SAARLAND UNIVERSITY
COMPUTER SCIENCE

Prof. Dr. Reinhard Wilhelm
Prof. Dr. Sebastian Hack
Dipl.-Inform. Daniel Grund
Michael Jacobs, M.Sc.

Compiler Construction Project WS11/12

Project task C. Parser and AST Construction

Implement a recursive descent parser for MiniJava:

- The parser must reject all words that are not derivable with the grammar, G, given in the language specification. The parser must accept all correct MiniJava programs. Remember that $L(G) \supseteq$ MiniJava.
- Also, construct a syntax tree for syntactically correct inputs.
- Check your implementation against the provided test cases and write additional test cases on your own.
- The next project task will be to pretty-print source code from the AST in two different flavors.

Before you start hacking the parser, plan ahead:

- Find the ambiguities in G and its left-recursive productions and resolve them as you deem it fit. For your revised grammar, G', determine the FiFo-sets and a k such that G' is $SLL(k)$.
- How to implement the k-lookahead capability in your lexer/parser?
- How to represent the AST? What classes and class hierarchy for AST nodes do you need, e.g. `Expression`, `BinaryExpression`?

Additional technical requirements and restrictions:

- `mjavac --parse [file]` must perform the syntactical analysis and accept (reject) the syntactically correct (incorrect) programs and terminate with return code 0 (1).

Please check in your solution into your repository until 2011-11-17, 12:00, noon.