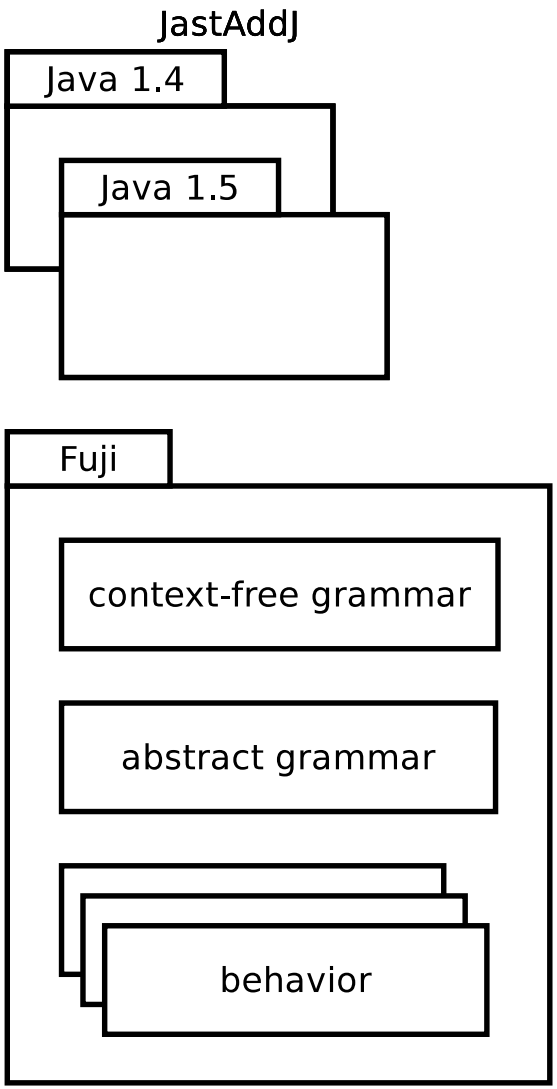

# Erweiterung von Fuji (JastAdd Workflow)



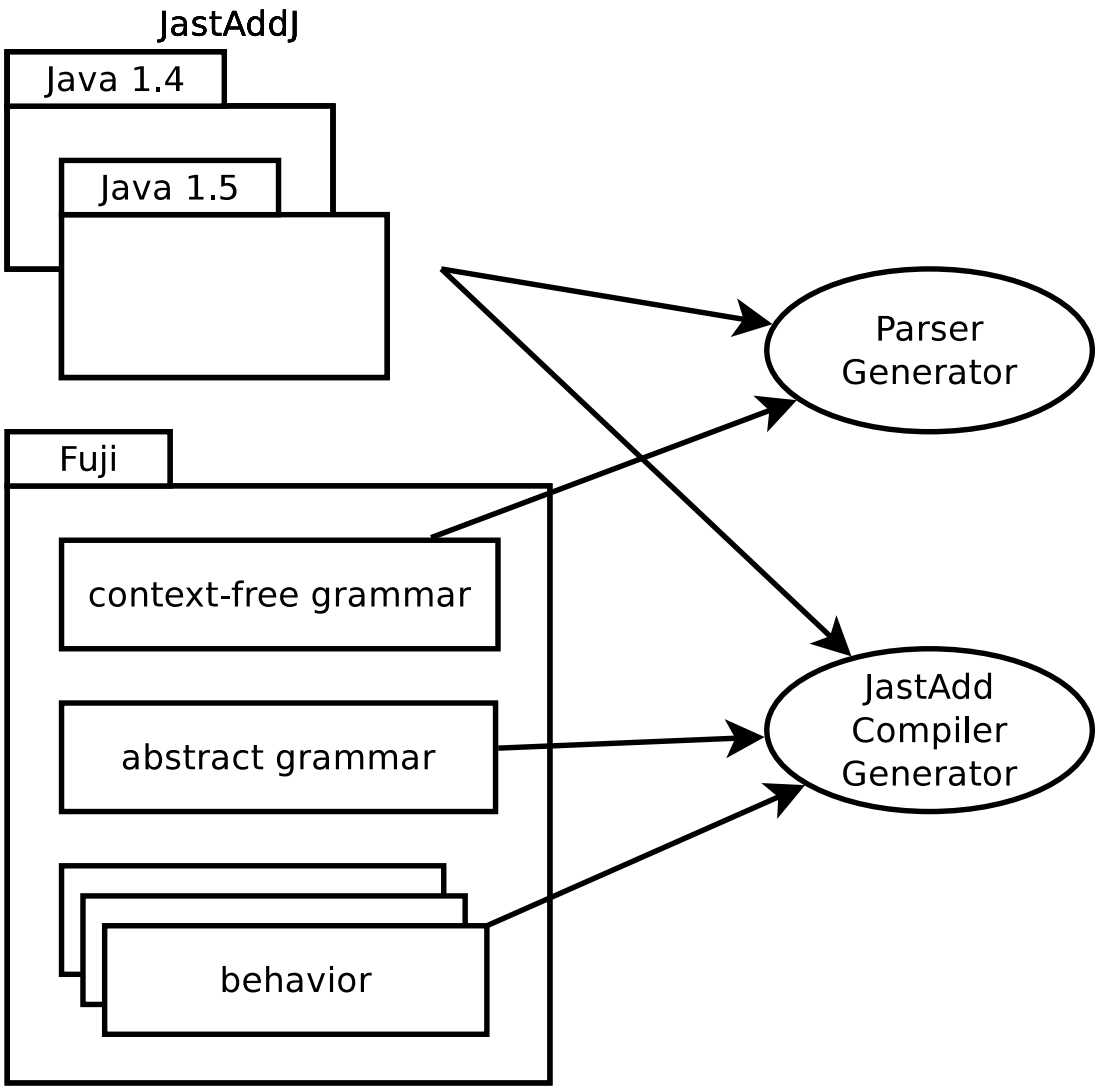
# Erweiterung von Fuji (JastAdd Workflow)



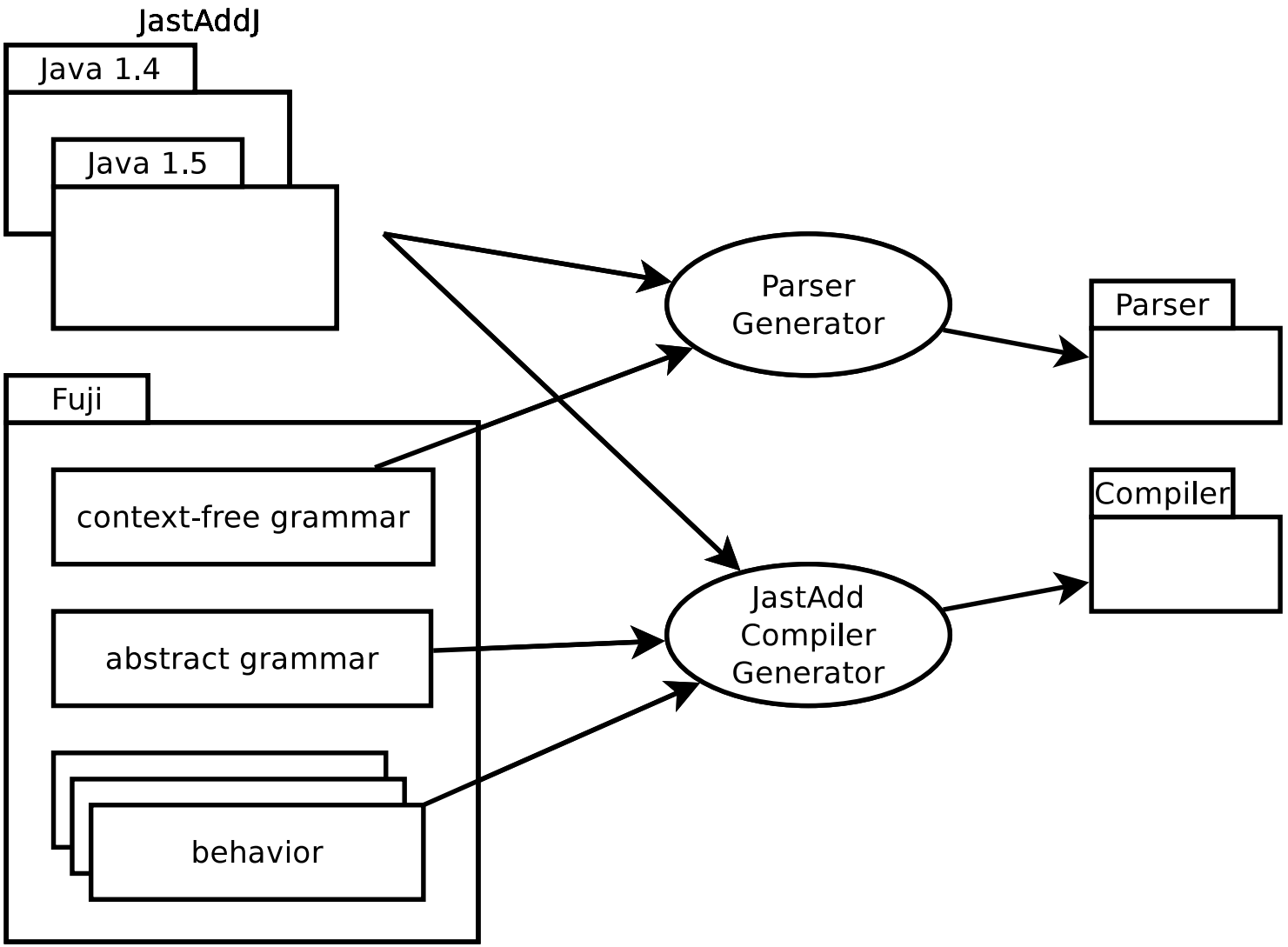
# Erweiterung von Fuji (JastAdd Workflow)



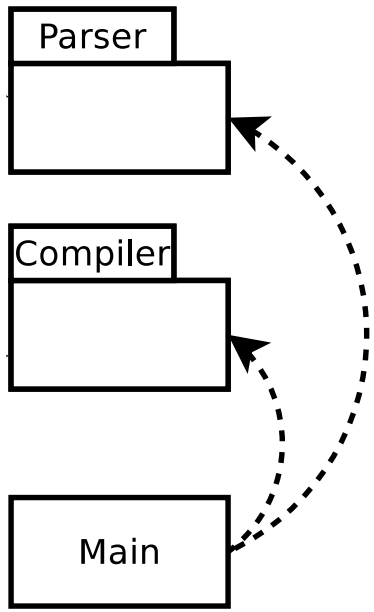
# Erweiterung von Fuji (JastAdd Workflow)



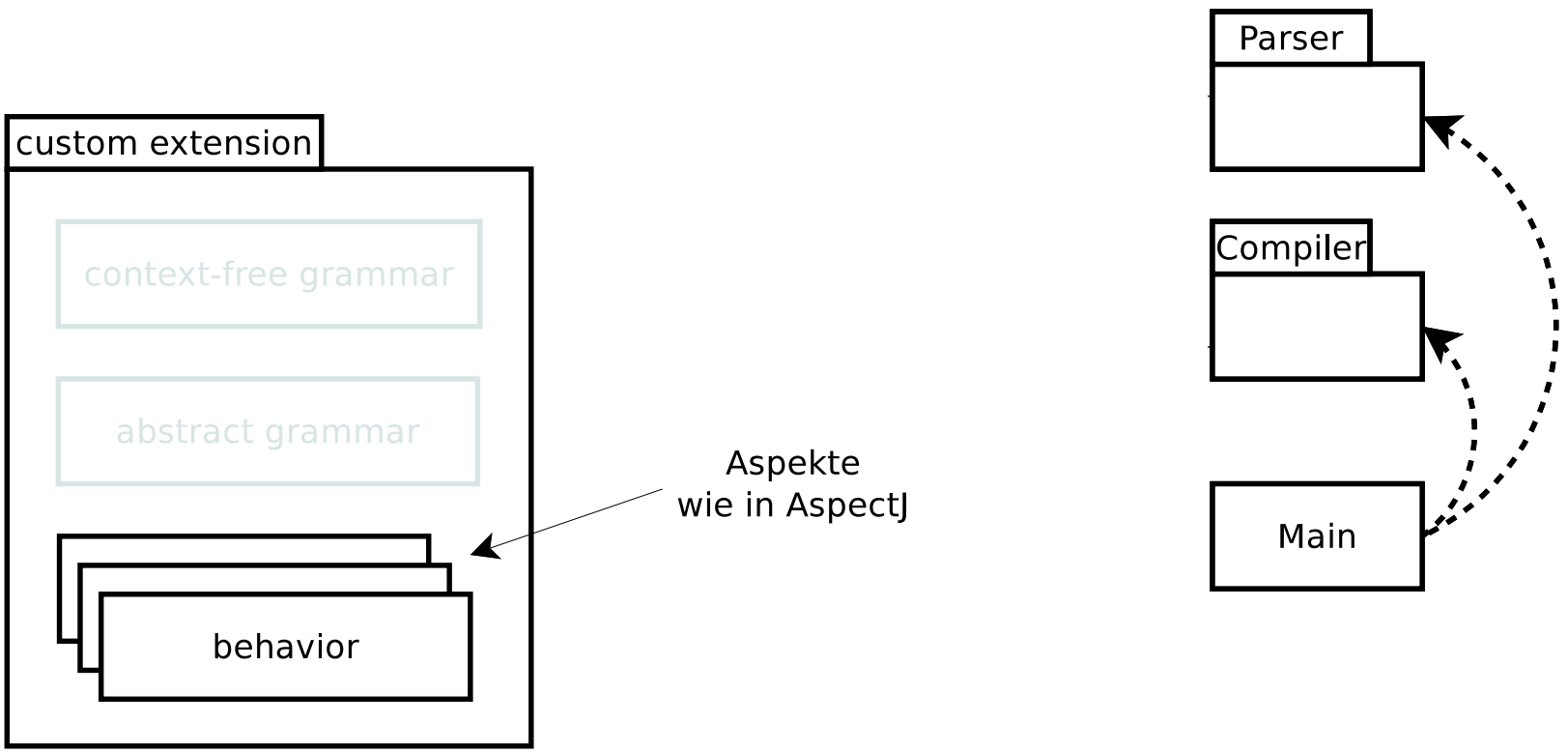
# Erweiterung von Fuji (JastAdd Workflow)



# Erweiterung von Fuji (JastAdd Workflow)

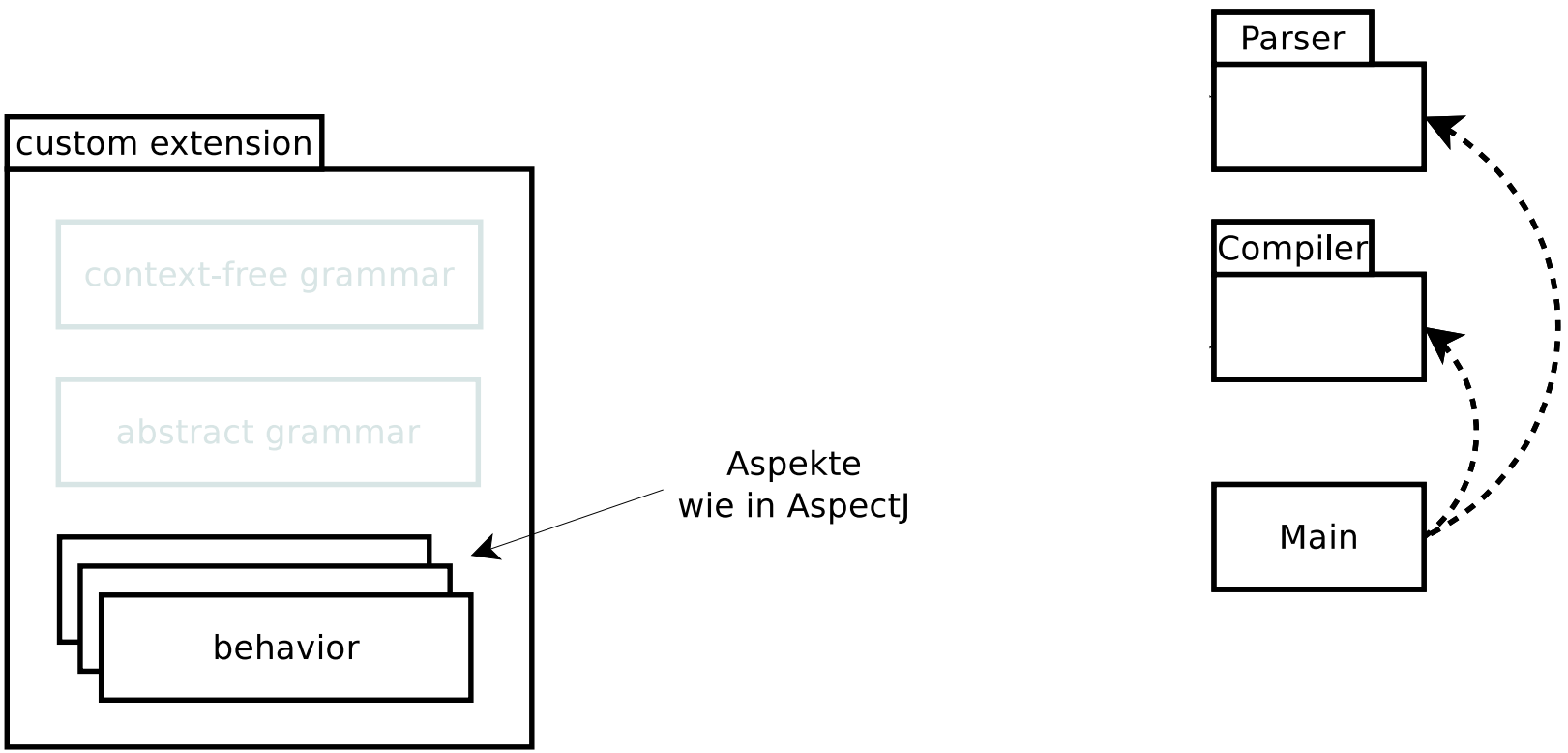


# Erweiterung von Fuji (JastAdd Workflow)



# Erweiterung von Fuji (JastAdd Workflow)

http://fosd.de/fuji

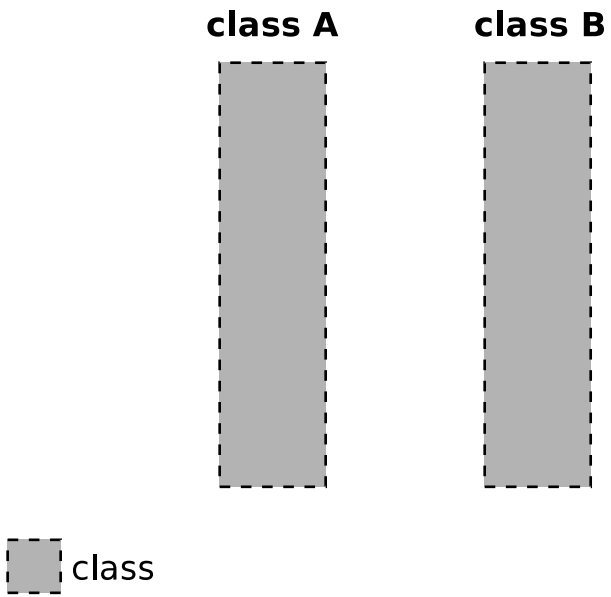


# Fuji. Einige Eigenschaften

- Erweiterbar
- FOP-Modifiers
- FOP-Analysen
- Superimopsitionsalgorithmus ist leicht austauschbar
- Source-to-Source Übersetzung mit Typechecks

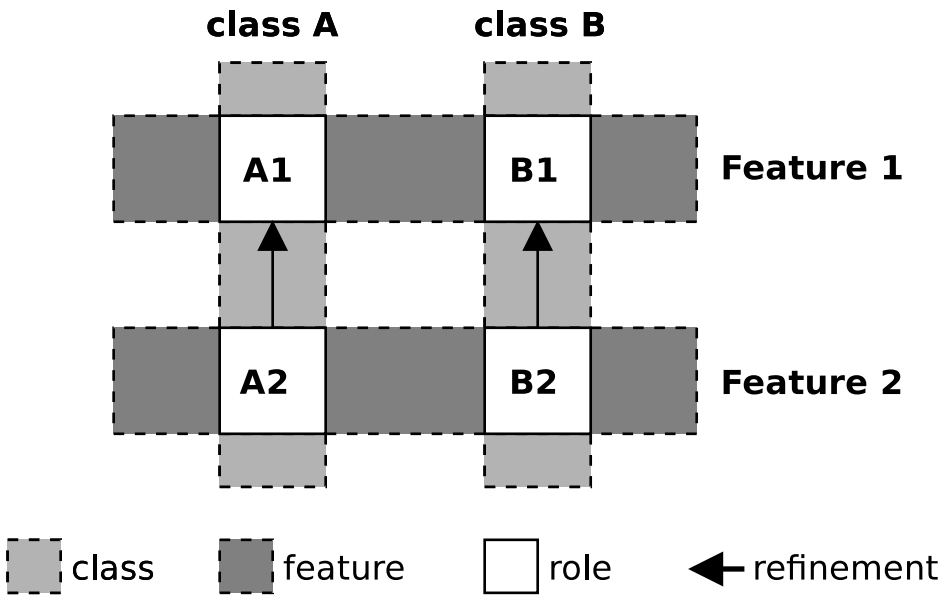
# FOP-Modifiers

- OOP-Modifiers: `private` `(default)` `public`



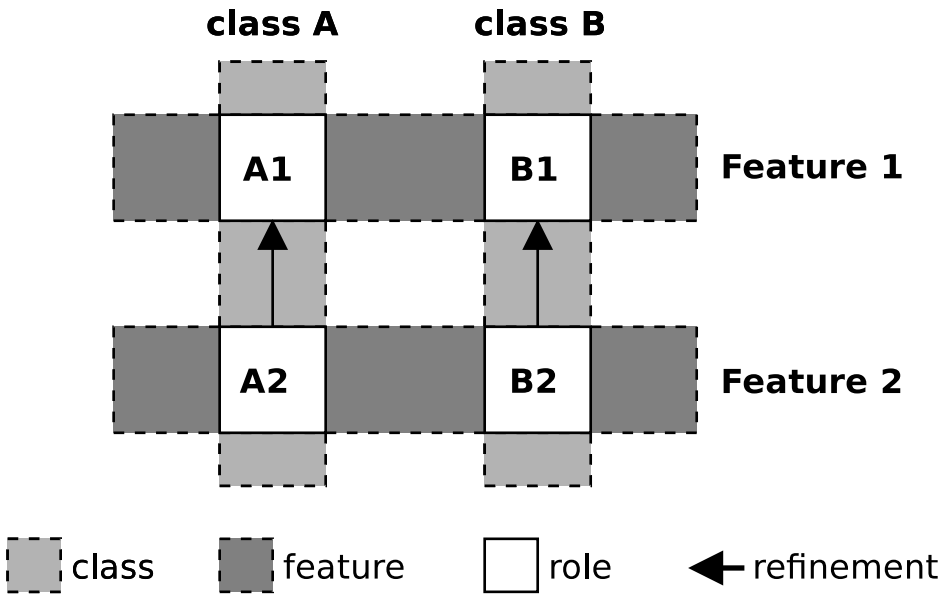
# FOP-Modifiers

- OOP-Modifiers: `private` `(default)` `public`



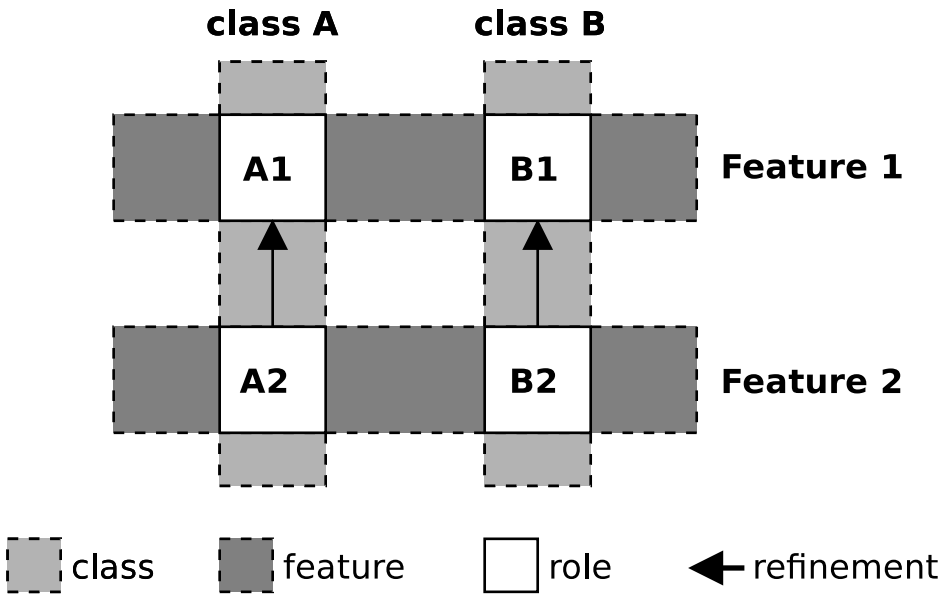
# FOP-Modifiers

- OOP-Modifiers: `private` `(default)` `public`
- FOP-Modifiers: `feature` `subsequent` `program`



# FOP-Modifiers

- OOP-Modifiers: `private` `(default)` `public`
- FOP-Modifiers: `feature` `subsequent` `program`




- Erweiterungen an:
  - Syntax
  - Typsystem

# FOP-Modifiers. Analyse

- 10 SPLs verschiedener Komplexität und aus verschiedenen Domänen.
- In ∼80% der Fällen konnte die Sichtbarkeit von Members weiter eingeschränkt werden.
- In ∼76% der Fällen konnte die Sichtbarkeit auf `feature` eingeschränkt werden.
- Schlussfolgerung: Es gibt Potential für Datenkapselung auf der Feature-Ebene.

# Evaluation. Example SPLs

| SPL | Domain | #F | LOC | Compilation sec. |
|---|---|---|---|---|
| GUIDSL | SPL configuration tool | 26 | 10084 | 56.54 |
| Violet | UML editor | 88 | 7194 | 20.18 |
| Prevayler | object persistence lib. | 6 | 5268 | 18.62 |
| PKJab | Jabber client | 8 | 3373 | 12.53 |
| ZipMe | ZIP library | 13 | 3520 | 07.09 |
| TankWar | strategic game | 15 | 2830 | 07.34 |
| AJStats | software analysis tool | 20 | 13226 | 05.78 |
| Notepad | text editor | 10 | 891 | 05.59 |
| GPL | graph library | 9 | 646 | 04.53 |
| EPL | expression evaluation | 11 | 105 | 01.36 |

# Zu meinem Forschungsgebiet

- SPL Group, Uni. Passau
- Nicht-Funktionale Eigenschaften von SPLs:
  - Performanz
  - Speicherbedarf
  - Energiebedarf
- Ziel: NF-Eigenschaften einzelner Produkte möglichst genau vorauszusagen.

# Zu meinem Forschungsgebiet

- SPL Group, Uni. Passau
- Nicht-Funktionale Eigenschaften von SPLs:
  - Performanz
  - Speicherbedarf
  - Energiebedarf
- Ziel: NF-Eigenschaften einzelner Produkte möglichst genau vorauszusagen.

- **TU Dortmund:** Olaf Spinczyk, Matthias Meier...
- **Friedrich-Alexander Uni. Erlangen:** Julio Sincero, Reinhard Tartler, Daniel Lohmann...
- **Uni. Magdeburg:** Norbert Siegmund...
- **Philipps-Uni. Marburg:** Christian Kästner...


Fuji

# Aspekte

AspectFoo.jrag:

```
1  aspect AspectFoo {
2
3    /* Does the compilation units represent a role? */
4    public boolean Program.isRole(CompilationUnit cu) {
5      return roles.contains(cu);
6    }
7
8    /* Is the current modifier an FOP 'feature' modifier? */
9    syn lazy boolean Modifiers.isFeature() =
10     numModifier("feature") != 0;
11
12   /* Semantic check: how many FOP modifiers are specified?*/
13   refine Modifiers void Modifiers.checkModifiers() {
14     refined();
15     if(numFOPModifiers() > 1) error("Too many FOP modifiers!");
16   }
17
18   /* Class definition. */
19   public class ComposeVisitor { /* ... */ }
20 }
```