End Semester Project

Compiler I

Anugrah Nambiar AMENU4AIE21016

Classes And Function Used:

Classes:

a. SyntaxAnalyzer:

The analyser program operates on a given source, where the source is a file path

For each source xxx.java file, the analyser would →

1.create a Tokenizer and create an output file called xxx.xml.

2.use the compilation engine to compile the input tokenizer into the output file.

b. Tokenizer:

It converts the given source file into jack language tokens and at the same time remove all the white space from file into xml language.

For example:

<keyword>class</keyword>

<identifier>main</identifier>

<symbol>{</symbol>

c. CompilationEngine:
It compiles the work which we get from the tokenizer.
The output file should contains series of code generated by compile xxx() routines and this must be implemented using jack grammer rules.

Parsed structured is the final output in the file

Function:

In syntax Analyzer

There is only a static main function which takes the file path of the input file and 2 output file to the compilation engine constructor and calls the compile class method of the compilation engine.

Tokenizer class

1.tokenizer.advance()

It checks is there is any more tokens left using the pointer and if there is it will increment the pointer and go to the next line of the input file for translation

and at the same time it keeps a record the type of current token is

whether it is a keyword, symbol, int_constant,identifier,symbol

2.pointerBack():

decrease the pointer attribute by -1 for token.advance()

3.initRegs():

It is an API in which string is defined which is used for pattern matching ,it is used for searching and manipulation in java.

4.hasmoreTokens():

check if there is anymore tokens.

5.getcurrentToken():

returns the current token

6.tokentype():

returns the type of the current token

7.keyword():

returns the current selected token keymap value from the current selected token else throw an illegalstateexception.

8.symbol():

returns the current selected token’s zero index character -keymap value from the current selected token else throw an illegalstateexception

9.identifier():

returns the current selected token’s keymap value from the current selected token else throw an illegalstateexception

10.intVal():

returns the integer value of the current token else throw an illegalstateexception.

11.stringVal():

returns the string value of the current token

else throws an illegalstateexception

12.isOp():

returns the current token’s keymap symbol value

13.nocomments(String str):

removes all the comments

14.noSpaces(String str):

removes all the white spaces

15.noBlockComments(String str):

Remove all the multiple line comments.

In compilation engine

1.compileClass():

Which writes into the file the keywords, class, identifier tokens into the file and check if there is closing curly keyword or in simple words compile the entire class until the curly closing bracket is met.

2.CompileClassVarDec():

Check if there is any variable declaration or start a subroutine.

3.Compiletype()

It creates the token codes for keyword and identifier but gives error of any other token type

4.Compilesubroutine();

Compiles the content method or constructor function by compiling the parameters,identifier,tokens and check if there is a void method.

5.CompilesSubroutinebody():

Compiles a subroutine body and check if there is { symbol

It writes the opening and closing statements<statements></statements>

It writes the opening and closing statements<subroutinebody></subroutinebody>

Token into the file

6.compileStatement():

compiles the entire sequence of statements

it doesn't handle the enclosing '{}'.

7.Compile parameterlist():

Compiles a parameter list by checking if there is a parameterlist, if the next token is ')' then go back

8.compileDo():

it is used for compiling a do statement and calls a compileSubroutineCall()

and check if there is ‘;’ symbol and produce their tokens.

9.compileExpression():

compiles the expression into xml file in form of tokens

10.compileLet():

compiles a let statement into xml file in form of tokens

11.compileWhile():

compiles a while statement into xml file in form of tokens and checks whether the symbols such as ‘(‘ or ‘)’ or ‘{‘ or ‘}’symbol and produce their tokens.

12.CompileReturn():

Compiles a return statements.

13.compileIf():

compiles an if statement and if there is an ‘else’ then it also compiles else clause also using compileStatement()

14.compileTerm():

check if it is an identifier and check if the subroutine distinguishesbetween

a variable, an array entry, subroutine call, the next token must be ‘[‘,”(“,”,”

and other token should not be a part of it.

15.compileExpressionList():

compiles a comma separated list of expression and also check if it is empty.

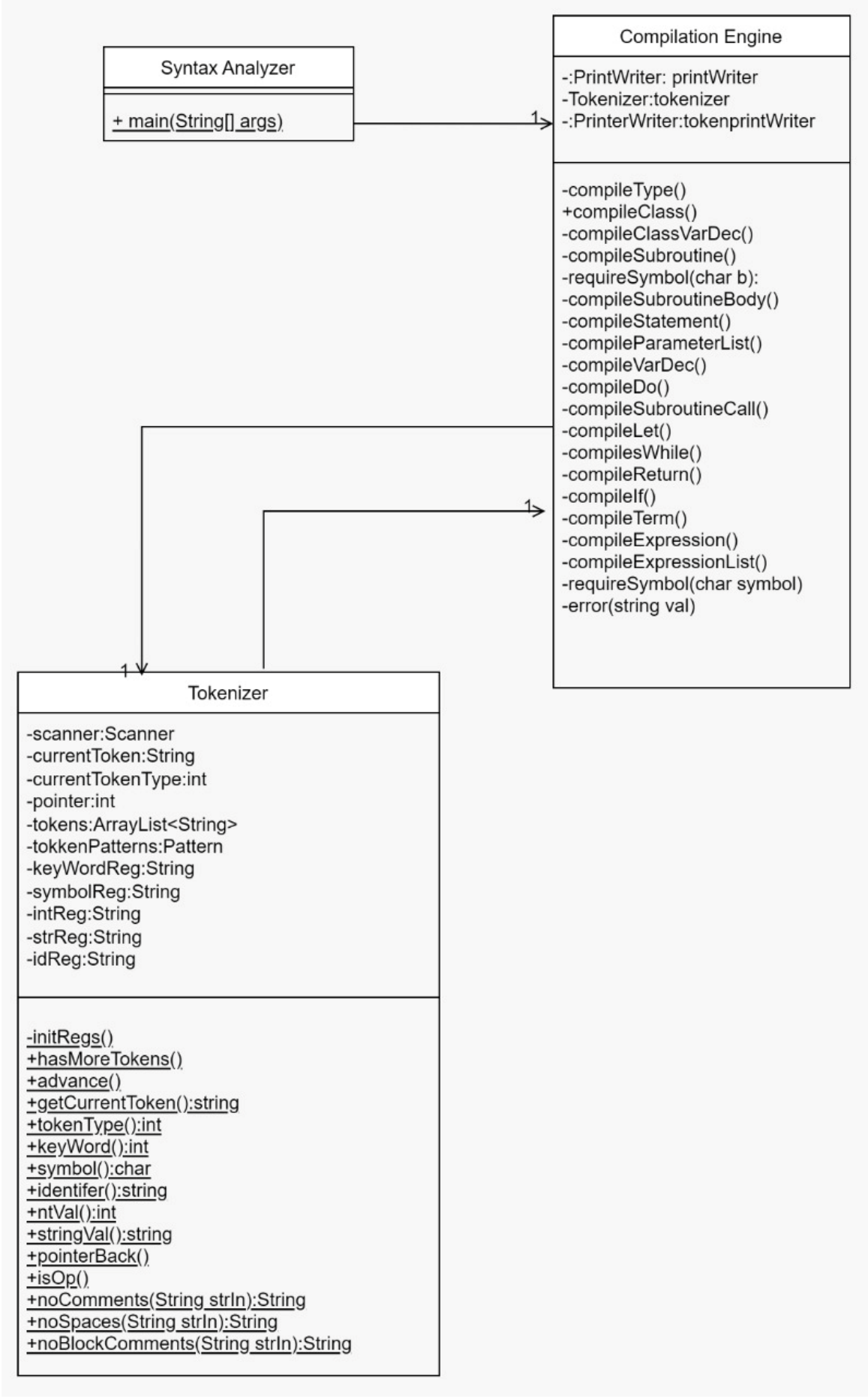
16.error(val):

throws an illegalStateException statement of the missing token

17.requireSymbol():

check if it is a symbol and converts it into tokens else print error symbol name

Q2.


Syntax Analyzer
+ main(String[] args)
1
Compilation Engine
-:PrintWriter: printWriter
-Tokenizer:tokenizer
-:PrinterWriter:tokenprintWriter
-compileType()
+compileClass()
-compileClassVarDec()
-compileSubroutine()
-requireSymbol(char b):
-compileSubroutineBody()
-compileStatement()
-compileParameterList()
-compileVarDec()
-compileDo()
-compileSubroutineCall()
-compileLet()
-compilesWhile()
-compileReturn()
-compileIf()
-compileTerm()
-compileExpression()
-compileExpressionList()
-requireSymbol(char symbol)
-error(string val)
1
1
Tokenizer
-scanner:Scanner
-currentToken:String
-currentTokenType:int
-pointer:int
-tokens:ArrayList<String>
-tokkenPatterns:Pattern
-keyWordReg:String
-symbolReg:String
-intReg:String
-strReg:String
-idReg:String
-initRegs()
+hasMoreTokens()
+advance()
+getCurrentToken():string
+tokenType():int
+keyWord():int
+symbol():char
+identifer():string
+ntVal():int
+stringVal():string
+pointerBack()
+isOp()
+noComments(String strIn):String
+noSpaces(String strIn):String
+noBlockComments(String strIn):String

Q3.what I learned through this project?

I learned about how the high level language works in deeper manner and new function called regex in java .I learned about classes, subroutines, constructors, variables, statements, and expressions. Jack grammar, parse tree creation, and the parsing of various statements such as compile ,WhileStatement, compileTerm, compileLetStatement, compileExpression, and compileExpression. There was also instruction in Jack Grammar and its various lexical elements (keyword, symbol, integerConstant, StringConstant, identifier), Jack Analyzer, and Jack Tokenizer. I learned about working of the syntax analyser of high-level language and how it converts the code into tokens and by using the token creating it into a parse tree. To use the understanding of the working of syntax analyser and implement a working model with the help of 3 classes syntax analyser, tokenizer and compilation engine. I viewed different API models of the classes and how we can use such ideas to make much more complex compiler for programming language like java.

Q4. My Contributions

a. Compileparameterlist() →Which check if there is a parameterlist,if next token is ‘)’,else it go back

b.Compile if()→Compiles if and else statements by checking if the required symbols are there such as ‘(‘,’)’,’{‘,’{‘ . and print the statements <statements> and </statements> .compileStatement inside them and if there is a else statement ,it checks if there are any keyword present inside it and also check if there is ‘{‘ and ‘}’ inside the statements.

c.compileExpression()→it will check if the tokenizer symbol is > or < or & or any other symbols and create a tokens for that symbols and call the compiler term() else it would invoke pointerBack() command and break through the while loop

d.error(string val)→ gives an exception statement when encountered.

e.requireSymbol(char symbol)→checks if the symbol mentioned has same symbol as tokenizer.symbol(),if it is true then it creates a sequences of tokens else it gives an error.

f.compileExpressionList()→checks if there is any expression ,if there is it creates a expression using do and compileExpressoin() else if a symbol is encounter ,it would use the tokenizer.pointerBack()

Q5.

```
    File f=new File(pathname: "C:\\Users\\anugr\\Desktop\\10\\ArrayTest\\Main.jack");
    File ft = new File(pathname: "C:\\Users\\anugr\\Desktop\\10\\ArrayTest\\Main.xml");
    File tp = new File(pathname: "C:\\Users\\anugr\\Desktop\\10\\ArrayTest\\MainT.xml");

    CompilationEngine compilationEngine = new CompilationEngine(f,ft,tp);
    compilationEngine.compileClass();

    System.out.println(x: "File created : " );
}
```