CA4003 Compiler Construction

Assignment 1 description

Ciara Edwards 12308356

**Declaration**

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: Ciara Edwards Date: 08/11/15

**Resources used**

* CA4003 Notes by David Sinclair.
* “Generating Parsers with JavaCC 2nd Edition” a book written by Tom Copeland.

**Description of code**

Options

* I have Java Unicode escape= true as it translates hex values into Unicode before lexical analysis just like java files
* Ignore\_case=true as case is insignificant in BasicL.

Main

* Inside my main class I create an instance of a BasicLParser and call it parser.
* I then check the length of the argument supplied when running the program if it is 0 read from cmd else read from a file
* I then catch any file exceptions and print an error message for the user.
* The final option in the if statement catches any user who supplies an args length not equal to 0 or 1.
* Finally the parser calls my method prog() to begin the program, on success I print a success message otherwise I catch any parsing exceptions and print the error message.

Comments

* For /\* \*/ comments I declare a variable call commentNesting which increments when it see /\* as it is in a comment and decrements when /\* occurs. I have an if statement which checks the value of commentNesting and when the value is 0 it switches out of the comment.
* The second type of comment is – which is skipped until a newline or return character is read.

Skip/Keywords/Tokens

* Next I outline which characters to skip along with keyword tokens and symbols for the grammar
* I declare my static tokens which are NUM, ID and STRING to match the requirements in the assignment description.
* Lastly I handle any unrecognised tokens

Grammar rules

* Next I created each of the methods specified by the assignment description with some exceptions.
* The first exception is my condition method the main reason I split this up is to remove left recursion. My condition method splits up the “condition AND OR condition” to prevent recursion by introducing more levels of precedence. The line “notAssosiated() [(<OR> associated())?]” is an expansion unit, this is a set of one or more expansion choices, so any legal parse of the expansion unit is any parse of the nested expansion choices. So it can either run notAssosiated() or run OR associated() or an empty token sequence as the “[“ “]” symbol allows for this. The “[“ “]” symbols mean anything between these symbols can or cannot be supplied.
* The next method I created was notAssosiated() this is the same logic as the condition method only using the AND operator.
* The associated is basically the assignments condition method without the AND/OR option as they are handled above. I have placed a LOOKAHEAD for my expression as without it had a choice conflict with fragment as there are two LBR options one for condition and the other for expression. However due to the lookahead I had to place an optional function call following ID in both fragment and expression in order for function calls and if statements to parse correctly.
* The next method which I altered was expression due to a left recursion error. I got the idea to split the operators up from the CA4003 notes on removing left recursion by introducing a term method.
* My fragment is basically as outlined in the specification with the exception of ID here I give the option of a function call following an ID either with or without arguments specified.
* There are all the exceptions in my grammar rules.