apl. Prof. Dr. Thomas Noll
Sebastian Junges, Christoph Matheja, Matthias Volk

Compiler Construction 2017
— Programming Exercise 5 —

Upload in L2P until July 18th before the exercise class.

General Remarks

- This programming exercise is optional and provides bonus points for the exercise class.

Programming Exercise 1 (* 3 Bonus Points)

In this exercise we implement a semantic check. In our *WHILE* language we require that every variable identifier is *declared* before the variable is used (read or set). Additionally, a variable defined inside a scope like an `if` statement or a `while` loop is not visible outside this scope. We do not care whether a variable has been *initialised* before it is read. Examples:

This is valid:

```
int x; int y;
if (x <= y) {
    write("Hello world.");
}
write(x);
```

This is not valid (`y` is undefined and `z` is undefined outside the `if`-statement):

```
int x;
if (x <= y) {
    int z;
}
write(z);
```

Implement `checker.DeclarationChecker.checkDeclaredBeforeUsed()`.

Hint: for this you do not need to implement any attributed grammars and their evaluation. Instead simply walking the abstract syntax tree once and checking the required property suffices.

To check the type of a given non-terminal or token, you can use the methods in `util.WhileAlphabet`.

You can also generate a visualization of the abstract syntax tree obtained by the parser. With the optional commandline argument `-dot output.dot` the abstract syntax tree is written as dot output to the given file. Then the following command generates a PDF depicting the tree:

```
$dot -Tpdf output.dot -o output.pdf
```