一、 工具说明

本工具是用来计算正则表达式的五个属性：是否可空（Nullable）、First 集、Last 集、FollowLast 集合和 Flexible 属性。

计算方法在./src/main/java/cn/ac/ios/TreeNode/TreeNode.java 中的 calculateFiveAttributesNullableAndFirstAndLastAndFlexibleAndFollowLast 函数。主函数也在 TreeNode.java 中。

二、 使用方法为：

在 TreeNode.java 中修改 String regex = “...”, 然后执行即可

例如 String regex = “(\d+[abc]+)+”;

```
public static void main(String[] args) throws InterruptedException {
    String regex = "(\\d+[abc]+)+";
    TreeNode treeNode = createReDoSTree(regex);
    printTree(treeNode);
    treeNode.calculateFiveAttributesNullableAndFirstAndLastAndFlexibleAndFollowLast();
    System.out.println(treeNode.isNullable());
    System.out.println(treeNode.getFirst());
    System.out.println(treeNode.getLast());
    System.out.println(treeNode.getFollowLast());
    System.out.println(treeNode.isbFlexible() + " " + treeNode.getdFlexible());
}
```

输出结果为：

```
'- (\d+[abc]+)+
   |- (\d+[abc]+)
   |  |- (
   |  |- \d+[abc]+
   |  |  |- \d+
   |  |  |  |- \d
   |  |  |  '- +
   |  |  '- [abc]+
   |  |     |- [abc]
   |  |     |  |- [
   |  |     |  |- a
   |  |     |  |- b
   |  |     |  |- c
   |  |     |  '- ]
   |  |     '- +
   |  '- )
   '- +

false
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[a, b, c]
[a, b, c, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
true -1.0
```

其中多叉树为对正则表达式 r = (\d+[abc]+)+经过 ANTLR 工具解析得到的语法树。

而后五行分别为：

(1) r 是否为可空的(Nullable)

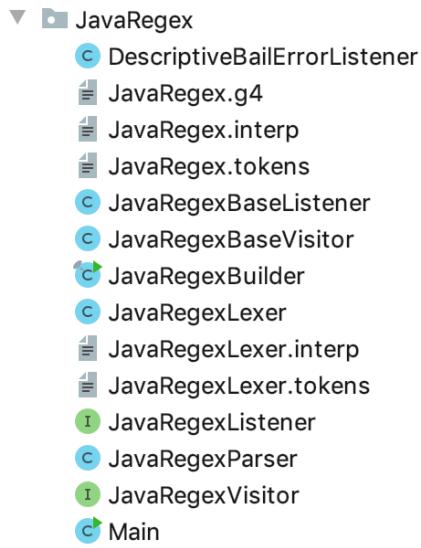
(2) r 的 first 集

(3) r 的 last 集

(4) r 的 followLast 集

(5) r 的 bFlexible 属性、dFlexible 属性

其中 ANLTR 支持多种语言的解析，包括 PCRE、Python、Java、JavaScript，对应语法文件在相应的文件夹中，例如 Java 的语法树文件在 JavaRegex 中



三、 已支持的正则表达式形式：

Definition 2. Real-world Regular Expression (regex). A regex over Σ is a well-formed parenthesized formula, consisting of operands in $\Sigma^* \cup \{\backslash i \mid i \geq 1\}$[3]. Besides the common rules governing standard regular expressions (e.g. $r_1|r_2, r_1r_2, r_1\{m,n\}$ defined in Definition 1), a regex also has the following constructs: (i) capturing group (r); (ii) non-capturing group $(?:r)$; (iii) lookarounds: positive lookahead $r_1(?=r_2)$, negative lookahead $r_1(?!r_2)$, positive lookbehind $(?<=r_2)r_1$, and negative lookbehind $(?<!r_2)r_1$; (iv) anchors: Start-of-line anchor $\hat{}$, End-of-line anchor $, word boundary \b, and non-word boundary \B; (v) lazy quantifiers: $r??, r*?, r+?$, and $r\{m,n\}?$; and (vi) backreference \i.

本工具支持上述定义 2 中除了反向引用的所有形式，例如[^abc], [\d\w], \S, ^abc$, (?=a)\w+, a++, \w{2,4}，(?:xyz)等等

四、 特别说明

1. 本工具默认使用 java 语法树进行解析，如需更换，可通过调用 createReDoSTree 函数来传入语言。
2. 对于特殊字符“.”，在正则表达式中代表除\r 和\n 外的任意字符，工具为了对此符号进行了优化。具体为：若表达式中含有补集类符号，如[^abc], \S, \W, 则全集定义为{正则表达式中出现的字符}∪{字母数字下划线}∪{感叹号}。例如 r = “[^a-z]”的 first 集、last 集、followLast 集为：

```
[!, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, _]
[!, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, _]
[]
```

此优化体现在函数 calculateFiveAttributesNullableAndFirstAndLastAndFlexibleAndFollowLast 的开头部分，如下图。

```
2251        if (rootAlphabet.isEmpty()) {
2252            // 为处理补结点的问题，首先获取树的字母表Σ, getLetterSet(true)
2253            TreeNode treeNodeCopy = SerializationUtils.clone( object: this);    // 深拷贝
2254            // 去补
2255            removeNegateSymbol(treeNodeCopy, Constant.SimplyLevel.HIGH);
2256            // 删除零宽断言
2257            deleteZeroWidthAssertion(treeNodeCopy);
2258            // 使用重写后的去首尾^$
2259            treeNodeCopy.deleteCaretAndDollarSymbols();
2260            // 新版重写空串
2261    //          treeNodeCopy = removeBlankStr(treeNodeCopy);
2262            treeNodeCopy.removeBlankStr();
2263            // 重写反向引用后 删除NonCapturingGroupFlag ?:
2264            treeNodeCopy.deleteNonCapturingGroupFlag();
2265
2266
2267            rootAlphabet = treeNodeCopy.getLetterSet( addDefaultw: true, repairBackSlashElements: true);
2268        }
```

若注释掉这一部分，则全集默认为 unicode 字符集（65536 个字符）。

3. 对于 lookaround 的处理，目前为直接定义为无效值（空集），这是一个需要优化的点。