Prof. Dr. Peter Thiemann
Matthias Keil

Winter Semester 2016/2017

Compiler Construction

http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2016ws/

Exercise Sheet 2

1 Lexing and Parsing of MiniJava (6 + 2 + 2 Points)

MiniJava is a subset of Java. The semantics of a MiniJava program is given by its semantics as a Java program. Overloading is not allowed in MiniJava. The MiniJava statement `System.out.println( ... );` can only print integers. The MiniJava expression `e.length` only applies to expressions of type `int[]`.

Grammar

Program	→	*MainClass ClassDecl**
MainClass	→	**class** *id* **{ public static void main (** **String** [ ] *id* **) {** *VarDecl* Statement** **} }**
ClassDecl	→	**class** *id* **{** *VarDecl* MethodDecl** **}**
	→	**class** *id* **extends** *id* **{** *VarDecl* MethodDecl** **}**
VarDecl	→	*Type id* **;**
MethodDecl	→	**public** *Type id* **(** *ParamList?* **) {** *VarDecl* Statement** **return** *Exp*; **}**
ParamList	→	*Type id ParamRest**
ParamRest	→	**,** *Type id*
Type	→	**int** [ ]
	→	**boolean**
	→	**int**
	→	*id*
Statement	→	**{** *Statement** **}**
	→	**if (** *Exp* **)** *Statement* **else** *Statement*
	→	**while (** *Exp* **)** *Statement*
	→	**System.out.println (** *Exp* **) ;**
	→	*id* **=** *Exp*;
	→	*id* **[** *Exp* **]** **=** *Exp*;
Exp	→	*Exp op Exp*
	→	*Exp*[*Exp*]
	→	*Exp* **.length**
	→	*Exp* **.** *id* **(** *ExpList?* **)**
	→	**true**
	→	**false**
	→	*id*
	→	⟨integer literal⟩

$\rightarrow$ **this**
$\rightarrow$ **new int** [*Exp*]
$\rightarrow$ **new** *id* ( )
$\rightarrow$ **!** *Exp*
$\rightarrow$ (*Exp*)

ExpList $\rightarrow$ *Exp ExpRest**

ExpRest $\rightarrow$ **,** *Exp*

id $\rightarrow$ ⟨identifier⟩

op $\rightarrow$ &&
$\rightarrow$ +
$\rightarrow$ -
$\rightarrow$ *
$\rightarrow$ <

Project - Part 1

- Implement a lexer and parser for MiniJava in SableCC. Insert for the package declaration `Package minijava;` . You may assume that an identifier is a sequence of letters, digits, and underscores, starting with a letter. Further, integer literals will be only given in decimal notation and without suffix.

  Remember, there are two kinds of comments in Java: block comments (`/* text */`) where all the text from the ASCII characters `/*` to the ASCII characters `*/` is ignored, and end-of-line comments (`// text`) where all the text from the ASCII characters `//` to the end of the line is ignored.

- As the concrete syntax often is rather complex and not-suitable for tree traversals, SableCC 3.6 offers the possibility to specify also a simpler abstract syntax within the grammar specification. Define such an abstract grammar for MiniJava and annotate the concrete syntax to define how it is translated to the abstract syntax.

- Describe the overall structure of your specification shortly. In particular, explain how you implemented operator precedence.

Submission

- Deadline: **17.11.2016, 12:00 (noon)**. Late submissions will not be accepted.

- Submit your solution to the subversion repository. Your submission will consist of one folder (exercise2) which includes your solution.

- Your solution will consist of a file `minijava.sable` with the grammar specification and a pdf `minijava-<your name>.pdf` with a description.

- If invoking SableCC on the grammar specification leads to errors, you will receive no points.

- If it can be compiled, but crashes or loops on all test cases, it will receive no points.
- You are strongly encouraged to test your solution with the provided test data. Add test cases as you might think necessary. You need not submit your own test cases.
- The description must be limited to one page. Submitting more than one page will lead to reduction in points.
- The description may be either German or English. Clear and understandable style is required.