

## Exercise Sheet 9

This exercise sheet is about implementing the Visitor Pattern and building a symbol table for MiniJava programs. In order to make the exercise interesting, MiniJava is enhanced with local declarations at the beginning of compound statements. Thus, the following program is considered to be a valid MiniJava program:

```
int i , j , k , n;
i = read();
k = -1;
j = read();
n = 0;
while(j >= -k) {
    int k, j;
    if(i > j)
        i = i - k;
    else
        j = j % i;
    n = (n - k) * 2 - 1;
}
write(i);
```

The attached project provides a Lexer and a Parser for MiniJava. The project contains classes to represent the nodes of the AST of a MiniJava program. The example main program reads a file given on the command line and provides you with the corresponding AST. It is recommended that you start by running the program using a graphical debugger and the example from above and click through the AST to get familiar with the different classes.

### Assignment 9.1. Visitor

Your first task is to implement the Visitor Pattern in order to allow client code to traverse a MiniJava AST. As a reminder, implementing the Visitor Pattern consists of the following steps:

1. implementing an `accept()` method for every class to be visited,
2. implementing a `Visitor` class that contains a `visit()` method for every class to be visited,
3. and implementing the `visit()` methods (only a visit instead of a pre and post method is needed) in such a way that the AST is traversed from left to right.

Test your visitor by counting the number of integer constants in a MiniJava program.

### Assignment 9.2. Symbol Table

We are going to use the visitor for the construction of the symbol table. In the course of this, we represent the symbol table data using pointers: Each identifier obtains a reference to its declaration. Hence, your first task is to add a reference to `Decl` objects in the `Expr.Identifier` class.

Next, implement a visitor for calculating the symbol table data. Your visitor class needs to be derived from the visitor class implemented for the first exercise. As you know from the lecture, the state of the visitor contains a map that maps each identifier to a stack of its currently active scopes. Make sure that your visitor reports a meaningful error message through an exception in case the MiniJava program refers to an undeclared variable.

Finally, test your implementation using a graphical debugger.