CSL314-18 THEORY OF PROGRAMING LANGUAGES

[Semantic Analyzer]

MARASIGAN, VEM AIENSI A.
3BSCS-1

Laboratory Activity 005
October 21, 2023

SOURCE CODE

SemanticAnalyzer (main class)

```
import java.util.LinkedList;
import java.util.Scanner;

public class SemanticAnalyzer {
    public static void main(String[] args)
    {
        LinkedList<Lexeme> sentence = new LinkedList<>();
        Scanner in = new Scanner(System.in);

        System.out.println("---[ Welcome to Semantic Analyzer ]---\n" +
                "by Marasigan, Vem Aiensi : 3BSCS-1\n");

        do{
            try{
                System.out.print("Enter Source Language:  ");
                sentence = toList(in.nextLine());

                if(syntaxCheck(sentence))
                {
                    //System.out.println(sentence.get(0).getDataType()
                    //+" "+ sentence.get(sentence.size()-2).getDataType());
                    if(sentence.get(0).getDataType().equals(sentence.get(sentence.size()-2).getDataType()))
                        System.out.println("Semantically Correct!\n");
                    else System.out.println("Semantically Incorrect :(\n");
                }
                else System.out.println("Syntax is not correct");

            }catch (Exception e) {
                System.out.println("Please include a delimiter [semicolon]");
            }
        }while(true);
    }

    static LinkedList<Lexeme> toList(String input)
    {
        LinkedList<Lexeme> sentence = new LinkedList<>();

        for(int i=0, j=0; j<input.length(); )
        {
            String data = "";

            if(input.charAt(i)== ' ') {
                i++; j++;
            }
            else if(input.charAt(i) == '\'') {
                for(j += 3; i<j; i++)
                    data += input.charAt(i);
                sentence.add(new Lexeme(data));
            }
            else if(input.charAt(i) == '"') {
                while(input.charAt(j) != '"')
                    j++;

                for(;i<=j;i++)
                    data += input.charAt(i);
                j++;
                sentence.add(new Lexeme(data));
            }
            else if(input.charAt(i) == ';') {
                sentence.add(new Lexeme(";"));
                i++;
                j++;
            }
            else {
                while (input.charAt(j) != ' ' && input.charAt(j) != ';')
                    j++;

                for (; i < j; i++)
                    data += input.charAt(i);
```

```
                sentence.add(new Lexeme(data));
            }
        }
        return sentence;
    }

    static boolean syntaxCheck(LinkedList<Lexeme> sentence)
    {
        boolean pass = true;
        // this checker is still weak as it only checks 4 tokens
        if ((!sentence.get(0).getToken().equals("<data_type>")) ||
           (!sentence.get(1).getToken().equals("<identifier>")) ||
           (!sentence.get(2).getToken().equals("<assignment_operator>")) ||
           (!sentence.get(3).getToken().equals("<value>")))
            pass = false;

        return pass;
    }
}
```

Lexeme (class object)

```
public class Lexeme {
    private String lexeme;
    private String token;
    private String dataType = "String";

    public Lexeme(String lexeme)
    {
        setLexeme(lexeme);

        if (lexeme.equals("="))
            setToken("<assignment_operator>");
        else if (lexeme.equals(";"))
            setToken("<delimiter>");
        else if (lexeme.equals("+") || lexeme.equals("-") ||
                lexeme.equals("/") || lexeme.equals("*"))
            setToken("<operator>");
        else if (lexeme.equals("int") || lexeme.equals("double") ||
                lexeme.equals("String") || lexeme.equals("char"))
            setToken("<data_type>");
        else if (isValue(lexeme))
            setToken("<value>");
        else
            setToken("<identifier>");

        if(getToken().equals("<data_type>"))
            switch ((lexeme))
            {
                case "char": setDataType("Character"); break;
                case "String": setDataType("String"); break;
                case "double": setDataType("Double"); break;
                case "int": setDataType("Integer"); break;
            }

        if(getToken().equals("<value>"))
            switch (dtAnalyzer(lexeme))
            {
                case "char": setDataType("Character"); break;
                case "double": setDataType("Double"); break;
                case "int": setDataType("Integer"); break;
                default: break;
            }
    }
    static boolean isValue(String input)
    {
```

```
        boolean value = false;

        for(int i = 0; (input.charAt(i) >= 48 && input.charAt(i) <= 57) ||
                input.charAt(i) == '.' ;i++)
        {
            value = true;
            if(i == input.length()-1)
                break;
        }
        if((input.charAt(0)=='"' && input.charAt(input.length()-1)=='"') ||
                (input.charAt(0)=='\'' && input.charAt(input.length()-1)=='\''))
            value = true;

        return value;
    }

    static String dtAnalyzer(String data){
        String type = "";
        boolean hasDecimal = false;
        for(int i = 0; i<data.length(); i++)
            if(data.charAt(i) == '.') {
                hasDecimal = true;
                break;
            }

        if(data.charAt(0) == '\'')
            type = "char";
        else if(hasDecimal)
            type = "double";
        else type = "int";

        return type;
    }



    public String getLexeme() {
        return lexeme;
    }

    public void setLexeme(String lexeme) {
        this.lexeme = lexeme;
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }

    public String getDataType() {
        return dataType;
    }

    public void setDataType(String dataType) {
        this.dataType = dataType;
    }
}
```

OUTPUT

```
"C:\Program Files\Java\jdk-19\bin\java.exe" "-javaagent:D:\Programs\Intel
---[ Welcome to Semantic Analyzer ]---
by Marasigan, Vem Aiensi : 3BSCS-1

Enter Source Language:  int x = 1;
Semantically Correct!

Enter Source Language:  int x = 2.0;
Semantically Incorrect :(

Enter Source Language:  String message = "How can I be Happy?";
Semantically Correct!

Enter Source Language:  String str = 'c';
Semantically Incorrect :(

Enter Source Language:  char letter = "test";
Semantically Incorrect :(

Enter Source Language:  char letter = 'v';
Semantically Correct!

Enter Source Language:  String v = "Vergil";
Semantically Correct!

Enter Source Language:  double power = "absoloute!";
Semantically Incorrect :(

Enter Source Language:  double power = 99.99;
Semantically Correct!

Enter Source Language:  double need = "motivation";
Semantically Incorrect :(

Enter Source Language:  String ask = "Where's my motivation?";
Semantically Correct!
```

JAVA CODES HERE