

# Teaching Announcements

2010/11

[< Previous](#)

[Next >](#)

## COMP204 - PRACTICAL ASSIGNMENT 2 (2011)

25/03/2011

For the second practical assignment you are asked to write a Java program to implement a simple parser and symbol table.

In many programming languages, certain constructs can be combined at will to arbitrary complexity. The rules defining how this may be done are best stated recursively. The following BNF syntax shows how an assignment and arithmetic expression might be defined:

```

<statement> ::= <iden><assign><expression><terminal> | END
<expression> ::= <term> | <expression><addop><term>
<term> ::= <primary> | <term> <multop> <primary>
<primary> ::= <unsignednumber> | <iden> | ( <expression> )
<unsignednumber> ::= <digit> | <unsignednumber> <digit>
<digit> ::= 0 | 1 | 2 | ... | 9
<iden> ::= <letter>
<letter> ::= a | b | c | ... | y | z
<multop> ::= * | /
<addop> ::= + | -
<assign> ::= =
<terminal> ::= ;
    
```

Items in angle brackets <> are constructs in the language. They are defined in terms of themselves or other constructs. The symbol ::= means 'is defined as.' The symbol | means 'or.' The symbol END corresponds to the string "END".

Your task is to write a Java program which determines whether one or more given character string is a legal expression according to the rules above, by using a top-down recursive parsing approach. When a statement includes an assignment (i.e. <iden>=<expression><terminal>) then the <iden> should be added into the symbol table. Any <iden> characters within an expression (i.e. to the right of an assignment symbol) should already exist within the symbol table.

The symbol table can be a simple, fixed sized array that \*only\* stores the idens that are defined. There is no need to store values, or to evaluate the arithmetic expressions.

The program should be called Parse.java (spelt exactly as shown), and it should take in multiple strings as input when it runs, for example:

```

a=44;
c=9*7;
b=32*(2+c);
d=a*b+(c-4);
    
```

For each statement, if the statement is legal, then the output from your program should be the single uppercase word "TRUE". If the statement is illegal, then the output from your program should be the single uppercase word "FALSE". Input should continue until the string END is typed in, in which case you should display the symbol table (see sample output opposite).

You may assume the expression contains no spaces, and that all operands are single lower-case characters.

Your program should consist of a set of mutually recursive methods corresponding to the items in the grammar. To get you started, here is the



## Deadline:

3pm, Fri 8th April '11



**A PDF (printable) version of the assignment is also available**

## Sample Output

### Sample Run 1

```

Please input a string to be checked
a=44;
TRUE
b=5*3;
TRUE
c=a/23;
TRUE
a*b+(c-4);
FALSE
-4
FALSE
END
    
```

### Symbol Table

```

=====
Symbol iden 0 = a
Symbol iden 1 = b
Symbol iden 2 = c
    
```

### Sample Run 2

```

Please input a string to be checked
f=3*(2+(435*9));
TRUE
h=g
FALSE
END
    
```

### Symbol Table

```

=====
Symbol iden 0 = f
    
```

outline for an Expression method:

```
begin
  if term(string) then TRUE;
OTHERWISE
  for each <addop> in the string
    if expression(start of string up to addop) and term(string after addop)
      then TRUE;
OTHERWISE FALSE
end
```

Note how the structure of the method closely follows the recursive language syntax. Your other methods should do likewise.

## SUBMISSION INSTRUCTIONS

Firstly, check that you have adhered to the following list:

1. All of your code is contained in a single file. Do NOT use more than one file. The file's name MUST be 'Parse.java' (capital P; lower-case everything else). This means that the main class name must also be 'Parse'.
2. Your program is written in Java, not some other language.
3. Your file is a simple text file; it must not be compressed or encoded in any way.
4. The code for your expression method is based on the algorithm given above, your other methods follow a similar structure and you have thoroughly tested your program with both legal and illegal expressions.
5. Your program compiles and runs on the computer science department's Windows system. If you have developed your code elsewhere (e.g. your home PC), port it to our system and perform a compile/check test before submission. It is your responsibility to check that you can log onto the department's system well in advance of the submission deadline.
6. Your program does not bear undue resemblance to anybody else's! Electronic checks for code similarity will be performed on all submissions and instances of plagiarism will be severely dealt with. The rules on plagiarism and collusion are explicit: do not copy anything from anyone else's code, do not let anyone else copy from your code and do not hand in 'jointly developed' solutions.

To submit your solution you must **PRINT IT OUT AND SUBMIT IT ELECTRONICALLY**, and adhere to the following instructions:

### Printout:

- The printouts required are the source code for your Java program and the output it produces (i.e. no design documentation or test files are required: ***I will test your program to see if it works***)
- You must fill in a "Declaration on Plagiarism and Collusion Form", available from the One-Stop Student Shop, and attach this form to your printouts. (Work will NOT be marked unless accompanied by this form.)
- You must submit the above form and printouts of your work to the One-Stop Student Shop, room G09 on the ground floor of the Ashton Building before the deadline given above.

### Electronic submission:

- Your code must be submitted to the departmental electronic submission system at: <http://cgi.csc.liv.ac.uk/cgi-bin/submit.pl?module=comp204>
- You need to login in to the above system and select 'Practical 2' from the drop-down menu. You then locate the file containing your program that you wish to submit, check the box stating that you have read and understood the university's policy on plagiarism and collusion, then click the 'Upload File' button.

Work will be accepted only if it is submitted both electronically AND in the form of a printout plus plagiarism declaration form, following the above

## Lab Sessions

Labs have been arranged to allow for students to work on the assignment, and to request assistance from the Lab tutors.

Attendance is based on surname (as listed below); please try to attend the sessions allocated. If there are free terminals during other sessions, then you are welcome to use these, but do not prevent others from attending their allocated labs.

### Surnames A-F

Tuesday: 09.00 - 10.00 (H105, GH)

### Surnames G-K

Tuesday: 10.00 - 11.00 (H105, GH)

### Surnames L-Q

Thursday: 11.00 - 12.00 (H116, GH)

### Surnames R-Z

Thursday: 12.00 - 13.00 (H116, GH)

## MARKING SCHEME

Below is the breakdown of the mark scheme for this assignment. Each category will be judged on the correctness, efficiency and modularity of the code, as well as whether or not it compiles and produces the desired output.

### Implementation:

**statement method = 10 marks**  
**expression & term method = 10 marks**  
**primary method = 10 marks**  
**unsignednumber method = 10 marks**  
**addop, multop, letter, iden and digit methods = 15 marks**  
**symbol table class = 25 marks**

**Execution and Output = 10 marks**  
**Comments and layout = 10 marks**

This assignment contributes 10% to your overall mark for Comp 204.

instructions.

Finally, please remember that it is always better to hand in an incomplete piece of work, which will result in some marks being awarded, as opposed to handing in nothing, which will guarantee a mark of 0 being awarded. Demonstrators will be on hand during the Comp 204 practical sessions to provide assistance, should you need it.