Kennesaw State University

College of Computing and Software Engineering

Class: CS 4308

Section: W02

Project Deliverable 3

Author: Cohen Miller [Cmill283@students.kennesaw.edu](mailto:Cmill283@students.kennesaw.edu)

Date: 11/19/2023

Project Deliverable 3 Report

**Summary and Purpose of Assignment**

The goal of this project is to develop a complete interpreter program for a subset of the SCL language. This research paper outlines the development of the final component in this language processor, an executor that will work in conjunction with a parser and scanner program. This assignment required that the executor be developed in a language other than the one used to develop the previous assignment, so python was used.

**Initial Problem Statement**

In order to better understand the problem, the assignment was analyzed and broken down into a list of high level and low-level requirements. The high-level requirements dictate how the solution must behave, while the low-level requirements dictate some key implementation specifics that the solution must utilize.

|  |
| --- |
| High Level Requirements |
| *Must translate and execute statements from a subset of the Scl programming language*  *Must show the results of executing every statement recognized by the parser*  *Must be developed in a programming language that was not used in the previous deliverables* |

**Description of Solution**

This interpreter was developed using the grammar outlined in the Cohen’s\_SCL\_Grammar.txt file in the project folder. Since the last two components were written in java, python was selected for the final project. Consequently, both the parser and scanner from the previous deliverables were rewritten in the python language. The final solution has six unique classes. These classes are described below:

**The Scanner**

The **Scanner** class loads the code file and contains the next() function. When called by the parser, this function scans each lexeme and creates a new token object. This token object is then passed to the parser for use.

The **Token** class specifies the token object, which is used by the parser to build the parse tree. Each token contains a type and a value.

The **Type** class specifies the various types of tokens, this was used to describe the keyword list. When a new token is created, the token receives a type object depending on the value of the lexeme.

**The Parser**

The **Parser** class initializes the scanner class and builds a parse tree which will be used by the interpreter. This class contains several tools that are used by the parser, including the compare() function, and the getParseTree() function, which returns the head node of the parse tree.

The **Node** class specifies the node object, which is used to build the parse tree. Each node of the tree contains a value, type, and an array of children. The node class also includes helper functions for accessing and manipulating data inside a single node.

**The Executor**

The **Executor** class initializes the parser class and navigates the parse tree node by node. The executor class uses a python dictionary that contains information about each variable’s name, type, and data.

**List of Input Data and Results**

**sample.scl**

|  |  |
| --- | --- |
| **INPUT CODE** | **OUTPUT TO CONSOLE** |
| **import "scl.h"**  **implementations /\* multi-line-comment \*/**  **function main is**  **variables**  **define x of type double // a variable declaration**  **define t of type double // time of the second event**  **define pt of type double**  **begin**  **display "Hello ", "World!"**  **set x = 45.95**  **set t = 45.95 + 2.55**  **set pt = 10.00+ 5.00 \* 10.00**  **display "The value of x:", x**  **display "The value of t:", t**  **display "The value of pt:", pt**  **exit**  **endfun main** | **Imported scl.h**  **Hello World!**  **The value of x: 45.95**  **The value of t: 48.5**  **The value of pt: 60.0** |

**Comments and Conclusion**

Due to time constraints, only a few numbers of statements are supported by the parser and executor. Although I would’ve liked to implement some sort of control statement, such as an if statement, the interpreter framework is designed to support quick development, so implementing more statements would not be difficult.

**List of references**

**Textbook:** Concepts of Programming Languages: Concepts of Programming Languages,  
10E by Robert W. Sebesta, University of Colorado, Colorado Springs (2010,  
Pearson Education Company), ISBN-13: 9780131395312