UNIKASSEL
VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

# Software Tool Construction

Wintersemester 2023/24

Adrian Kunz

Vorlesung 2

# GitHub Classroom

https://classroom.github.com/a/aZr124gT

# ANTLR

```
@parser::header { ... }

file returns [ast.CompilationUnit cu]:
  (classes+=class)* EOF { $cu = new ast.CompilationUnit(this._input.sourceName, $classes.map(c ⇒ c.cn)); }
;

class returns [ast.Class cn]:
  DOC? 'class' ID '{' (fields+=field | constructors+=ctor | methods+=method)* '}' {
    $cn = new ast.Class($ID.text!, $fields.map(f ⇒ f.fn), $constructors.map(c ⇒ c.cn), $methods.map(m ⇒ m.mn))
    $cn.location = makeRange($ID);
    $cn.doc = cleanDoc($DOC);
  }
;
field returns [ast.Field fn]:
  DOC? 'var' ID ':' type ('=' expression)? ';'? {
    $fn = new ast.Field($ID.text!, $type.tn, $expression.e)
    $fn.location = makeRange($ID);
    $fn.doc = cleanDoc($DOC);
  }
;
ctor returns [ast.Constructor cn]:
  DOC? init='init' '(' (parameters+=parameter ','?)* ')' blockStatement {
    $cn = new ast.Constructor($parameters.map(p ⇒ p.pn), $blockStatement.bs)
    $cn.location = makeRange($init);
    $cn.doc = cleanDoc($DOC);
  }
;
method returns [ast.Method mn]:
  DOC? 'func' ID '(' (parameters+=parameter ','?)* ')' ':' type blockStatement {
    $mn = new ast.Method($ID.text!, $parameters.map(p ⇒ p.pn), $type.tn, $blockStatement.bs)
    $mn.location = makeRange($ID);
    $mn.doc = cleanDoc($DOC);
  }
;
```

# ANTLR – Grundlagen

- ANTLR – Another Tool for Language Recognition [1]
- ALL(*)-Parser-Generator
  - Adaptive [2]
  - Left to Right
  - Leftmost derivation
  - (*) Beliebiges Lookahead (=> deterministisch kontextfreie Sprachen)
  - Generator: Eingabe Grammatik, Ausgabe Parser
- Generiert C++, C#, Dart, Java, JavaScript, PHP, Python3, Swift, TypeScript

[1]: https://github.com/antlr/antlr4
[2]: Parr et al.: *Adaptive LL(*) Parsing: The Power of Dynamic Analysis* https://www.antlr.org/papers/allstar-techreport.pdf

# ANTLR – Syntax

```
grammar Expr;

// Parser-Regeln
prog: expr EOF ;
expr: expr ('*'|'/') expr
    | expr ('+'|'-') expr
    | INT
    | '(' expr ')' ;

// Lexer-Regeln
NEWLINE : [\r\n]+ -> skip;
INT : [0-9]+ ;
```

https://www.antlr.org/

# ANTLR – Lexer

```
// Charakter-Klassen wie bei RegEx
CharClass1 : [a-z] ;
// Alternativ: Charakter-Range
CharClass2 : 'a'..'z' ;
// Negierung: Alle anderen Zeichen
Negation : ~[abc] ;
// . matcht auch \n, aber *? verhindert das
NonGreedy : '//' .*? '\n' ;
```

Koch, Andreas: *Compiler 1: Grundlagen* https://www.esa.informatik.tu-darmstadt.de/archive/twiki/pub/Lectures/Compiler115De/antlr-v4-slides.pdf

# ANTLR – Parser

```
// Alternativen können #Labels haben
rulename : alternative1 # label
// Ohne Operator -> Konkatenation
| anotherParserRule LEXERRULE 'literal' # concat
// (|) für geschachtelte Alternativen
| A 'x' (B | C) # subrule
// * = 0..n, + = 1..n, ? = 0..1
| D* E+ F? (G H | I)+ # repeat
;
```

Koch, Andreas: *Compiler 1: Grundlagen* https://www.esa.informatik.tu-darmstadt.de/archive/twiki/pub/Lectures/Compiler115De/antlr-v4-slides.pdf

# ANTLR – Sonstiges

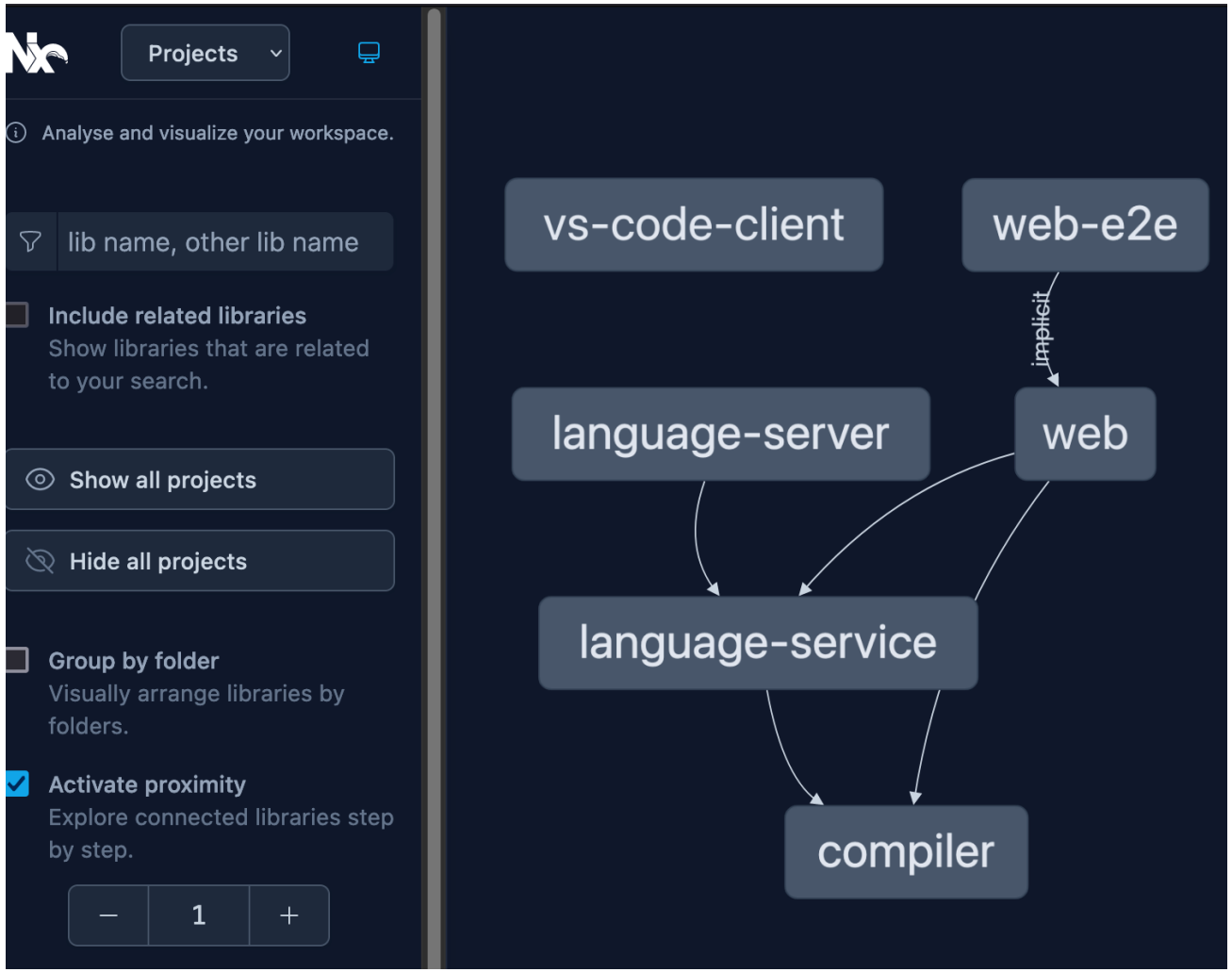
```
grammar Expr;

prog: expr EOF { console.log($expr.v) };

expr returns [int v]
    : lhs=expr '^'<assoc=right> rhs=expr      { $v = $lhs.v ** $rhs.v; }
    | lhs=expr '*' rhs=expr                   { $v = $lhs.v * $rhs.v; }
    | lhs=expr '+' rhs=expr                   { $v = $lhs.v + $rhs.v; }
    | INT                                     { $v = +$INT.text; }
    | '(' expr ')'                            { $v = $expr.v; }
    | '[' elems+=expr (',' elems+=expr)* ']' { $v = $elems.map(e => e.v); }
;

NEWLINE : [\r\n]+ -> skip;
INT : DIGIT+ ;
fragment DIGIT: [0-9];
```

# Nx


Projects
Analyse and visualize your workspace.
lib name, other lib name
Include related libraries
Show libraries that are related to your search.
Show all projects
Hide all projects
Group by folder
Visually arrange libraries by folders.
Activate proximity
Explore connected libraries step by step.
1
vs-code-client
web-e2e
implicit
language-server
web
language-service
compiler