HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 2

Automaten und formale Sprachen Praktikum

Teammitglieder: Luis Staudt
Dominik Meurer

1 Aufgabe 1

1. $\{\epsilon, r, rr, rrr, ...\}$
2. $\{\epsilon, rt, rtrt, rtrtrt, ...\}$
3. $\{\epsilon, r, t, rt, rrtt, rrrttt, ...\}$
4. $\{\epsilon, tr, trtr, trtrtr, ...\}$
5. $\{\epsilon, r, t, rt, tr, rtr, trt, trr, ...\}$
6. $\{\epsilon, r, t, rr, tt, rrr, ttt, ...\}$
7. $\{\epsilon, rt, rtrt, rtrtrt, ...\}$
8. $\{\epsilon, 0, 1, 00, 01, 10, 11, 001, 010, 011, ...\}$
9. $\{\epsilon, 0, 1, 00, 01, 10, 11, 001, 010, 011, ...\}$

```java
package task2;

import org.junit.jupiter.api.Test;

import java.util.regex.Pattern;

import static org.junit.jupiter.api.Assertions.*;

public class RegexTest {

    @Test
    public void testGleitkommazahlen() {
        String gleitkomma = "[+-]?(([123456789]\\d*)(\\.\\d+)?|0(\\.\\d+)?)([eE]\\d+)?";

        assertTrue(Pattern.matches(gleitkomma, "22"));
        assertTrue(Pattern.matches(gleitkomma, "+2"));
        assertTrue(Pattern.matches(gleitkomma, "-10000000"));
        assertTrue(Pattern.matches(gleitkomma, "2244444444"));
        assertTrue(Pattern.matches(gleitkomma, "+2.2"));
        assertTrue(Pattern.matches(gleitkomma, "-23.211111"));
        assertTrue(Pattern.matches(gleitkomma, "-23e1"));
        assertTrue(Pattern.matches(gleitkomma, "23E1"));
        assertTrue(Pattern.matches(gleitkomma, "-23E123456634"));
        assertTrue(Pattern.matches(gleitkomma, "-211113.124566E123456634"));
        assertTrue(Pattern.matches(gleitkomma, "0"));
        assertTrue(Pattern.matches(gleitkomma, "0.0"));

        assertFalse(Pattern.matches(gleitkomma, "00"));
```

```
29          assertFalse(Pattern.matches(gleitkomma, "1."));
30          assertFalse(Pattern.matches(gleitkomma, "|1"));
31          assertFalse(Pattern.matches(gleitkomma, "|245"));
32          assertFalse(Pattern.matches(gleitkomma, "23e"));
33          assertFalse(Pattern.matches(gleitkomma, "23f1"));
34          assertFalse(Pattern.matches(gleitkomma, ""));
35          assertFalse(Pattern.matches(gleitkomma, ".0"));
36          assertFalse(Pattern.matches(gleitkomma, "7..0"));
37          assertFalse(Pattern.matches(gleitkomma, "e4"));
38          assertFalse(Pattern.matches(gleitkomma, "++e4"));
39          assertFalse(Pattern.matches(gleitkomma, "+-12"));
40          assertFalse(Pattern.matches(gleitkomma, "+ 12"));
41          assertFalse(Pattern.matches(gleitkomma, "1:0"));
42          assertFalse(Pattern.matches(gleitkomma, "1.1.0"));
43
44      }
45
46      @Test
47      public void testZeitpunkt() {
48          String zeitpunkt = "\\d{4}-(0[123456789]|1[012])-([012]\\d|3[01])T(0?\\d|1\\d
        |2[01234]):\\d{2}:\\d{2}:\\d{3}";
49
50          assertTrue(Pattern.matches(zeitpunkt, "2012-09-30T23:28:51:544"));
51          assertTrue(Pattern.matches(zeitpunkt, "2012-12-30T12:28:51:544"));
52          assertTrue(Pattern.matches(zeitpunkt, "0000-01-01T00:00:00:000"));
53          assertTrue(Pattern.matches(zeitpunkt, "2020-12-31T23:59:59:999"));
54
55          assertFalse(Pattern.matches(zeitpunkt, "    -  -  T  :  :  :   "));
56          assertFalse(Pattern.matches(zeitpunkt, "2012-20-30T23:28:51:544"));
57          assertFalse(Pattern.matches(zeitpunkt, "2012-10-32T23:28:51:544"));
58          assertFalse(Pattern.matches(zeitpunkt, "012-12-30T22:28:51:544"));
59          assertFalse(Pattern.matches(zeitpunkt, " 012-12-30T22:28:51:544"));
60          assertFalse(Pattern.matches(zeitpunkt, "2012-00-30T22:28:51:544"));
61          assertFalse(Pattern.matches(zeitpunkt, "2012-12-30t5:28:51:544"));
62          assertFalse(Pattern.matches(zeitpunkt, "2012-12-30T30:28:51:544"));
63          assertFalse(Pattern.matches(zeitpunkt, "2012-12-30T00:8:51:544"));
64          assertFalse(Pattern.matches(zeitpunkt, "2012-12-30T21:28:1:544"));
65          assertFalse(Pattern.matches(zeitpunkt, "2012-12-30T21:28:511:544"));
66          assertFalse(Pattern.matches(zeitpunkt, "2012-12-30T21:28:51:5440"));
67      }
68
69      @Test
70      public void testKFZKennzeichen() {
71          String kennzeichen = "([a-zA-Z]){1,3} [A-Za-z]{1,2} ([1-9]\\d{0,2} [HE]|[1-9]\\d{0,3})
        $";
72
73          assertTrue(Pattern.matches(kennzeichen, "VS HH 1000"));
74          assertTrue(Pattern.matches(kennzeichen, "HH HH 1"));
75          assertTrue(Pattern.matches(kennzeichen, "H HH 1"));
76          assertTrue(Pattern.matches(kennzeichen, "HH HH 1 H"));
77          assertTrue(Pattern.matches(kennzeichen, "T# HH 1 H")); // LaTeX findet das Zeichen
        nicht. Ich bin zu faul fuer den Fix, also gibts das.
78
79          assertFalse(Pattern.matches(kennzeichen, "HH HH 1 "));
80          assertFalse(Pattern.matches(kennzeichen, "VS VS 0"));
81          assertFalse(Pattern.matches(kennzeichen, "VS VS1 1"));
82          assertFalse(Pattern.matches(kennzeichen, "VS 1"));
83          assertFalse(Pattern.matches(kennzeichen, "VS ABC 1"));
84          assertFalse(Pattern.matches(kennzeichen, "VS HH 0"));
```

```
85          assertFalse(Pattern.matches(kennzeichen, "VS HH 10000"));
86          assertFalse(Pattern.matches(kennzeichen, "VS HH 99 EH"));
87          assertFalse(Pattern.matches(kennzeichen, "HVSA HH 1 "));
88          assertFalse(Pattern.matches(kennzeichen, "HH HH 01"));
89      }
90  }
```

Code Listing 1: Implementierung