UNIVERSITÄT
DES
SAARLANDES

Programmierung 2 - SS19

Projekt 4 - URI Parser

Gideon Geier, Dominik Kempter, Pascal Lauer

05. Juni 2019

Universität des Saarlandes

Technische Hinweise

Git Projekt-Repository

Wir können das Projekt mit `git clone` unter folgender URL beziehen:

```
https://prog2scm.cdl.uni-saarland.de/git/project4/<NAME>
```

<NAME> = Euer Benutzername auf der Prog2-Website

Achtung!

Die Repositories sind nur innerhalb des Uninetzes erreichbar. Von außerhalb kann man eine VPN-Verbindung zum Uninetz einrichten.

Eine Anleitung steht auf der Website unter Software.

Ab auf euren Rechner

Was braucht ihr?

- Git
- Open JDK
- Eclipse (oder IntelliJ)

Windows-Nutzer: Drauf klicken. Runter laden. Installieren.
Unix-Nutzer: Nutzt für Git und Open JDK besser euren Paket-Manager.

Javadoc erstellen

Ins `src`-Verzeichnis wechseln und das folgende Kommando ausführen:

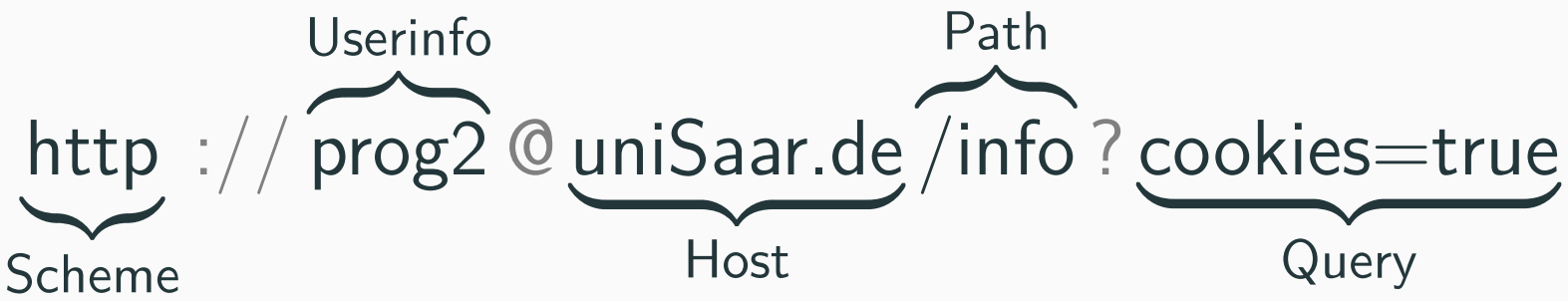
```
javadoc -d ../jdoc uri
```

Test-Deadline schon am Di 11.06.2019 23:59 Uhr

(Ja, das bedeutet früher anfangen.)

Was ist zu tun?

Uris „zerstückeln“



Grammatiken

Die gegebene Grammatik

```
URI           = scheme ":" hierarchical [ "?" query ]

hierarchical  = "//" authority path

scheme        = ALPHA *( ALPHA / DIGIT )

authority     = [ userinfo "@" ] host
userinfo      = *( pchar / ":" )
host          = IPv4address / reg-name

IPv4address   = dec-octet "." dec-octet "." dec-octet "." dec-octet

dec-octet     = ["0" ["0"]] DIGIT      ; 000-009 with optional leading zeros
              / ["0"] "1"-"9" DIGIT    ; 010-099
              / "1" DIGIT DIGIT        ; 100-199
              / "2" "0"-"4" DIGIT      ; 200-249
              / "25" "0"-"5"           ; 250-255

reg-name      = *pchar

path          = *( "/" *pchar )        ; begins with "/" or is empty

query         = *( pchar / "&" / "=" )

pchar         = unreserved / pct-encoded
unreserved    = ALPHA / DIGIT / "."
pct-encoded   = "%" HEXDIGIT HEXDIGIT

ALPHA         = "A"-"Z" / "a"-"z"
DIGIT         = "0"-"9"
HEXDIGIT      = DIGIT / "A"-"F" / "a"-"f"
```

Wie lese ich das? (1/2)

- `"…"` **Literale**

  z.B.: `"B"` $\rightsquigarrow$ `B` , `"Affe"` $\rightsquigarrow$ `Affe`, `"BAnAnE"` $\rightsquigarrow$ `BAnAnE`

- `… ␣ …` **Verkettung**

  z.B.: `"Affe"␣"BAnAnE"` $\rightsquigarrow$ `AffeBAnAnE`

- `… / …` **Auswahl** (oder)

  z.B.: `"Affe" / "BAnAnE"` $\rightsquigarrow$ `Affe`,

  . `"Affe" / "BAnAnE"` $\rightsquigarrow$ `BAnAnE`

`A` $\rightsquigarrow$ `W` bedeutet Ausdruck A akzeptiert Wort W.

Wie lese ich das? (2/2)

- `*(…)` **Wiederholung** (mehr davon)
  z.B.: `*("B")` $\rightsquigarrow$, `*("B")` $\rightsquigarrow$ `B`, `*("B")` $\rightsquigarrow$ `BB`, …
- `[…]` **optional**
  z.B.: `["Affe"]` $\rightsquigarrow$, `["Affe"]` $\rightsquigarrow$ `Affe`
- `;…` **Kommentare**
  In der Zeile wird alles nach dem Semikolon ignoriert.

`A` $\rightsquigarrow$ `W` bedeutet Ausdruck A akzeptiert Wort W.

Trennzeichen beachten!

- zu erkennende Substrings
- Trennzeichen
- Trennzeichen, gehört aber zum Path

null-optional: Markierter Teil muss nicht vorkommen.

Wenn nicht vorhanden soll null returnt werden.

""-optional: Markierter Teil muss nicht vorkommen.

Wenn nicht vorhanden soll der leere String returnt werden.

Beispiele

https://cms.sic.saarland/prog2/ ✓

prog2.saarland ✗

Hat kein Scheme.

https://uni-saarland.de ✗

„-“ ist kein gültiges Zeichen.

Strings

Umgang mit Uri

- Wie überprüfe ich substrings auf Korrektheit? $\rightarrow$ Grammatik
- Wie bekomme ich z.B. nur die userinfo?
  $\rightarrow$ Methoden auf Strings, Regex

Regular Expressions (Regex)

- Definieren Pattern, welche auf Strings matchen

.	Matcht auf alle Charaktere
abc	Matcht a gefolgt von b und c
[abc]	Matcht entweder a, b oder c
[^abc]	Matcht alles außer a, b und c
^regex	Matcht Regex auf Beginn der Zeile
regex$	Matcht Regex auf Ende der Zeile
\d	Eine Ziffer: [0-9]
\w	Ein Wort-Charakter: [a-zA-Z_0-9]
\s	Ein Whitespace: [\t\n\x0b\r\f]
*	Matcht beliebig oft
?	Matcht 1 oder 0 Mal

Regex - Quiz

Was matcht auf die Regex `"[PGD]"` ?

"Pascal"

"Gideon"

"Dominik"

"{}UriParser"

Regex - Quiz

Was matcht auf die Regex `"[PGD]"` ?

"Pascal" ✓

"Gideon" ✓

"Dominik" ✓

"{}UriParser" ✓

Regex - Quiz

Was matcht auf die Regex "\w*l" ?

"Pascal"

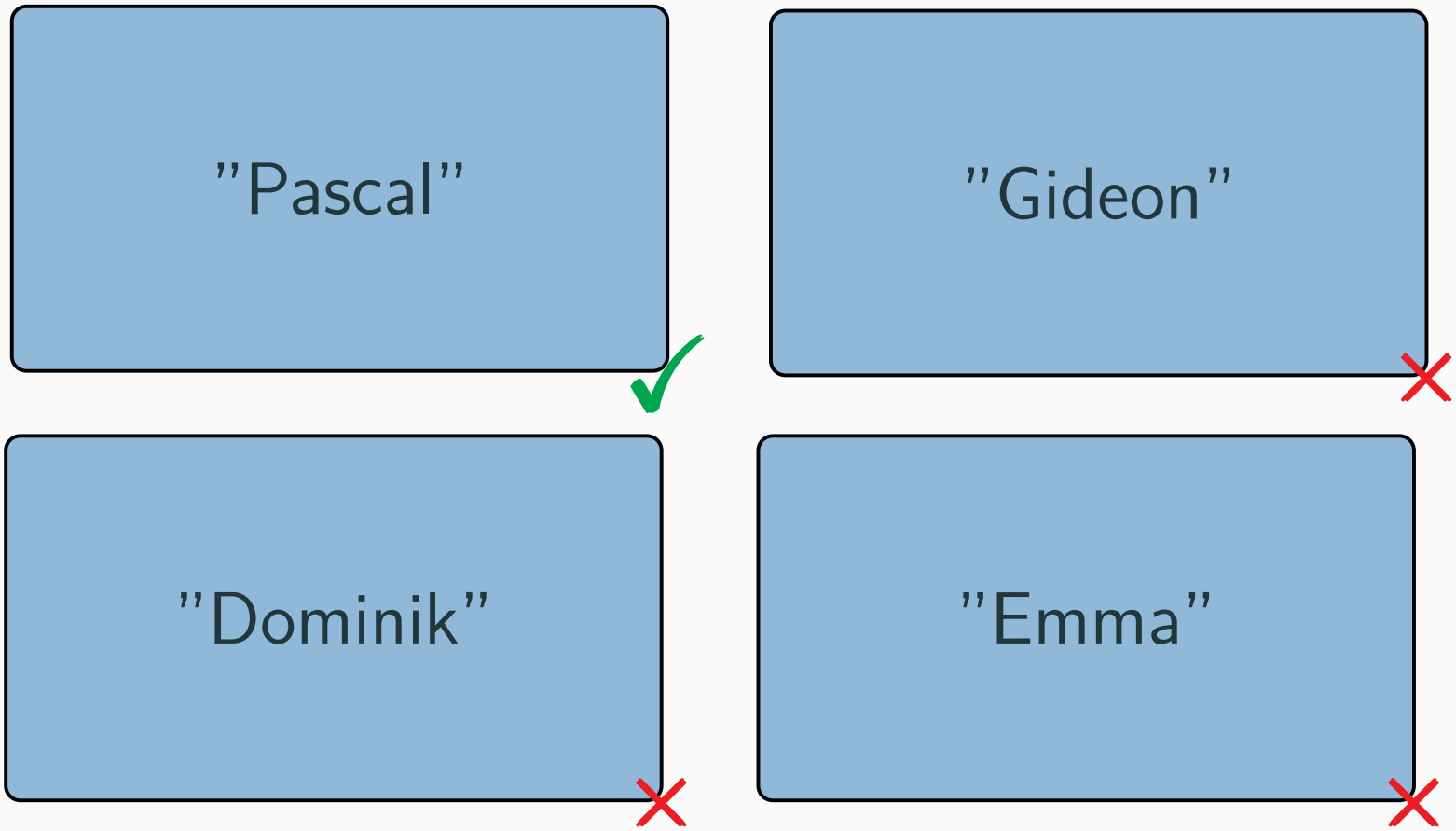
"Gideon"

"Dominik"

"Emma"

Regex - Quiz

Was matcht auf die Regex "\w*l" ?



Regex - Quiz

Was matcht auf die Regex `".*l"` ?

"Pascal"

"Gideon"

"Niklas"

"Jessi"

Regex - Quiz

Was matcht auf die Regex `".*l"` ?

"Pascal" ✓

"Gideon" ✗

"Niklas" ✓

"Jessi" ✗

Regex - Quiz

Was matcht auf die Regex `".*n$"` ?

"Pascal"

"Gideon"

"Dominik"

"Kalli"

Regex - Quiz

Was matcht auf die Regex `".*n$"` ?

Methoden auf Strings mit Regex

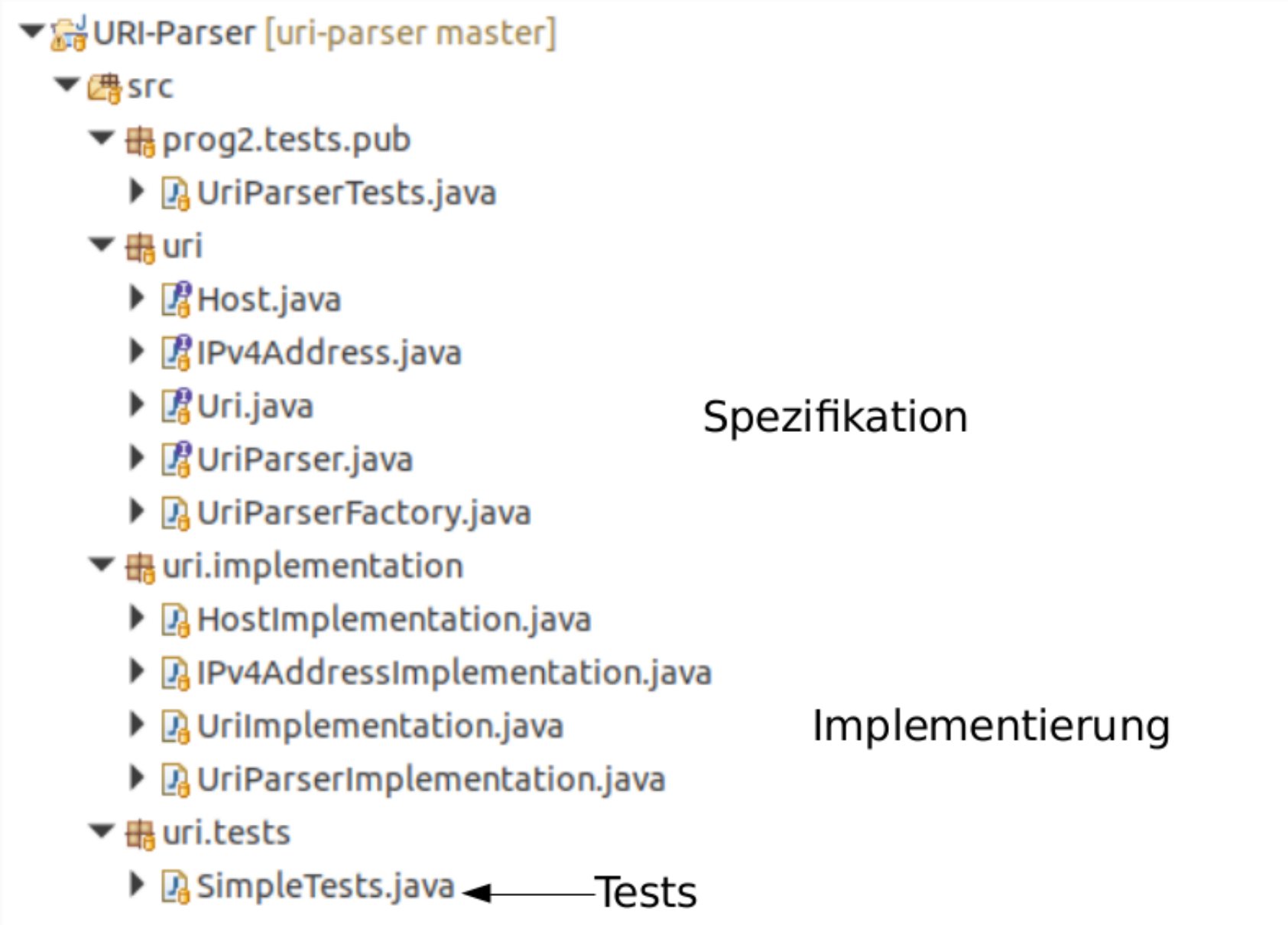
- public String[ ] split(String regex)
  $\rightarrow$ gibt ein durch die Regex getrennten String Array zurück.

```
s = "URI.parsen.macht.Spaß"
s.split("\\."); // ["URI","parsen","macht","Spaß"]
```

- public boolean matches(String regex)

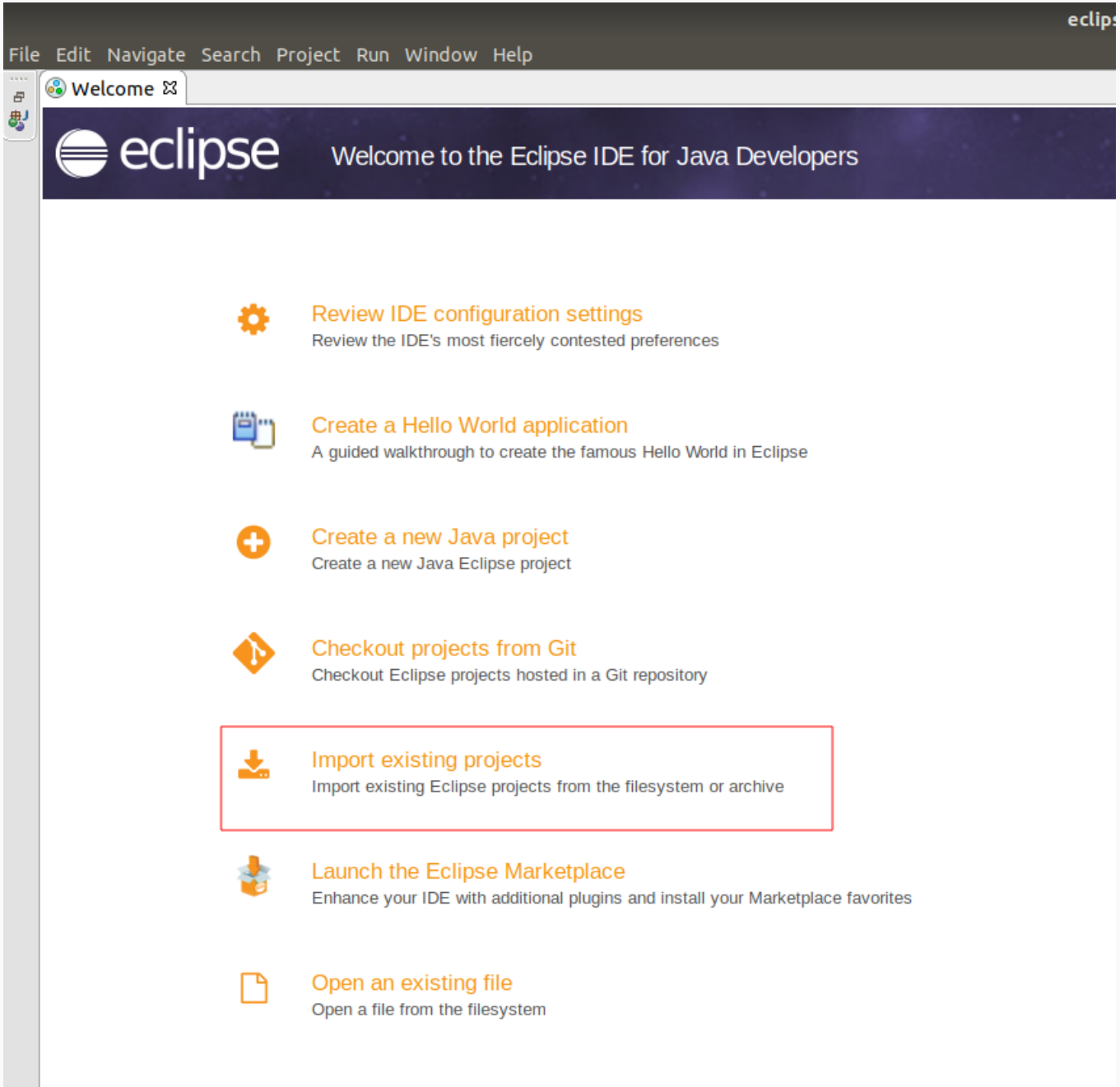
```
s.matches("^Uri"); // false
s.matches("^\\w*[^\\.].*B*a(nana)*.$"); // true
```

Projektstruktur

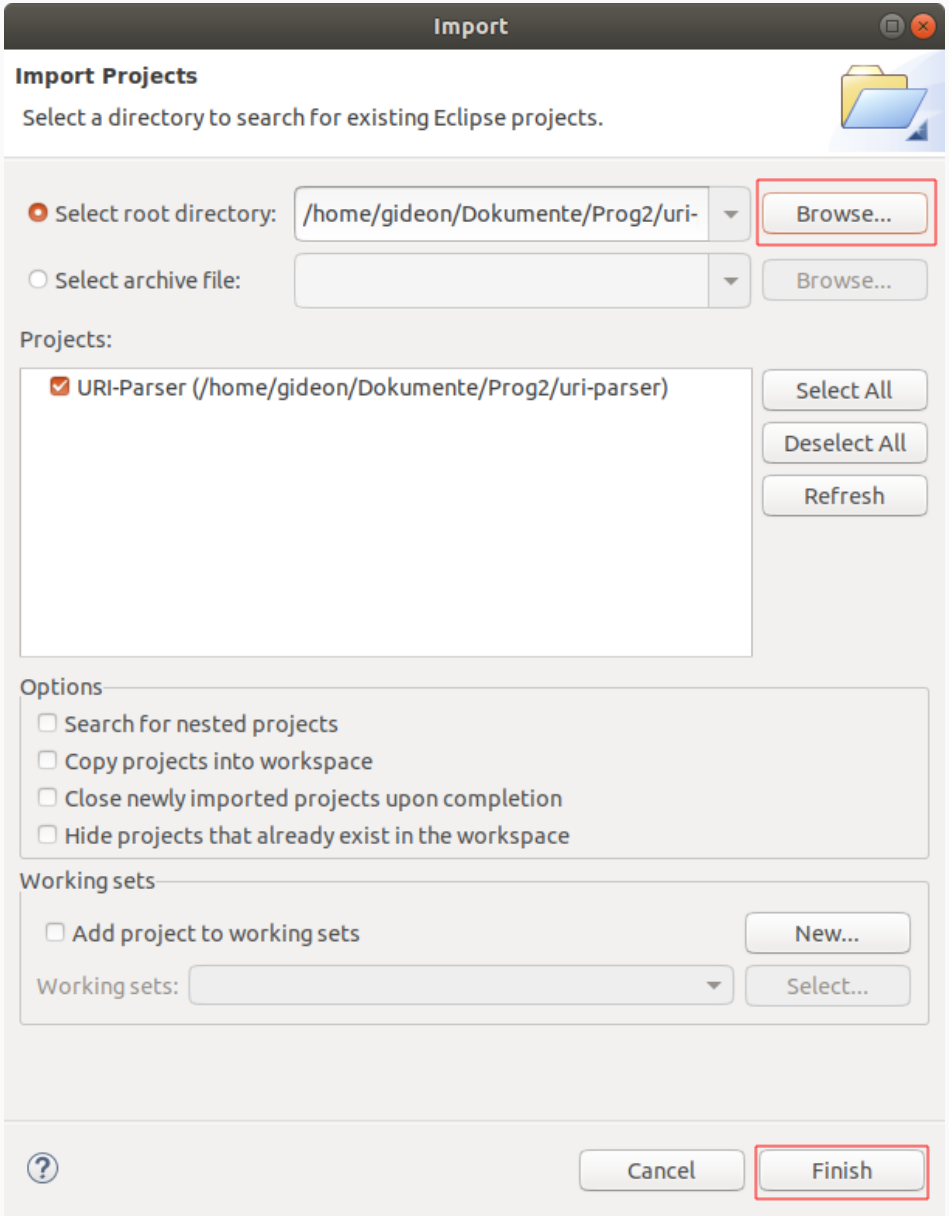


Projekt - Erste Schritte

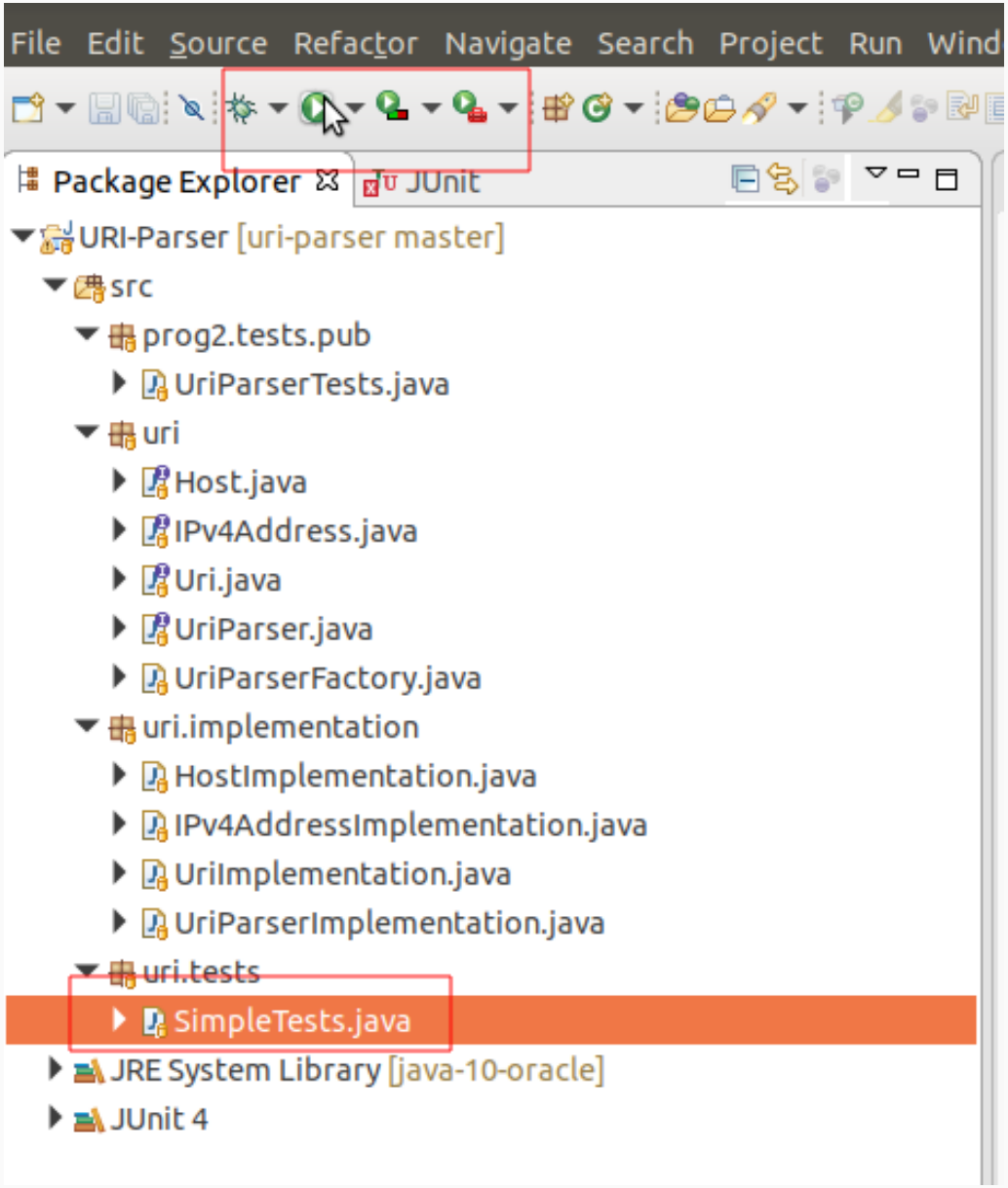
Projekt importieren...



Projekt importieren...

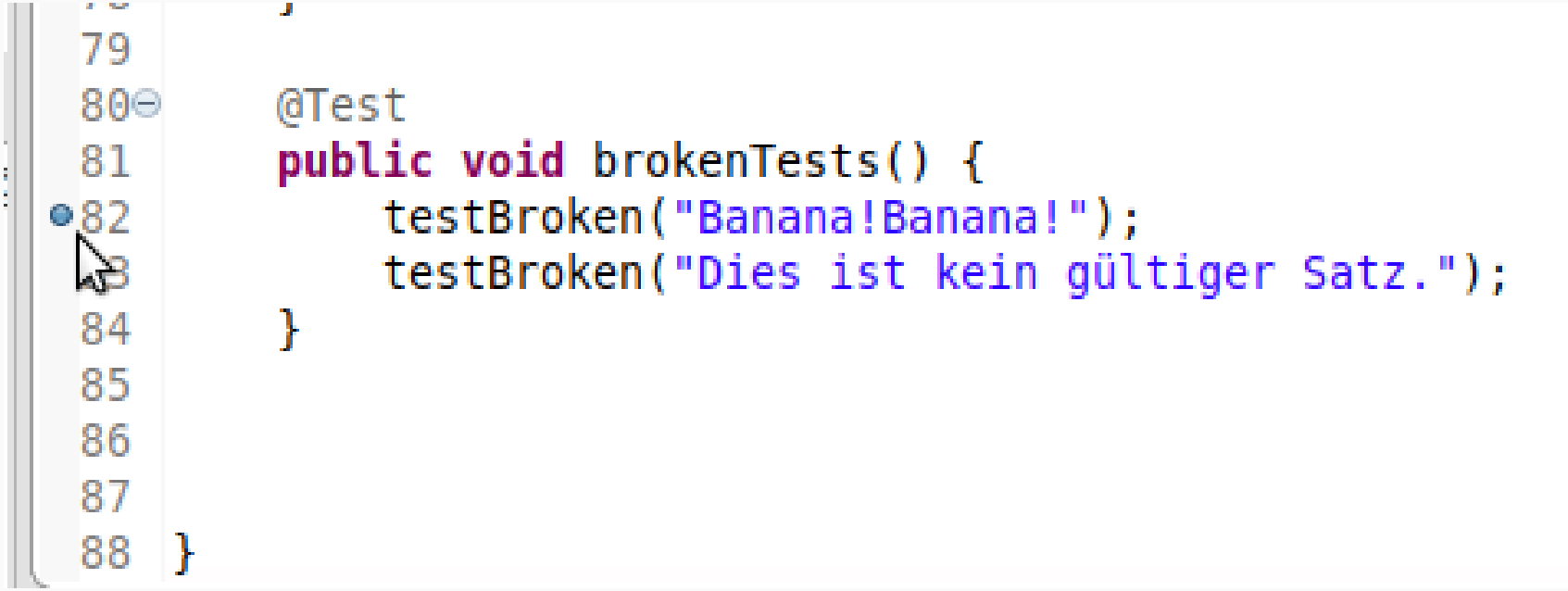


Tests ausführen

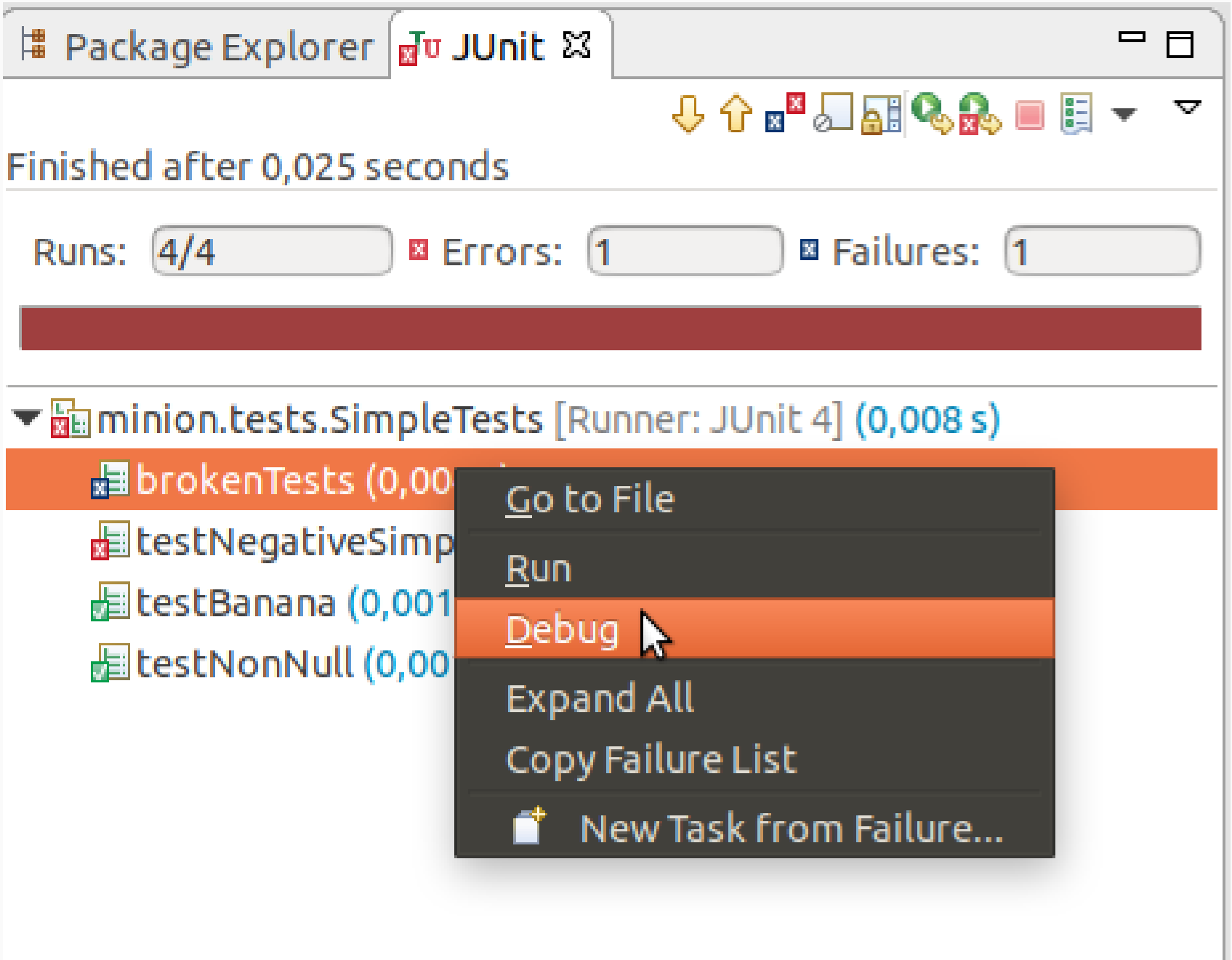


Breakpoint setzen

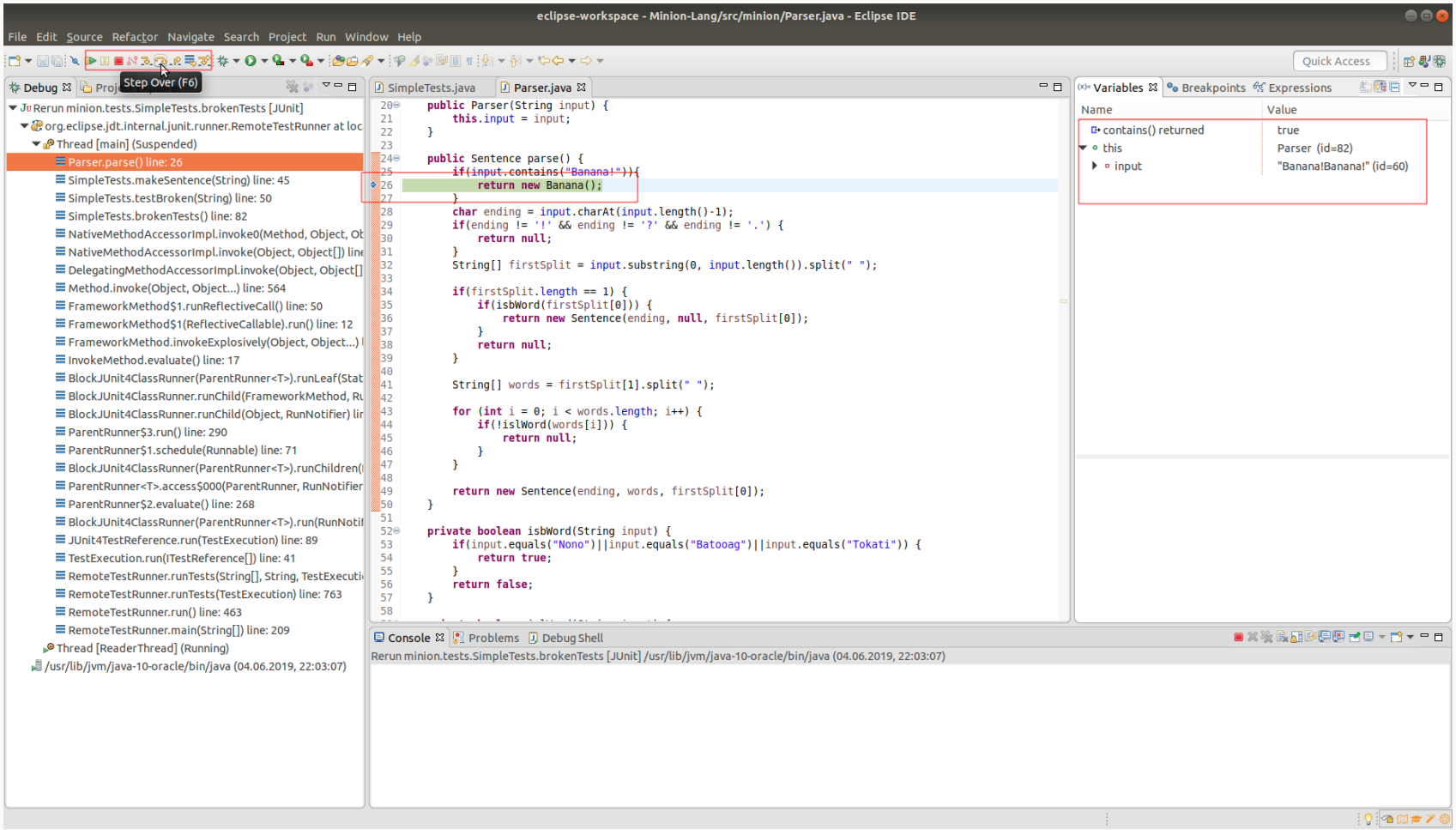
per Mausklick neben der Zeilennummer



Debugging starten



Programm analysieren




```
20    public Parser(String input) {
21        this.input = input;
22    }
23
24    public Sentence parse() {
25        if(input.contains("Banana!")){
26            return new Banana();
27        }
28        char ending = input.charAt(input.length()-1);
29        if(ending != '!' && ending != '?' && ending != '.') {
30            return null;
31        }
32        String[] firstSplit = input.substring(0, input.length()).split(" ");
33
34        if(firstSplit.length == 1) {
35            if(isbWord(firstSplit[0])) {
36                return new Sentence(ending, null, firstSplit[0]);
37            }
38            return null;
39        }
40
41        String[] words = firstSplit[1].split(" ");
42
43        for (int i = 0; i < words.length; i++) {
44            if(!islWord(words[i])) {
45                return null;
46            }
47        }
48
49        return new Sentence(ending, words, firstSplit[0]);
50    }
51
52    private boolean isbWord(String input) {
53        if(input.equals("Nono")||input.equals("Batooag")||input.equals("Tokati")) {
54            return true;
55        }
56        return false;
57    }
58
```

Das Mini(on)-Projekt

Minion-Projekt klonen

Um das Projekt zu clonen:

```
git clone https://prog2scm.cdl.uni-saarland.de/git/projektvorstellung m-lang
```

Um die Version mit Bugs anzusehen:

```
git checkout buggy
```

Um die Version ohne Bugs anzusehen:

```
git checkout fixed
```

Die Minion Grammatik

```
language = "Banana!" / sentence

sentence = bWord *(" " lWord) finish

finish   = "." / "?" / "!"

bWord  = "Nono" / "Batooag" / "Tokati"

lWord = "depdep" / "verlo" / "aka" / "ta" / "pak"
```