Exercise 7: Scripting Language for Boolean Expressions

The following grammar describes a little language that can be used to calculate the values of Boolean expressions, store them in variables, and print them:

```
Program   = { Statement }.
Statement = ident "=" Expr ";"
          | "print" Expr ";".
Expr      = Term { "||" Term }.
Term      = Factor { "&&" Factor }.
Factor    = "true" | "false" | ident | "(" Expr ")" | "!" Factor.
```

Turn this grammar into an attributed grammar so that the value of Boolean expressions can be calculated and stored in variables or printed. Keep the variables and their values in a global hash table (HashMap). An example program might look like this:

```
big = true;
small = ! big;
ready = false;
print (big || small) && ready || big && ! ready;
```

Solution

We use a global hash table to manage the names and their values:

```
HashMap<String, Boolean> values = new HashMap<String, Boolean>();
```

The attributed grammar then looks as follows:

```
Program  = { Statement }.

Statement                (. boolean val; .)
= ident                  (. String name = t.val; .)
  "="
  Expr <↑val>            (. values.put(name, val); .)
  ";"
| "print"
  Expr <↑val>            (. System.out.println(val); .)
  ";".

Expr <↑val>              (. boolean val, val2; .)
= Term <↑val>
  { "||" Term <↑val2>    (. val = val || val2; .)
  }.

Term <↑val>              (. boolean val, val2; .)
= Factor <↑val>
  { "&&" Factor <↑val2>  (. val = val && val2; .)
  }.

Factor <↑val>            (.  boolean val; .)
=  "true"                (.  val = true; .)
|  "false"               (.  val = false; .)
|  ident                 (.  Boolean c = values.get(t.val);
                            if (c != null) val = c;
                            else { error("undeclared name"); val = false; } .)
|  "(" Expr <↑val> ")"
|  "!" Factor <↑val>     (.  val = ! val; .) .
```