

Compilers
Introductory Project
Spring 2010
Due Friday, Feb. 5th, 2010

The purpose of this assignment is to get you back into the Java programming groove, and to give you a practical introduction to some of the theoretical topics you will see over the next several weeks. This is *not* a trivial assignment! Start early, and come by my office to ask questions. For this project you will create a SimpleLogic parser. That is, you will write a program that takes as input a SimpleLogic program, and determines if that program contains any syntax errors.

1 SimpleLogic Programming Language

1.1 Tokens

The legal tokens in SimpleLogic are:

Token name	Description
Left Brace	{
Right Brace	}
Left Parenthesis	(
Right Parenthesis)
Equals	=
Semicolon	;
Keyword do	do
Keyword while	while
Keyword true	true
Keyword false	false
Identifier	Sequence of letters that is not a keyword Examples: x, foo, bigshot

Note that all letters are lower-case in a valid SimpleLogic program.

1.2 SimpleLogic Statements

A SimpleLogic statement is either an assignment statement, a do-while statement, a while statement, or a block statement.

1.2.1 Assignment statement

An assignment statement is of the form:

<Identifier> = <expression>;

where `<Identifier>` is just an identifier token as described above, and an `<expression>` is either an identifier token, the keyword `true`, the keyword `false`, or `(<expression>)`. Thus all of the following are valid assignment statements:

```
x = y;
foo = (true);
w = z;
z = ((z));
```

And *none* of the following are valid assignment statements:

```
x = while;
true = false;
true = true;
x = ;
y = ((false));
```

1.2.2 do-while statement

A do-while statement is of the form

```
do <statement> while (<expression>);
```

where `<expression>` is an expression as defined above (identifier, boolean literal, parentheses allowed), and a `<statement>` is any SimpleLogic statement. Thus the following are all legal SimpleLogic statements:

```
do x = false; while (x);
do x = true; while (true);
do x = y; while ((false));
```

Note that whitespace (tabs, spaces, end of line characters) are allowed between SimpleLogic tokens, so the following is a valid SimpleLogic statement:

```
do
  do
    x = true;
  while (x);
while (z );
```

1.2.3 Block Statement

A block Statement consists of a list of zero or more statements within braces. So, the following is a legal block statement:

```
{ x = true;
  y = z;
  z = w;
}
```

The following is also a legal block statement:

```
{
  x = true;
  do {
    w = y;
    y = true;
  }
  while (z);
}
```

Finally, the following is also a legal block statement:

```
{}
```

1.2.4 while statement

A while statement is of the form

```
while (<expression>) <statement>
```

where <expression> is an expression as defined above (identifier, boolean literal, parentheses allowed), and a <statement> is any SimpleLogic statement. Thus the following are all legal SimpleLogic statements:

```
while (x) x = false;
```

```
while (y)
  do
    x = false;
  while(z);
```

```
while (true) {
  x = true;
  y = false;
}
```

A program in the SimpleLogic programming language consists of a single statement (which could be a block statement, and could have arbitrarily many statements nested inside of it).

1.3 Examples

1.3.1 Valid SimpleLogic Programs

```
{ x = true;
  { y = z;
    w = carpark;
  }
}
```

```

do {
    train = wreck;
    while (parsnip)
        cat = dog;
} while (foolsgold);
}

```

and

```
{ }
```

and

```
x = true;
```

1.3.2 Invalid SimpleLogic Programs

Missing a }:

```

{ x = true;
  { y = z;
    w = carpark;
  }
}

```

Missing two ;'s:

```

{ x = true;
  y = z;
  do x = true while (x) }

```

Numbers are not valid in indentifiers:

```
{ x = ident1fi3r; }
```

2 Assignment

Write a java program `MiniParser.java` that takes as a command line parameter the name of the input file. Your program should either write “valid program” or “invalid program” to standard out, depending upon whether the input file contains a syntactically correct SimpleLogic program. Your program does not need to identify what the errors are, or how many errors there are – it just needs to return if the entire program is valid or not.

Recursion is your friend in this assignment! As noted at the beginning of this assignment, you should start early, and come by my office with any questions.

3 Submission

Submit your project to subversion, to the directory

<https://www.cs.usfca.edu/svn/<username>/cs414/MiniParser/>.