Since most of the java code produced by the team is trivial (e.g. PrettyPrinter to print the abstract syntax tree, main class to read input file and hand to the lexer), effort shall be spent in reviewing the quality of the grammar files (used by LPG to generate the parser).

Because of the limitation in features of LPG as a full-blown programming language (like most other domain specific languages), not much can be reviewed and said about many aspects of software construction (e.g. class/method organization; implementation of patterns, dependency between classes/methods, exception handling, performance, robustness etc…).

In my opinion, the most important aspects of code in a DSL are correctness, readability, and maintainability. For a DSL, readability is much closed to maintainability, thus they shall be grouped together.

Correctness

In this context, the program is considered to be correct if it implements the grammar spec. For a parser, it can be verified by:

- LPG compiler: a lot of aspects like LALR (k) comformance, conflicts, redundancy are checked by LPG compiler. Both the parser and lexer are LALR(k) comformed, and contain no conflicts.
- Review: grammar is reviewed by the team to guarantee it contains correct and complete grammar specification, with the exception of path element (needs to be prefixed with ./ or /).
- Testing: provided test cases are run sucessfully with LPG parser (with the exception of the path element, which needs to be adapted in the test case).

Readability/ Maintainability

The following review comments (mostly over readability) have been made and processed in the grammar files:

- Get rid of autogenarated Attribute0, Attribute1, Attribute2...; Predicate0, predicate1, Type0, Type1, Type2…. by giving name to individual items in grammar files (e.g. Type0 becomes list, Type1 becomes string etc…)
- Remove some warnings & informatives (unused tokens SLChar, OctalDigit3 …)
- Renamed the grammar files (test.g, javalexer.g) to more meaningful name (WaebricParser.g, WaebricParser.g)
- Reorder grammar definition to more logical regions (e.g. function, module move to almost top – top down approach).

Further:

- Naming of grammar elements: Grammar elements, when possible, are kept consistent with the naming of waebric spec. This makes maintainability easier with evolution of the waebric

language. In some cases, like for statement, to prevent ambiguity which leads to shift-reduce or reduce conflict, definition of statement has to be splitted up and new statement elements have to be created. The naming in that case still made self-explainatory.

- The parser and lexer files are organized into logical region (top-down), with related elements kept close together. This improves readability and maintainability.

Robustness

This is more an aspect of LPG instead of grammar files, but it is worthy to note that LPG is very robust against “faulty” grammar files (in some cases it could be the implementation is not complete according to grammar spec). LPG parser when encounters an error can go on parsing the program, and reports all the errors it encounters.