Swinburne University Of Technology

Faculty of Science, Engineering and Technology

ASSIGNMENT COVER SHEET

Subject Code:	CSO30023
Subject Title:	Languages in Software Development
Assignment number and title:	5, JavaCC – First Steps
Due date:	**September 22, 2014, 10:30, on paper**
Lecturer:	Dr. Markus Lumpe

Your name:_____

Marker's comments:

Problem	Marks	Obtained
1	-	-
2	72	
Total	72	

Extension certification:

This assignment has been given an extension and is now due on _____

Signature of Convener:_____

Assignment 4

COS30023 - Languages in Software Development

Daniel Parker - 971328X

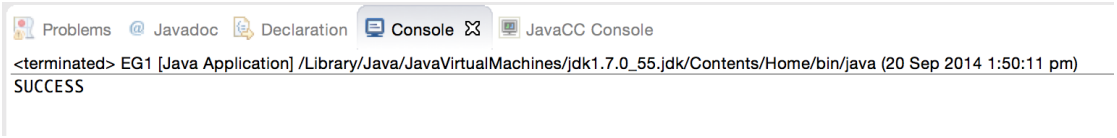
September 20, 2014

1. Parsing

1.1. Test Input

```
a := 5 + 3; b := {print (a, a-1), 10 * a}; print (b)
```

1.2. Parser Result



2. Source

```
/**
 * JavaCC template file created by SF JavaCC plugin 1.5.17+ wizard for JavaCC 1.5.0+
 */
options
{
  JDK_VERSION = "1.7";
  static = false;
}

PARSER_BEGIN(StraightLine)

import java.io.*;

public class StraightLine
{
        public static void main( String[] Args ) {
                try {
                        StraightLine lParser = new StraightLine( new FileInputStream( Args[0] ) );
                        lParser.CompilationUnit();
                        System.out.println( "SUCCESS" );
                } catch (ParseException e) {
                        System.out.println( "Syntax Error : \n"+ e.toString() );
                } catch (FileNotFoundException e) {
                        System.out.println( e.toString() );
                }
        }
}
```

```
PARSER_END(StraightLine)

SKIP :
{
        " "
|
        "\r"
|
        "\t"
|
        "\n"
|
        "\r\n"
|
        < "//" (~["\n"])* "\n">
}

void CompilationUnit():
{}
{
        Statements() < EOF >
}

void Statements():
{}
{
        Statement() (";" Statement())*
}

void Statement():
```

```
{}
{
                <IDENTIFIER> ":=" Expression()
        |
                "print" "(" ExpressionList() ")"
}

void Expression():
{}
{
        Term() (("+" | "-") Term())*
}

void Term():
{}
{
        Primary() (("*" | "/") Primary())*
}

void Primary():
{}
{
                <IDENTIFIER>
        |
                <INTEGER>
        |
                "(" Expression() ")"
        |
                "{" Statement() "," Expression() "}"
}
```

```
void ExpressionList():
{}
{
        Expression() ("," Expression() )*
}

TOKEN :
{
          < INTEGER: (["0"-"9"])+ >
|
        < IDENTIFIER: ["a" - "z", "A" - "Z"](["a"-"z", "A"-"Z", "0" - "9", "_"])* >
}
```