19AIE112: Elements of Computing Part 2

EndSem Project – Compiler I

PROJECT REPORT

Name: R. Rakesh **Roll No.: AM.EN.U4AIE21052**

Objective:

For the EndSem project - build a Syntax Analyzer that parses Jack programs according to the Jack grammar, producing an XML file that renders the program's structure using marked-up text. Builda Basic Analyzer that handles everything.

a) Classes/Methods in Our Code

- Compilation Engine

1. CompileClass()

   Used for compiling the class. The token is obtained from the tokenizer, and then a class keyword check is performed. If not, a error is displayed.

2. CompileVarDec()

   Compiles a static or field variable declaration. The tokenizer will provide the token and check the segment of the variable, just as we did for the Compile class. i.e., if it is static or field. Error is shown if there is no segment.

3. CompileMethods()

   Compiles the complete method, class or function. It will verify that the function name, return type, parameter list, and closing bracket are all valid. It will return an error if the validation is flawed. If it is not validated, errors will be printed. The compilation of var dec and statements will then start.

4. CompileVarDecs()

   The compilation of var declarations inside a subroutine is done using this technique. First, it will look for datatype errors, and if any are found, an error message will be displayed. The Identifiers that are contained in the declaration will then be verified. If the identification name has a problem, the error method is used.

5. CompileStatements()

   Compiles a sequence of statements. Does not handle the enclosing “{}”.

6. Compilewhilestatement()

   Compiles the while statement.

7. compileIfstatement()

   compiles an if statement. Possibly with a trailing else clause.

8. compileelsestatement()

   The function will parse through the statements and display the error according to the grammar.

9. compileDoStatement()

   The do statement will be parsed first. After the symbol, it will search for the identifier name. Along with checking for expressions, it will also look for the opening brackets and jump to the parameter list if one is there.

10. compileLetStatement()

    compiles a let statement. It checks for identifier name, the equal to symbol.

11. compileReturnStatement()

    It is used for parsing Return statements.

12. compileExpressionlist()

    We parse the expression list.

13. compileExpression()

    Compiles expression of Jack language by using grammar of jack. We use a while loop and calls the function term and identify the symbols.

14. compileTerm()

    Parses through the terms present. It then checks whether the terms for errors.

19AIE112: Elements of Computing Part 2

- Jack Analyser

    1. Run()

       Run method calls compile class method and then initializes the process.

    2. Main()

       Takes input from user and the input is either an path of the file or directory.

- Tokenizer

    1. GetToken()

       It returns the current token taken from string as substrings.

    2. Advance()

       Skips comments and whitespaces in the string double quotes.

    3. Is Numeric()

       We are determining whether the token in use is numerical or not. If the answer is yes, it will return true; else, it will.

    4. TokenType()

       Returns the type of the current token, as a constant.

    5. Symbol()

       Returns the character which is the current token. It calls only if Tokentype is Symbol.

6. Keyword()

   Returns keyword which is the current token, as a constant.

7. Stringval()

   Returns the string value of the current token, without the two enclosing double quotes.

8. Identifier()

   Returns the identifier which is the current token.

9. GetTypeText()

   Returns what token is, like keyword, identifier or symbol.

10. GetTag()

    Returns final xml code line and returns the xml code.

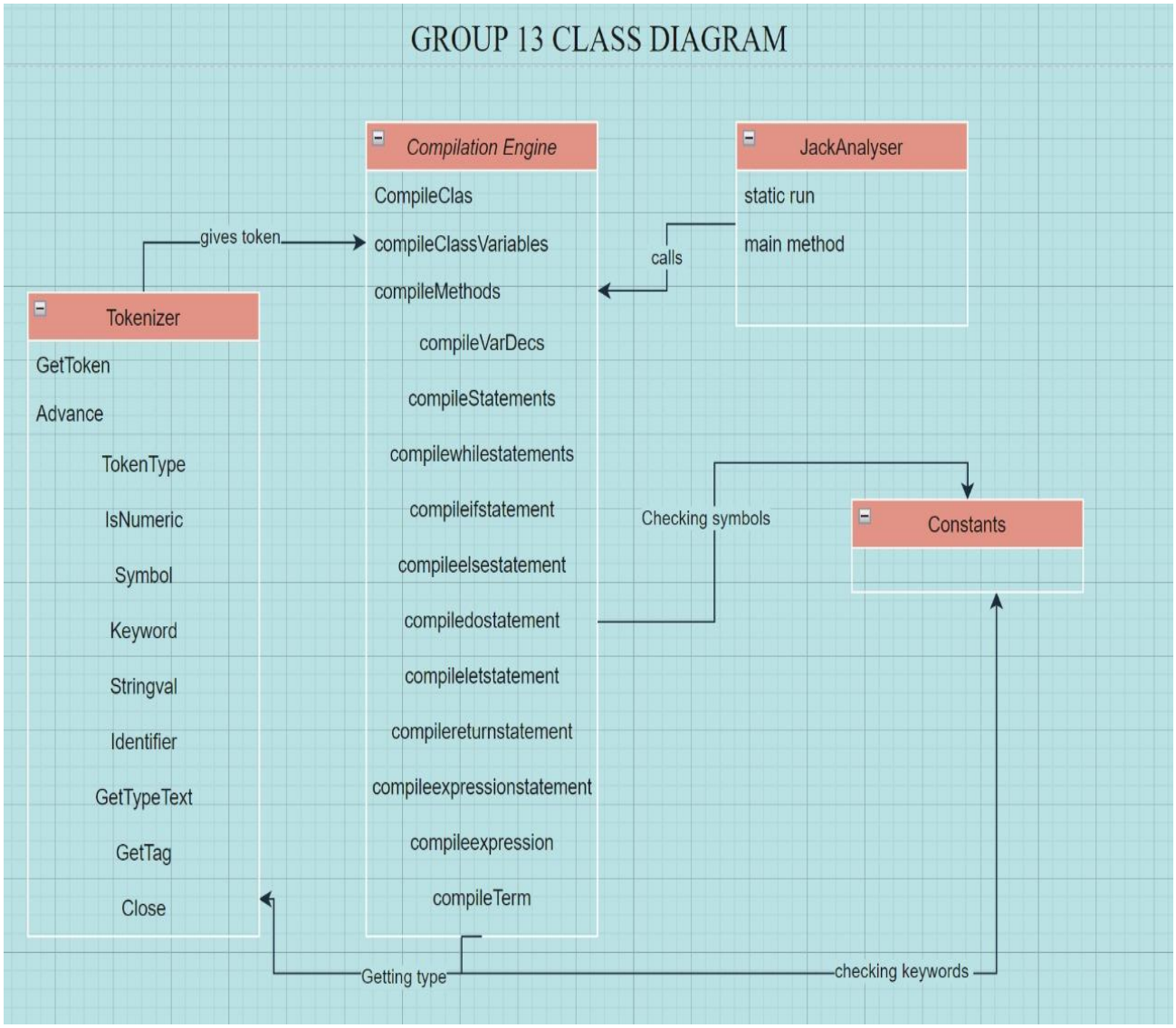
11. Close()

    Closes scan file.

12. JackTokenizer Constructor

    Opens the input .jack file and gets ready to tokenize it.

b) Class Diagram



c) Contribution

My contribution to the project we have did includes work on the Tokenizer class.

- In the tokenizer class I have worked on the GetToken, TokenType, Symbol, Stringval, GetTag & GetTypeText.
- In GetToken(), what we're doing is it returning the current token which is taken from string as substrings.
- In TokenType(), Here, we check the current tokens against a text file containing a symbols table. Using HashMap, we read the current token type from the symbols table file, and if it matches a keyword, it returns the keyword's number from the Constant class where we initialized it.
- In Symbol (), we check whether the token is numeric or not and returns a Boolean value.
- In Stringval(), used to check whether the token is a string or not. If it is not an error then we replace the '*' to null.
- GetTag(), here I have implemented it to return the final xml code.
- GetTypeText(), here I am returning what the token is like keyword, identifier or symbol etc.

d) Insights

The course gave us a better understanding of the modern software hierarchy, introduced us to the fundamentals of Boolean algebra, and helped us further comprehend how the modern computer functions. We gained knowledge of the software nand2tetris as well as all the logic gates that aided us.

We studied about many memory kinds.

A and C instructions were covered, along with the bottom level of memory hierarchy.

I had the opportunity to learn more about VM translator and Assembler. I became known that the compiler changes the code to VM and then ASM in order to hack a file (binary codes). I now understand the syntax of the Jack programming language, how to write methods and constructors, and how to tell Jack from Java in terms of usage and grammar.

I understood stack and its functions thanks to the VM chapter. I gained knowledge about what I already knew about how RAM operates, how various memory parts are split up inside, and why we need them.

e) Command to Invoke Syntax Analyser

We run the Jack Analsyer of Group 13 and then we inputs the files or file path and the output file will be outputted to the location of the source files.

21AIE112: Elements of Computing Part 2