Complier Lab (CSE4112)

Assignment #3 Syntax Analysis of Javali

Assignment Details: We have been working on a subset of the Javali language syntax. Now we will work on the complete grammar of Javali language. Please refer to the complete syntax of the language and the semantics of the language.

Your task in this assignment is to parse a complete program written in this language. However, for this assignment we will assume couple of restrictions:

1. In addition to the main class there will be classes but no class inheritances. These classes may have a number of variables and method declarations. We will generate 3AC corresponding to the program, i.e., code corresponding to each method (put a label to indicate the start of 3AC for a method in the sequence of 3AC for the whole program) and method calls. See the 3AC format given in previous assignment for method call. You may assume, for this assignment, all parameters are passed using call-by-value.

2. We will not implement any of the different semantic checking in this assignment. For example, the syntax allows nested method definitions, but our code generator is not expected to deal with them. Further, the Javali syntax accepts certain incorrect constructs, such as this[0], that will be rejected by the semantic analyzer later on. You are not required to parse constructs that you will not support (e.g., nested methods) or constructs that are not correct (e.g., this[0]). If you would prefer, however, to parse such constructs, you are permitted to do so, so long as your semantic analyzer (the next Assignment) rejects them. But you will implement different syntax error messages (but not semantic ones). These will be mostly proper error messages for different tokens being missing. Even if you cannot find all possible syntax errors your parser needs to recover from the error and parse the whole program. See the 'error' terminal for YACC/Bison and how to writing production rules with error terminals.

3. You do not need to generate short circuit code for relational operators but instead of generating code like t1 = a > b generate the code fragment:
100: if a > b goto 103
101: t1 = 0
102: goto 104
103: t1 = 1

4. In Javali arrays are one dimensional and array of objects can be declared. For this assignment evaluate (generate 3AC) for the array index expression (simpleExpr) then simply use 3AC corresponding for one dimensional arrays (e.g., a[t4]=5 or a[t4].ele = 5). We will examine storage allocation for local variables of methods and arrays in an assignment on code generation.

Assignment Due:12/04. You may work on this and subsequent assignments in teams of two. But form teams with your respective group members.