DCU University's Declaration on Plagiarism
Assignment Submission Form

This form must be filled in and completed by the student submitting an assignment. Assignments submitted without the completed form will not be accepted.

Name: Eoghan McMullen
Programme: CASE 4
Module Code: CA4003
Assignment Title: Assignment 1
Submission Date: 14/11/2015

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. I have read and understood the Assignment Regulations set out in the module documentation. I have identified and included the source of all facts, ideas, opinions, and viewpoints of others in the assignment references. Direct quotations from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the source cited are identified in the assignment references. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.
I have read and understood the referencing guidelines found at http://www.library.dcu.ie/citing&refguide08.pdf and/or recommended in the assignment guidelines.

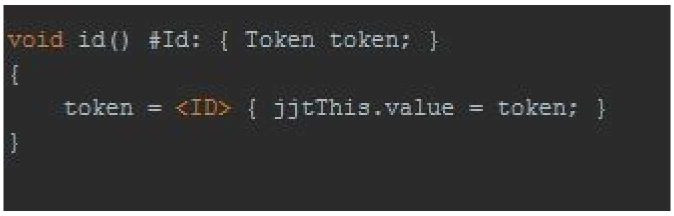
Name: Eoghan McMullen Date: 14/11/2015

Abstract Syntax Tree

JJTREE File

To create the AST the .jj file had to be converted to a .jjt. My .jj file needed to be altered slightly to accomadate creating the .jjt. Certain production rules needed to be broken down in order to allow them to be returned as tokens.

For example here <id> is returned as a Token:



```
void id() #Id: { Token token; }
{
    token = <ID> { jjtThis.value = token; }
}
```

As pointed out in the tutorial video I set the `NODE_DEFAULT_VOID = true;`
This meant that all rules were passed through to a visitor without specifically declaring a hashtag.

Once the .jjt file was ready the AST could be printed.

The SimpleNode `program.dump(" ");` method was used to print out the AST, hashtags were added to appropriate production rules. Simple node extracted data from the chosen production rules and printed it

On compiling the jjt fle a visitor interface was created. A vistor file which implemented the visitor interface was created which could print a Symbol table and deal with symantic checks was created. node values were returned from functions within the visitor file `BasicLSemanticVisitor,java`such as number:

```
public Object visit(ASTNumber node, Object data)
{
  return node.jjtGetValue();
}
```

ASTID() returns a id nodes values, as seen below:

```
public Object visit(ASTId node, Object data)
{
  return node.jjtGetValue();
}
```

Id’s were stored in an children Array in SimpleNode. These id were all extracted from the array in SimpleNode and added to a list of tokens which gets returned.

```
public Object visit(ASTIdList node, Object data)
{
  List<Token> tokens = new ArrayList<>();
  for (int i = 0; i < node.jjtGetNumChildren(); i++)
  {
      SimpleNode oNode = (SimpleNode) node.jjtGetChild(i);
      Token id = (Token) oNode.jjtGetValue();
      tokens.add(id);
  }
  return tokens;
}
```

After creating a list of Tokens they can be called in a function to run through them while running Semantic checks and add them to the Symbol table.

For example the function `public Object visit(ASTconst_decl node, Object data)` was used to to go through the id’s from the list of Tokens and store them to the STC object. They were added to the symbol table from there.

The Symbol table was constructed with the following data structure: `HashMap<String, HashMap<String, STC>>`
This allows the symbol table to be indexed by scope(e.g. global etc.). Once a scope has been accesed then the functions and variables can be accessed.

scope will change depending on which method is being used(could be in main etc) Default is always “global” and changes appropriately.

Each time a new main or function gets entered a scope gets added into a symbol table along with a HashMap.

STC structure:

Has a contructor and list of methods to return and store data to and from STC’s.

Semantic Check

My code will check to see if an id is already within the current or Global scope. Error messages are printed when an id is found which already is declared.