The Language Javalette

BNF-converter

April 19, 2011

This document was automatically generated by the *BNF-Converter*. It was generated together with the lexer, the parser, and the abstract syntax module, which guarantees that the document matches with the implementation of the language (provided no hand-hacking has taken place).

The lexical structure of Javalette

Identifiers

Identifiers $\langle Ident \rangle$ are unquoted strings beginning with a letter, followed by any combination of letters, digits, and the characters `_ '`, reserved words excluded.

Literals

Integer literals $\langle Int \rangle$ are nonempty sequences of digits.

Double-precision float literals $\langle Double \rangle$ have the structure indicated by the regular expression $\langle digit \rangle + \text{`.'} \langle digit \rangle + (\text{`e'`-'}? \langle digit \rangle +)?$ i.e. two sequences of digits separated by a decimal point, optionally followed by an unsigned or negative exponent.

String literals $\langle String \rangle$ have the form `"`x`"`, where x is any sequence of any characters except `"` unless preceded by `\`.

Reserved words and symbols

The set of reserved words is the set of terminals appearing in the grammar. Those reserved words that consist of non-letter characters are called symbols, and they are treated in a different way from those that are similar to identifiers. The lexer follows rules familiar from languages like Haskell, C, and Java, including longest match and spacing conventions.

The reserved words used in Javalette are the following:

```
boolean  double  else
false    if      int
return   true    void
while
```

The symbols used in Javalette are the following:

```
(    )    ,
{    }    ;
=    ++   --
-    !    &&
||   <    >
+    *    /
%    <=   >=
==   !=
```

Comments

Single-line comments begin with `#`, `//`.
Multiple-line comments are enclosed with `/*` and `*/`.

The syntactic structure of Javalette

Non-terminals are enclosed between ⟨ and ⟩. The symbols ::= (production), | (union) and ϵ (empty rule) belong to the BNF notation. All other symbols are terminals.

⟨*Program*⟩ ::= ⟨*ListTopDef*⟩

⟨*TopDef*⟩ ::= ⟨*Type*⟩ ⟨*Ident*⟩ `(` ⟨*ListArg*⟩ `)` ⟨*Block*⟩

⟨*ListTopDef*⟩ ::= ⟨*TopDef*⟩
  | ⟨*TopDef*⟩ ⟨*ListTopDef*⟩

⟨*Arg*⟩ ::= ⟨*Type*⟩ ⟨*Ident*⟩

⟨*ListArg*⟩ ::= ϵ
  | ⟨*Arg*⟩
  | ⟨*Arg*⟩ `,` ⟨*ListArg*⟩

⟨*Block*⟩ ::= { ⟨*ListStmt*⟩ }

$$
\begin{array}{lcl}
\langle \mathit{ListStmt} \rangle & ::= & \epsilon \\
 & | & \langle \mathit{Stmt} \rangle \ \langle \mathit{ListStmt} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Stmt} \rangle & ::= & \texttt{;} \\
 & | & \langle \mathit{Block} \rangle \\
 & | & \langle \mathit{Type} \rangle \ \langle \mathit{ListItem} \rangle \ \texttt{;} \\
 & | & \langle \mathit{Ident} \rangle \ = \ \langle \mathit{Expr} \rangle \ \texttt{;} \\
 & | & \langle \mathit{Ident} \rangle \ ++ \ \texttt{;} \\
 & | & \langle \mathit{Ident} \rangle \ -- \ \texttt{;} \\
 & | & \texttt{return} \ \langle \mathit{Expr} \rangle \ \texttt{;} \\
 & | & \texttt{return ;} \\
 & | & \texttt{if (} \ \langle \mathit{Expr} \rangle \ \texttt{)} \ \langle \mathit{Stmt} \rangle \\
 & | & \texttt{if (} \ \langle \mathit{Expr} \rangle \ \texttt{)} \ \langle \mathit{Stmt} \rangle \ \texttt{else} \ \langle \mathit{Stmt} \rangle \\
 & | & \texttt{while (} \ \langle \mathit{Expr} \rangle \ \texttt{)} \ \langle \mathit{Stmt} \rangle \\
 & | & \langle \mathit{Expr} \rangle \ \texttt{;}
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Item} \rangle & ::= & \langle \mathit{Ident} \rangle \\
 & | & \langle \mathit{Ident} \rangle \ = \ \langle \mathit{Expr} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{ListItem} \rangle & ::= & \langle \mathit{Item} \rangle \\
 & | & \langle \mathit{Item} \rangle \ \texttt{,} \ \langle \mathit{ListItem} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Type} \rangle & ::= & \texttt{int} \\
 & | & \texttt{double} \\
 & | & \texttt{boolean} \\
 & | & \texttt{void}
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{ListType} \rangle & ::= & \epsilon \\
 & | & \langle \mathit{Type} \rangle \\
 & | & \langle \mathit{Type} \rangle \ \texttt{,} \ \langle \mathit{ListType} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr6} \rangle & ::= & \langle \mathit{Ident} \rangle \\
 & | & \langle \mathit{Integer} \rangle \\
 & | & \langle \mathit{Double} \rangle \\
 & | & \texttt{true} \\
 & | & \texttt{false} \\
 & | & \langle \mathit{Ident} \rangle \ \texttt{(} \ \langle \mathit{ListExpr} \rangle \ \texttt{)} \\
 & | & \langle \mathit{String} \rangle \\
 & | & \texttt{(} \ \langle \mathit{Expr} \rangle \ \texttt{)}
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr5} \rangle & ::= & - \ \langle \mathit{Expr6} \rangle \\
 & | & \texttt{!} \ \langle \mathit{Expr6} \rangle \\
 & | & \langle \mathit{Expr6} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr4} \rangle & ::= & \langle \mathit{Expr4} \rangle \ \langle \mathit{MulOp} \rangle \ \langle \mathit{Expr5} \rangle \\
 & | & \langle \mathit{Expr5} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr3} \rangle & ::= & \langle \mathit{Expr3} \rangle \ \langle \mathit{AddOp} \rangle \ \langle \mathit{Expr4} \rangle \\
& | & \langle \mathit{Expr4} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr2} \rangle & ::= & \langle \mathit{Expr2} \rangle \ \langle \mathit{RelOp} \rangle \ \langle \mathit{Expr3} \rangle \\
& | & \langle \mathit{Expr3} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr1} \rangle & ::= & \langle \mathit{Expr2} \rangle \ \texttt{\&\&} \ \langle \mathit{Expr1} \rangle \\
& | & \langle \mathit{Expr2} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{Expr} \rangle & ::= & \langle \mathit{Expr1} \rangle \ || \ \langle \mathit{Expr} \rangle \\
& | & \langle \mathit{Expr1} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{ListExpr} \rangle & ::= & \epsilon \\
& | & \langle \mathit{Expr} \rangle \\
& | & \langle \mathit{Expr} \rangle \ \texttt{,} \ \langle \mathit{ListExpr} \rangle
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{AddOp} \rangle & ::= & + \\
& | & -
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{MulOp} \rangle & ::= & \texttt{*} \\
& | & \texttt{/} \\
& | & \texttt{\%}
\end{array}
$$

$$
\begin{array}{lcl}
\langle \mathit{RelOp} \rangle & ::= & < \\
& | & <= \\
& | & > \\
& | & >= \\
& | & == \\
& | & \texttt{!}=
\end{array}
$$