Static Code Analysis

Lab. 2 – Parsing with JavaCC
Bartosz Bogacki '2007.

Helpful links:
https://javacc.dev.java.net/doc/docindex.html
http://www.cobase.cs.ucla.edu/pub/javacc/
http://www.engr.mun.ca/~theo/JavaCC-FAQ/javacc-faq-ie.htm
http://www.cs.put.poznan.pl/wcomplak/

Exercise 1:

Write syntax analyzer in JavaCC for language $a^n b^n$, where n>0. White spaces and new line characters should be ignored. If input stream is correct according to specified rules, the program should print "OK" message. Otherwise "Error" message should appear preceded with the line number where error occurred. For example, for the following input stream:

```
aaa
b
bb
```

the response of the program should be:

```
OK
```

for the input stream like:

```
aaa
ba
bb
```

the output should be:

```
2: Syntax error
```

Prepare two versions of the parser. One that uses incremented LOOKAHEAD value and the other based on left-factored grammar.

Exercise 2:

Input stream consists of a number of rows. Each row is built of a number of characters ‘x’ followed by zero or more white spaces. Characters form triangle shape, as a number of ‘x’ characters in row is equal to the row number. This means, that in first row is only one character ‘x’, in second row there are two characters ‘x’ and so on. Write syntax analyzer in JavaCC, that will verify that the input stream is valid according to specified rules. If so, then “OK” message should appear. Otherwise the parser should print “Error!!!” message.

For the following input stream:

```
x
xx
xxx
xxxx
xxxxx
```

The output should be:

```
OK
```

For the input stream like:

```
x
xx

xxxx
xxxxx
```

The output is:

```
Error !!!
```

Exercise 3:

Using prepared Java grammar file (for example the one at http://www.cs.put.poznan.pl/bbogacki/JavaParser.jj) write program that will extract all strings from Java program to XML file.

For example, for Java file:

```
/** "This shouldn't appear!" */
public class Test {
  public int a;
  private String text = "This should appear [0]";
}
```

The output should be:

```
<?xml version="1.0" encoding="UTF-8"?>
<class name="Test">
<string line="4" column="23" value="This should appear [0]"/>
</class>
```

Exercise 4:

Using prepared Java grammar file (for example the one at http://www.cs.put.poznan.pl/bbogacki/JavaParser.jj) write program that will verify if class extends `TestCase` class and if name of each public method, except `main()`, `setUp()` and `tearDown()` starts with `"test"`. If not, then the program should print appropriate error message.

Example:

For the following input stream:

```
 1 public class Test {
 2  public int a = 5;
 3
 4  public static void main(String[] args) {
 5    for (int i = 0; i < a; i++) {
 6      a += i;
 7    }
 8  }
 9
10  void testA() {
11  }
12
13  void foo() {
14  }
15
16  int bar() {}
17
18  void tearDown () {}
19 }
```

The following output should appear:

```
Class "Test" is not subclass of "TestCase".
Method: "foo" on line: 13 doesn't start with word "test".
Method: "bar" on line: 16 doesn't start with word "test".
```

Exercise 5:

Using prepared Java grammar file (for example the one at http://www.cs.put.poznan.pl/bbogacki/JavaParser.jj) write program that will verify if all referenced local variables and fields were declared. If the referenced variable wasn't declared, the program should print error message including variable name and line number where undeclared variable occurred.

Example:

For the following input stream:

```
1 public class Test {
2  public int a = 5;
3
4  public static void main(String[] args) {
5    for (int i = 0; i < a; i++) {
6      a += i * j;
7    }
8  }
9 }
```

The output should be:

```
Variable: "j" on line: 6 used, but not initialized.
```