

COP4020 Programming Assignment 9 and 10

The Ultimate Calculator

Part 1 of this assignment (Assignment 9) is to produce a calculator that combines the identifier/assignment capability from Assignment 6 and the power operator from Assignments 5 and 7. Part 1 is necessary to shape up your code for Part 2 (Assignment 10), which is to add exception handling to the ultimate calculator.

Prepare for this assignment by copying your source file `Pr6.java` to `UCalc.java`. Change the class name in the new copy so that it will compile using the command

```
javac UCalc.java
```

and run using the command

```
java UCalc [filename].
```

All subsequent modifications should be made to `UCalc.java`. The deliverable for both assignments is the file named `UCalc.java`.

Part 1: Programming Assignment 9

Begin this assignment by making all changes necessary to your `UCalc.java` program so that it functions correctly (both theoretically and practically). You may consult the posted solutions and evaluation reports for previous assignments to accomplish this revision.

Extend the `UCalc.java` program with the power operator \wedge , as introduced in Assignment 5 and continued in Assignment 7. You will need to modify the BNF grammar to accommodate the new operator (as done for Assignment 5) and annotate this new grammar with semantic rules (as done for Assignments 6 and 7). Be sure to include the complete annotated grammar for the `UCalc` in the header documentation as well as the appropriate productions and semantic rules as documentation for the implementing subroutines. Also, be sure that your implementations conform to the documentation.

Compile, run, and test your program with at least these expressions:

```
Test Expressions
-----
let x = 1;
(let x = 1) + x;
(let a = 2) + 3*a - 5;
(let x = (let y = (let z = 1))) + x + y + z;
1+(let x = 1)+(let y = 2)+(1 + x)*(1 + y)-(let x = y)-(let y = 1)-x;
1 + (let a = (let b = 1) + b) + a + 1;
(let a = (let a = (let a = (let a = 2) + a) + a) + a) - 9;
(let x = 2)^(let y = 3);
(let y = 3)^(let x = 2);
```

(Correct responses are 1, 2, 3, 4, 5, 6, 7, 8, and 9 respectively.)

Programming hints are available from

<http://www.cs.fsu.edu/~engelen/help.txt>

Part 2: Programming Assignment 10

Add Java exception handling to your UCalc.java code by defining two exception classes `SyntaxError` and `RuntimeError`. A `SyntaxError` exception should be thrown when an illegal character is found, a closing `)` is not found, or a `=` is not used in a `let` expression. A `RuntimeError` exception should be thrown when an identifier is encountered for which no value can be found. These will be the exceptions tested.

Include in the file documentation a list of each type of exception that your program throws. If you find other cases appropriate for exceptions of either type, and implement them, include these in the documentation lists for possible extra credit.

The exceptions should propagate the error to the main program which prints the diagnostics of the error. You must handle these errors using Java exceptions and the message should be printed by a Java exception handler in a `catch` clause.

Compile, run, and test your program with at least these expressions:

Test Expression	Correct Response
-----	-----
1+(2*3;	syntax error: ')' expected
(let x 5) + x;	syntax error: '=' expected
(let x = 5) (let y = 6);	syntax error: operator expected
(let x = 5 let y = 6);	syntax error: ')' expected
(ler x = 5) ^ (let y = 6);	runtime error: 'ler' undefined
(let x = 5) + y;	runtime error: 'y' undefined

Submit this final version of the UCalc.java code.