CS5230 ASSIGNMENT 1

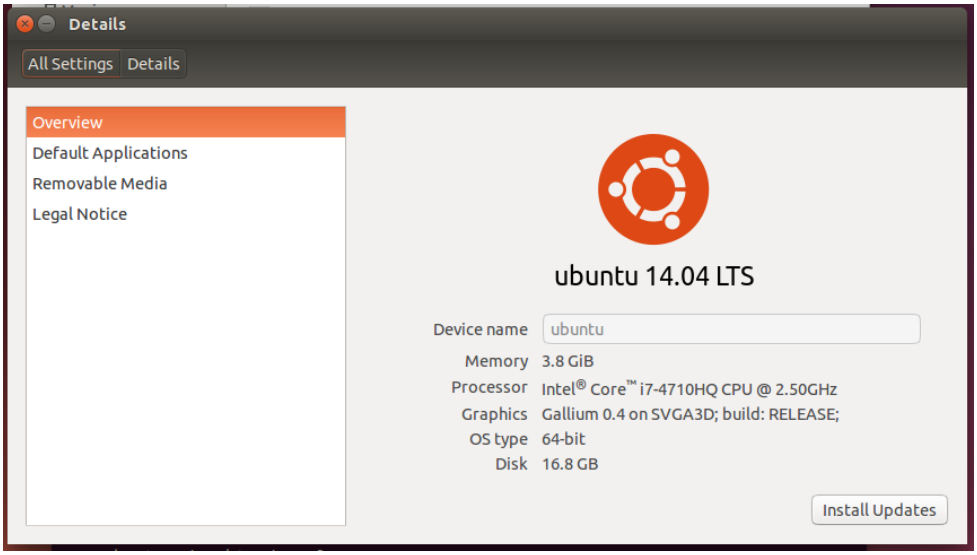
Name: Yang Mo Matric No: A0091836X

1. Submission info:

YangMo-assignment-1.tar.gz

2. System Specification

This program assignment is tested and run with an Ubuntu system running on and windows 10 laptop using a virtual machine software. Here is the screenshot of the system:



3. Tools used:

1. Java 1.8.0_72

```
mo@ubuntu:~/Desktop/cup$ java -version
java version "1.8.0_72"
Java(TM) SE Runtime Environment (build 1.8.0_72-b15)
Java HotSpot(TM) 64-Bit Server VM (build 25.72-b15, mixed mode)
```

2. Sublime Text 2 Editor

scanner.java | Calculator.java

```java
/*
  Author: Yang Mo
  Matric No : A0091836X
*/

import java_cup.runtime.*;

public class Calculator {

    public static void main(String[] argv) throws Exception{

    System.out.println("\n========================================");
    System.out.println("\nPlease type your arithmethic expression:");
    System.out.println("\n========================================\n");
    Parser p = new Parser(new scanner());
    p.parse();
    }
}
```

4. File Structure:

Name	Size	Type	Modified
jar	2 items	Folder	Jan 27
calc.cup	3.6 kB	Text	06:50
Calculator.java	395 bytes	Text	Jan 27
makeCalc.sh	165 bytes	Program	06:12
Readme.txt	270 bytes	Text	07:22
scanner.java	5.7 kB	Text	07:13

Name	Size	Type	Modified
java-cup-11b.jar	119.8 kB	Archive	Oct 1 2015
java-cup-11b-runtime.jar	28.8 kB	Archive	Oct 1 2015

java-cup-11b.jar
Jar file used for Parser generation

java-cup-11b-runtime.jar
Jar file used for Parser runtime

calc.cup
CUP specification file for the simple calculator

calculator.java
Program main entry point for the calculator

scanner.java
Customized scanner class for scanning user input

makeCalc.sh
Shell script to

- generate parser.java, sym.java
- compile all java classes
- run the calculator

Readme.txt
Running instructions

5. Running Instructions

First navigate to the directory /cup

There are two ways to compile and run the calculator:

1. Running the shell script

```
mo@ubuntu:~/Desktop/cup$ bash makeCalc.sh
```

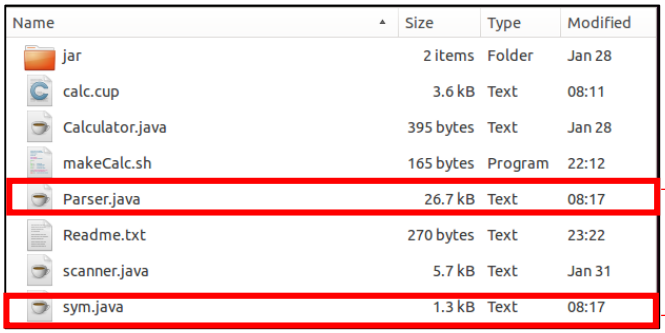
```
------- CUP v0.11b 20150930 (SVN rev 66) Parser Generation Summary -------
  0 errors and 0 warnings
  21 terminals, 4 non-terminals, and 21 productions declared,
  producing 44 unique parse states.
  0 terminals declared but not used.
  0 non-terminals declared but not used.
  0 productions never reduced.
  0 conflicts detected (0 expected).
  Code written to "Parser.java", and "sym.java".
---------------------------------------------------- (CUP v0.11b 20150930 (SVN r
ev 66))
Note: Some input files use or override a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

========================================

Please type your arithmethic expression:

========================================
```

2. Run the commands step by step (commands can be found in Readme.txt):
   a. java -jar jar/java-cup-11b.jar -interface -parser Parser calc.cup



Name	Size	Type	Modified
jar	2 items	Folder	Jan 28
calc.cup	3.6 kB	Text	08:11
Calculator.java	395 bytes	Text	Jan 28
makeCalc.sh	165 bytes	Program	22:12
Parser.java	26.7 kB	Text	08:17
Readme.txt	270 bytes	Text	23:22
scanner.java	5.7 kB	Text	Jan 31
sym.java	1.3 kB	Text	08:17

Parser.java and sym.java are generated

   b. javac -cp jar/java-cup-11b-runtime.jar:. *.java
      Compile the java files
   c. java -cp jar/java-cup-11b-runtime.jar:. Calculator

      Run the calculator

6. Supported Features:

1. Ease of use, no need for ';' at the end of expression. Just type and hit enter.

```
========================================

Please type your arithmethic expression:

========================================

10+2*log(10,100)-sqrt(-16+100/5)
= 12.0
```

2. Various forms of input number

- Floating point numbers

```
1.234-0.234
= 1.0
```

- Exponent form of floating points

```
1.23e2 - 23
= 100.0
```

- Hexadecimals a la C style

```
0xdeadbeef-0xa * 10/10 + log(10,10)
= 3.73592855E9
```

- Integers in decimal

```
-123+23+10/2+100*(9-(1+2))
= 505.0
```

3. Various functions (Including advanced functions)

- Basic Functions +, -, *, /, ^, (,)

```
1+2-3*4/5 + (12-2)*((12-2)/(6-1))^2
= 40.6
```

- Advanced: sqrt [Square Root]

```
sqrt4 + sqrt((sqrt100)^2)
= 12.0
```

- Advanced: log(Base, Input) [log function with any Base]

```
log(2,8)+log(10,1000)+log(16,1)
= 6.0
```

- Advanced: ln [log function with Base as e]

```
ln4+ln2+ln1
= 2.0794415416798357
```

- Advanced: lt [log function with Base as 10]

```
lt10+lt100+lt(999+1)
= 6.0
```

- Advanced: sin, cos, tan and negative sign

```
sin45/cos45 + -tan(-5+50)
= 0.0
```

7. Points to elaborate

- To implement the feature of “type and hit enter” without the need of adding the semicolon “;” and the end of the expression. At first it appeared that the workflow of parser and scanner are not so clear and the first few attempts failed. But in the end, by utilizing and flag `isExpressionStarted` this issued was solved.
- Due to the requirement that the calculator should support various types of input number, how to compute the result from those different types of numbers became an issue. To make the result precise and to allow mixed use of different types of numbers in the same expression, I decided to scan and convert numbers of all possible types and then convert them all into Double for computation.
- `log(a,b)` function takes two input numbers and both of them are after the operator “`log`”. Hence it cannot be treated like `a + b` by `expr:e1 PLUS expr:e2` . To solve this, an new terminal `COMMA[,]` is introduced to allow grammar like below:

```
/* log(Base, Input) eg. log(10,100) --> Result is 2 */
LOG LPAREN expr:e1 COMMA expr:e2 RPAREN
  {: RESULT = Math.log(e2)/Math.log(e1); :}
|
```

8. Calculator Cheat sheet (A & B are expressions):

Function	Meaning	Usage	Example
+ - * /	basic ones	A+B	1+2–3*4/5
^	power	A^B	2^4, (2+1)^(2+2)
%	mod	A%B	10%3
log	log	log(A,B), Base is A	log(10,100), log((11-1), 100)
ln	log with base e	lnA, Base is e	ln3, ln(100-2)
lt	log with base 10	ltA, Base is 10	lt10, lt(101-1)
sqrt	square root	sqrtA	sqrt4, sqrt(99+1)
sin,cos,tan	trigonometric	sinA, cosA, tanA	sin30, sin(20+10)
Negative sign -		-A	-10,-20
(,)	parenthesis	(A)	log((2+2),4^4)