

جامعة أم القرى
UMM AL-QURA UNIVERSITY



Project Scanner

23163109-3 class:1

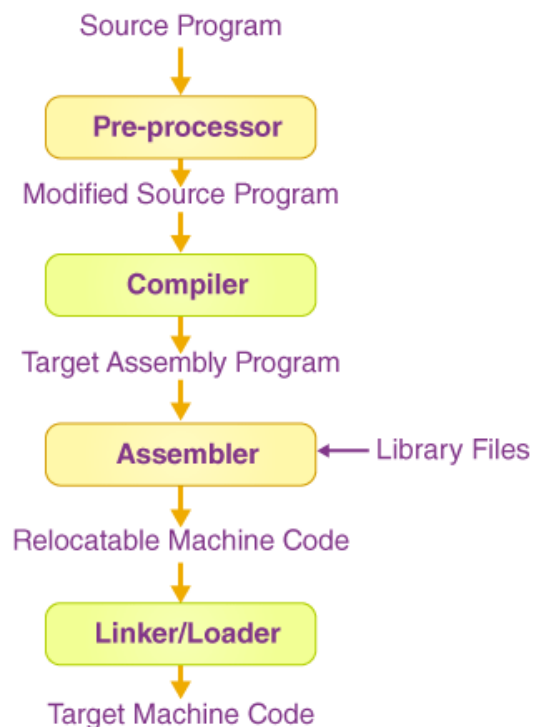
Students:

[Redacted student names]

Introduction:

compiler : A compiler is one component in a tool chain of programs used to create executables from source code. Typically, when you invoke a single command to compile a program, a whole sequence of programs are invoked in the background.

Steps for Language Processing System



Fun fact:

What do you call a compiler that can't find any errors in your code? A wishful compiler.

Programming language:

Key words of our language

If, else, elseif, for, while, int,
char, double, string.

we built lexical analysis which identifies our unique language defined down below:

Operations mathematical:

• / - + =

RE for Number:

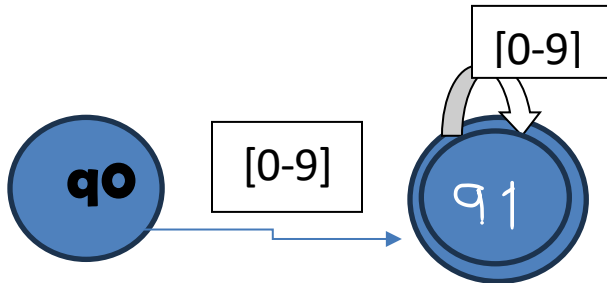
digit = "[#][0-9]+ "

RE for identifier = "[?][a-zA-Z]+ ";

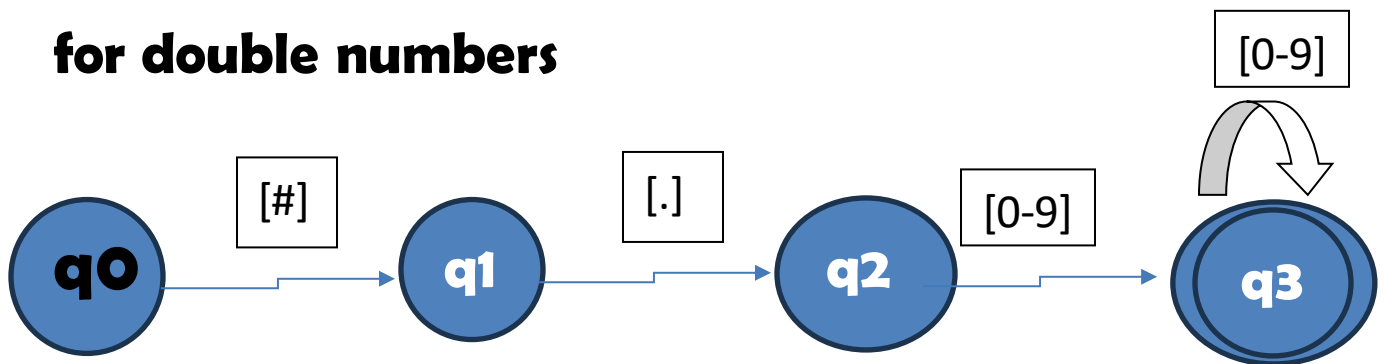
RE for comments = "[\$][a-zA-Z@\$#]+"

DFA In our project

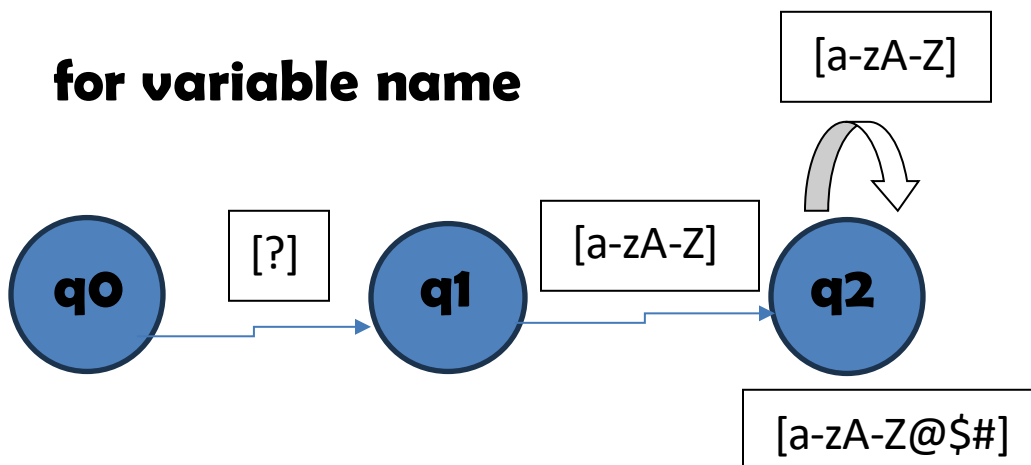
For an integer number:



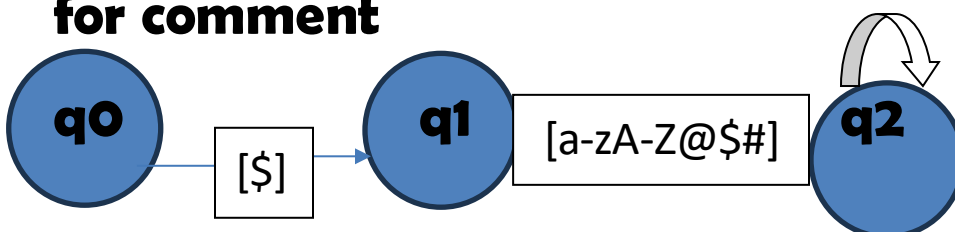
for double numbers



for variable name



for comment



Scanner construction:

We've used java language to form our Scanner

1-it will read the input from text file.

2.Determine RE for letter/digit then it provides tokens for it.

3-if it cant find the matched token for letter/digit it will match it with an Error token.

4.Loop this until of file.

Implementation:

We will run our code using to input files:

First example read file called “test.txt” that will have correct input:

Output:

```
PS C:\Users\b-6r\Downloads\Algorithms_Checkpoint_4\Algorithms_Checkpoint_4\Checkpoint4\src> cd 'c:\Users\b-6r\Downloads\Algorithms_Checkpoint_4\Algorithms_Checkpoint_4\
r\workspaceStorage\88dc71b700acf9434250fb5ce865b70c\redhat.java\jdt_ws\src_74574c8f\bin' 'comp'
<?num,identifier>
<=,operation>
<19,integer>
<,,symbol>
<double,keyword>
<#646.646,double>
<||,Logical Operation>
<if,keyword>
<$buy$#@,comment>
PS C:\Users\b-6r\Downloads\Algorithms_Checkpoint_4\Algorithms_Checkpoint_4\Checkpoint4\src>
```

Second example read file “text.txt” that has incorrect inputs:

Output:

```
PS C:\Users\b-6r\Downloads\Algorithms_Checkpoint_4\Algorithms_Checkpoint_4\Checkpoint4\src> c:: cd 'c:\Users\b-6r\Downloads\Algorithms_Checkpoint_4\Algorithms_Checkpoint_4\Checkpoint4\src'; & 'C:\Program Files\Eclipse Adoptium\jdk-17.0.8-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\b-6r\AppData\Roaming\Code\User\workspaceStorage\08dc71b700acf9434250fb5ce865b70c\redhat.java\jdk_ws\src_74574c8f\bin' 'comp'
<num,error in token !!! >
<=,operation>
<#19,error in token !!! >
<;,Symbol>
<&,Symbol>
<if,keyword>
<9.99,error in token !!! >
<comment$,error in token !!! >
PS C:\Users\b-6r\Downloads\Algorithms_Checkpoint_4\Algorithms_Checkpoint_4\Checkpoint4\src>
```

Our code:

```
import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Pattern;


public class comp {

    // Regular expressions for token categories

    private static String identifierRegularExpression = "[?][a-zA-Z]+";

    private static String integer = "[0-9]+";

    private static String doublenum = "[#][0-9]+.[0-9]+";

    private static String comment = "[$][a-zA-Z@$#]+";

    private static String output = "\\[a-zA-Z]+\\";
```

```

// Lists to hold keywords, symbols, operations, and logical operations

private static List<String> keywords = new ArrayList<>();

private static List<String> symbols = new ArrayList<>();

private static List<String> operations = new ArrayList<>();

private static List<String> logicalOps = new ArrayList<>();


// Additional variables

private static ArrayList<String> fileLines = new ArrayList<>();

private static ArrayList<String> operators = new ArrayList<>();

private static String endOfLine;

private static String endOfWord;

private static String filename = "C:\\Users\\b-6r\\OneDrive\\Desktop\\test.txt";


// DFA states

enum State { Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, QE }


// Input source file and line counter

private static int line = 1;


static {

    // Initialize lists with keywords, symbols, operations, and logical operations

    keywords.add("if");

    keywords.add("else");

    keywords.add("elseif");

    keywords.add("for");

    keywords.add("while");

    keywords.add("int");

    keywords.add("char");

    keywords.add("double");

    keywords.add("string");

```

```

symbols.add("!");

symbols.add("&");

symbols.add(";");


operations.add("+");

operations.add("-");

operations.add("*");

operations.add("/");

operations.add("=");


logicalOps.add("||");

logicalOps.add("&&");
}


public static void main(String[] args) {

    // Code application logic here

    scan();
}


public static void scan() {

    int lineNumber = 1;

    try {

        Path filePath = Paths.get(filename);

        Files.lines(filePath).forEach(fileLines::add);


        for (String line : fileLines) {

            String[] lexemes = getLexemes(line);

            for (String lexeme : lexemes) {

                String tokenCategory = evaluate(lexeme);

```



```

        System.out.println("<" + lexeme + "," + tokenCategory + ">");
    }
    lineNumber++;
}
} catch (IOException e) {
    e.printStackTrace();
    System.out.println("File not found!");
}
}

```

```

public static String[] getLexemes(String line) {
    return line.split(" ");
}

```

```

public static boolean isKeyword(String lex) {
    return keywords.contains(lex);
}

```

```

public static boolean isOperator(String lex) {
    return operations.contains(lex);
}

```

```

public static boolean isLetter(char entry) {
    return (entry >= 'a' && entry <= 'z') || (entry >= 'A' && entry <= 'Z');
}

```

```

public static State executeTransition(State currentState, char entry) {
    switch (currentState) {
        case Q0:
            if (isLetter(entry))

```

```
        return State.Q7;

    else if (entry == '0')

        return State.Q2;

    else if (entry >= '1' && entry <= '9')

        return State.Q1;

    else

        return State.QE;

case Q1:

    if (entry == '.')

        return State.Q5;

    else if (Character.isDigit(entry))

        return State.Q1;

    else

        return State.QE;

case Q2:

    if (entry == '.')

        return State.Q3;

    else

        return State.QE;

case Q3:

    if (Character.isDigit(entry))

        return State.Q4;

    else

        return State.QE;

case Q4:

    if (Character.isDigit(entry))

        return State.Q4;

    else

        return State.QE;

case Q5:
```

```

        if (Character.isDigit(entry))

            return State.Q6;

        else

            return State.QE;

    case Q6:

        if (Character.isDigit(entry))

            return State.Q6;

        else

            return State.QE;

    default:

        return State.QE;

}

}

```

```

public static String evaluate(String str) {

    if (isKeyword(str)) {

        return "keyword";

    } else if (symbols.contains(str)) {

        return "symbol";

    } else if (isOperator(str)) {

        return "operation";

    } else if (logicalOps.contains(str)) {

        return "Logical Operation";

    }

}

```

/*Here's the continuation of the modified code:*/

```

else if (Pattern.matches(identifierRegularExpression, str)) {

    return "identifier";

}

```

```
    } else if (Pattern.matches(comment, str)) {  
        return "comment";  
    } else if (Pattern.matches(integer, str)) {  
        return "integer";  
    } else if (Pattern.matches(doublenum, str)) {  
        return "double";  
    } else if (Pattern.matches(output, str)) {  
        return "Output to the user";  
    } else {  
        return "error in token !!! " ;  
    }  
}  
}  
}
```

References:

- 1.java Documentation for RE.**
- 2.Oracle Java Tutorials- Path and Files.**
- 3.GeeksforGeeks – Introduction to Finite Automata (DFA).**

Distribution of Tasks:

Abdullah	Defined the keywords for the Compiler/Scanner.
Tariq	Wrote the code to ensure that letter/digit that belongs to the string of reserved letter/digit that make up our compiler/scanner
Mohammed	Defined the structure for the keywords will be stored
Mohammed	Wrote the code to be read in the user input from the text file and display the matching result
Abdullah	Wrote RE that are about how to write Identifier, integer, doublenum, comment and text within our Language.
Tariq	Wrote the code to check whether a letter or digit belongs or matches the RE that made up the Language we have defined.

Thank you very much for reviewing our project hopefully we did great.