**CS5214 –DESIGN OF OPTIMISING COMPILERS**

**Programming Assignment 1**

|  |  |
| --- | --- |
| Submitted By | Victor Domingo dela Cruz |
| Matriculation ID | A0120578E |
| Submitted File Name | victor-delacruz-assignment-1.tar.gz |
| Submission Date | 05 Feb 2016 |

Table of Contents

1 Description of Assignment 1 3

2 Solution Package Overview 3

3 Solution Description 4

4 Scanner 5

4.1 Supported Symbols and Operations 5

4.2 Supported Numbers 5

5 Scanner 6

5.1 Supported Symbols and Operations 6

5.2 Supported Numbers 6

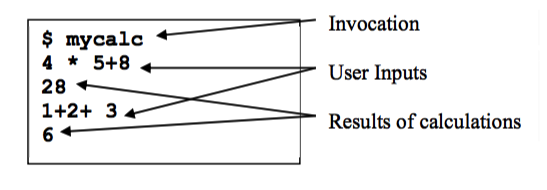
6 Lessons Learned 7

6.1 Management Challenges 7

6.2 Technical Challenges 7

# Description of Assignment 1

Using JLex (or JFlex) and CUP, implement a simple calculator that will parse an arithmetic expression and return the result. Here is an example of the behaviour of your program:



# Solution Package Overview

The following tree view shows the key structure of the submitted implementation solution for the programming assignment.

.

├── Calculator /\* main folder \*/

│   ├── Calculator /\* package contains the compiled Calculator classes \*/

│   ├── Calculator.cup /\* CUP input specification \*/

│   ├── Calculator.lex /\* JLex input specification \*/

│   ├── JLex /\* folder contains JLex source and classes \*/

│   │   ├── Main.java /\* JLex source code

│   ├── JavaCup /\* contains all CUP related files and binaries \*/

│   │   ├── java\_cup /\* package contains the java\_cup classes \*/

│   ├── README /\* file describing compilation and invocation \*/

│   ├── Yylex.java /\* scanner file, renamed output from JLex \*/

│   ├── mycalc /\* bash shell script to compile and run everything in one go \*/

│   ├── parser.java /\* parser code, output from CUP \*/

│   ├── setenv /\* adds Calculator dir to ${PATH} in bash \*/

│   └── sym.java /\* symbol code, output from CUP \*/

# Solution Description

The solution is broken down into two main process components, the **Scanner** and the **Parser**. The specifications for each are described in details in the later sections.

* The ***scanner*** is implemented using the ***JLex*** lexical analyzer framework.

|  |  |  |
| --- | --- | --- |
| **Process Steps** | **Command** | **Remarks** |
| 1. For platform compatibility, the JLex source code is compiled prior to initial use | ***> javac -target 1.8 JLex/Main.java*** | where: -target 1.8 option corresponds to the java –version installed on the machine (Java 8 in this case) |
| 1. A file **Calculator.lex** is provided as an input specification |  | Refer to Scanner section for details |
| 1. The specification is then fed to the JLex scanner generator program to generate the customize scanner program | ***> java JLex.Main Calculator.lex*** | This step generates the java scanner program named Calculator.lex.java |
| 1. Rename the output file from last step to match class name later use with the parser program | ***> mv -f Calculator.lex.java Yylex.java*** | Where “-target 1.8” option corresponds to the java –version installed on the machine (Java 8 in this case) |

* The ***parser*** is implemented using the Java-based ***Constructor of Useful Parsers (CUP)***.

|  |  |  |
| --- | --- | --- |
| **Process Steps** | **Command** | **Remarks** |
| 1. A file Calculator.cup is provided as an input specification to CUP |  | Refer to Parser section for details |
| 1. The .cup file needs to be compiled using the .jar file provided in CUP framework | ***> java -cp JavaCup java\_cup.Main Calculator.cup*** | This step generates the java parser programs named *parser.java* and *sym.java* |
| 1. Compile the generated files together with the renamed scanner file from scanner section above | ***> javac -cp .:JavaCup -d . parser.java sym.java Yylex.java*** | This step generates the java scanner program |
| 1. The program is now ready to use and may be invoked using this command | ***> java -cp .:JavaCup Calculator.parser*** | Where “-target 1.8” option corresponds to the java –version installed on the machine (Java 8 in this case) |

* A script is provided as an alternative method for compilation and invocation for UNIX bash:

|  |  |  |
| --- | --- | --- |
| **Process Steps** | **Command** | **Remarks** |
| 1. Include “.” in $PATH environment var | ***> source setenv*** | Run from the root Calculator/ folder |
| 1. Compile and run | ***> mycalc*** | Modify the target –version accordingly |

# Scanner

The scanner module directly interacts with the user and is implemented using the **Jlex** framework. The download link for the source code is given below:

[*http://www.cs.princeton.edu/~appel/modern/java/JLex/current/Main.java*](http://www.cs.princeton.edu/~appel/modern/java/JLex/current/Main.java)

## Supported Symbols and Operations

The following table list downs the supported symbols, their meaning and example usages

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Symbol** | **Functionality** | **Examples** |  | **Symbol** | **Functionality** | **Examples** |
| + | Addition | 1+1e-3; 0xde + 2.5; |  | ~ | Bitwise negation | ~1; ~0xdead; |
| - | Subtraction | 1-1e-3; 0xde - 2.5; |  | << | Shift left | 1<<0xde; |
| \* | Multiplication | 1\*1e-3; 0xde \* 2.5; |  | >> | Shift right | 1>>0xde; |
| / | Division | 1/1e-3; 0xde / 2.5; |  | & | Bitwise AND | 1&0xde; |
| % | Modulo | 0xdeadbeef % 21; |  | ^ | Bitwise XOR | 1^0xde; |
| sqrt | Square root | sqrt(2); sqrt1e5; |  | | | Bitwise OR | 1|0xde; |
| log | Logarithmic | log(2); log1e5; |  | ^^ | Exponentation | 1^^1e-3; 0xde^^2.5; |
| sin | Sine | sin(2); sin1e5; |  | cot | Cotangent | cot(2); cot1e5; |
| cos | Cosine | cos(2); cos1e5; |  | sec | Secant | sec(2); sec1e5; |
| tan | Tangent | tan(2); tan1e5; |  | csc | Cosecant | csc(2); csc1e5; |
| ( ) | Precedence | (1+2)/3-4\*5; |  | ; | End-of-line | See above |

## Supported Numbers

The following table list downs the supported number types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Notation** | **Types** |  | **Notation** | **Types** |
| 0-9 | Decimal digits |  | 0-9a-f | Hexadecimal digits |
| …,-1,0,1,… | Long integers |  | 1.23 | Floating numbers |
| 0x | Hexadecimal Notation |  | 1.23e-5 | Scientific notation |

# Parser

The scanner part directly interacts with the user and is implemented using the **Jlex** framework. The download link for the source code is given below:

[*http://www.cs.princeton.edu/~appel/modern/java/JLex/current/Main.java*](http://www.cs.princeton.edu/~appel/modern/java/JLex/current/Main.java)

## Supported Symbols and Operations

The following table list downs the supported symbols, their meaning and example usages

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Symbol** | **Functionality** | **Examples** |  | **Symbol** | **Functionality** | **Examples** |
| + | Addition | 1+1e-3; 0xde + 2.5; |  | ~ | Bitwise negation | ~1; ~0xdead; |
| - | Subtraction | 1-1e-3; 0xde - 2.5; |  | << | Shift left | 1<<0xde; |
| \* | Multiplication | 1\*1e-3; 0xde \* 2.5; |  | >> | Shift right | 1>>0xde; |
| / | Division | 1/1e-3; 0xde / 2.5; |  | & | Bitwise AND | 1&0xde; |

# Lessons Learned

## Management Challenges

Due to unforeseen circumstances that happened within the team members, the development portion encounter shorthanded and directly caused the project slippage. Towards the end of the project phase, the team managed to get project extension and completed the remaining outstanding portions on time.

## Technical Challenges

Open source survey module was chosen after evaluated the survey functionality are fairly common. However, the next challenge is to choose the most suitable open source that can fit our system best. After spending tons of time in research and evaluate multiple open sources in the market, including those in java, .NET and PHP, we finally selected Lime Survey which is in PHP. Some difficulties were encountered during the integration between this PHP code and our main system that written in.NET, but those problems were gradually overcome.