CD MINI_PROJECT

Topic:

Lexical Analyzer
For
HTML TAGS

Team:

Mukesh -1PI13CS093
Aziaz -1PI13CS089
Niket Raj-1PI13CS100
Sharath.S-1PI13CS140

INTRODUCTION:

Lexical analysis is the process of analyzing a stream of individual characters (normally arranged as lines), into a sequence of **lexical** tokens (tokenization. for instance of "words" and punctuation symbols that make up source code) to feed into the parser.

Our project has 4 classes

1.Lexer.java

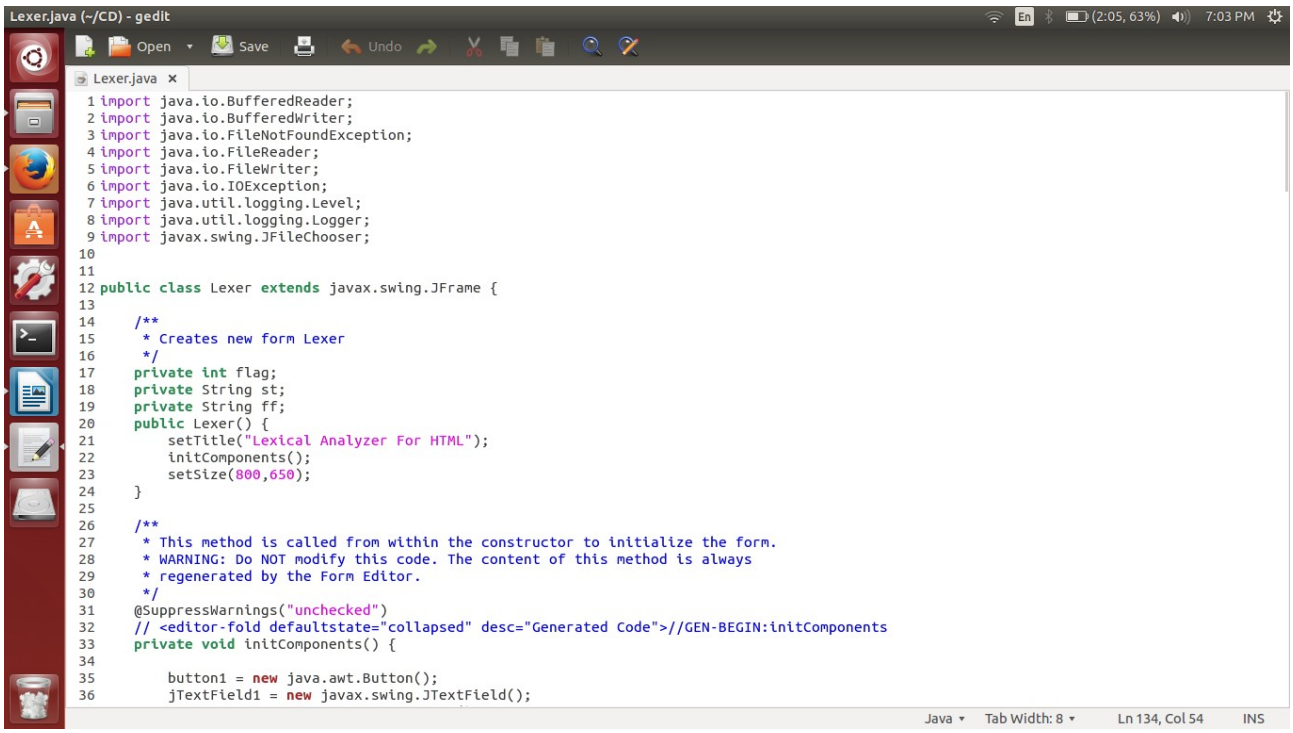
2.Token.java

3.MyTags.java

4.Output.java

Lexer.java:

This program hold the gui for the project and initializes other programs
which are defined above :



Token.java:

This class has constructors to initialize and get tokens

Token.java (~/CD) - gedit

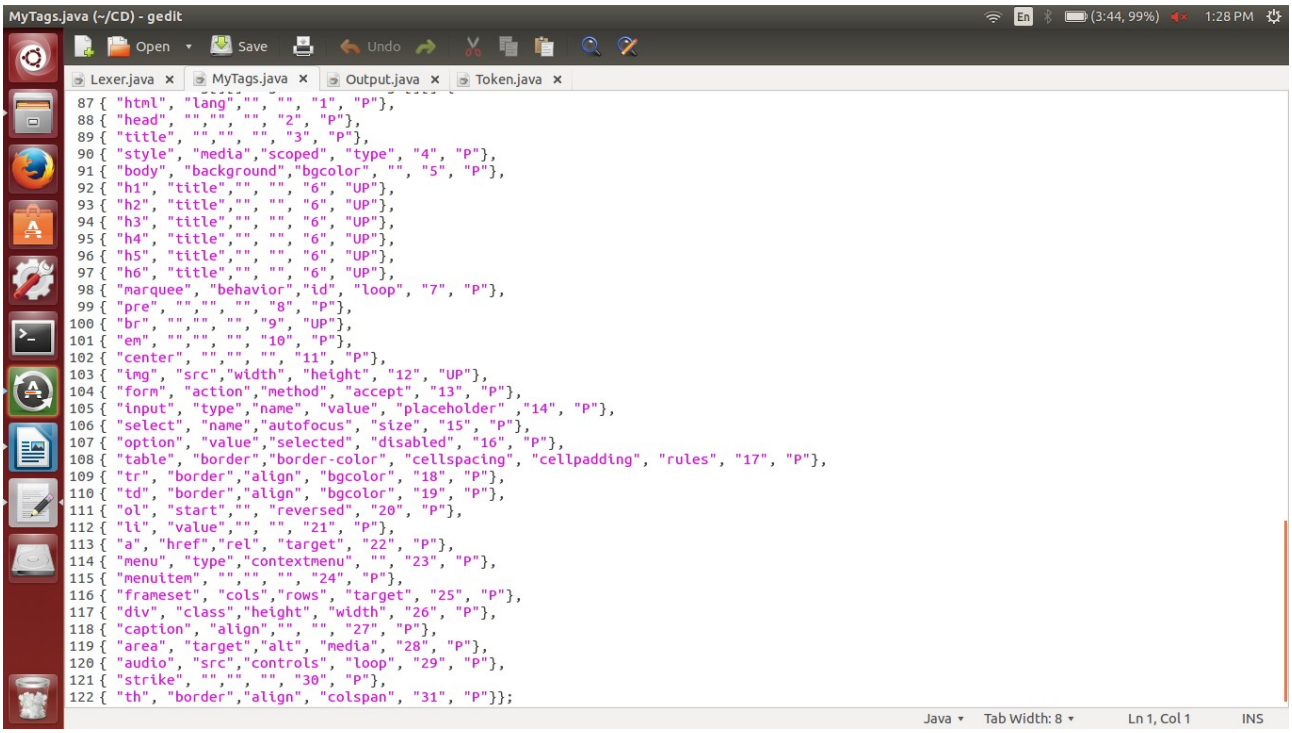
Lexer.java | MyTags.java | Output.java | Token.java

```java
public class Token
{
        String token_type, token_name, tag_type_or_attr_value, parent;

    public Token(String token_type , String token_name , String tag_type_or_attr_value) {
        this.token_type = token_type;
        this.token_name = token_name;
        this.tag_type_or_attr_value= tag_type_or_attr_value;
        this.parent = parent;
    }



    public String getToken_type() {
        return token_type;
    }

    public String getToken_name() {
        return token_name;
    }

    public String getTag_type_or_attr_value() {
        return tag_type_or_attr_value;
    }

    public String getParent() {
        return parent;
    }

}
```

Bracket match found on line: 5 | Java | Tab Width: 8 | Ln 10, Col 6 | INS

MyTags.java:

This class has a two-dimensional String array to store the attributes of for the tags we have defined and the necessary functions.




```
{ "html", "lang","", "", "1", "P"},
{ "head", "","", "", "2", "P"},
{ "title", "","", "", "3", "P"},
{ "style", "media","scoped", "type", "4", "P"},
{ "body", "background","bgcolor", "", "5", "P"},
{ "h1", "title","", "", "6", "UP"},
{ "h2", "title","", "", "6", "UP"},
{ "h3", "title","", "", "6", "UP"},
{ "h4", "title","", "", "6", "UP"},
{ "h5", "title","", "", "6", "UP"},
{ "h6", "title","", "", "6", "UP"},
{ "marquee", "behavior","id", "loop", "7", "P"},
{ "pre", "","", "", "8", "P"},
{ "br", "","", "", "9", "UP"},
{ "em", "","", "", "10", "P"},
{ "center", "","", "", "11", "P"},
{ "img", "src","width", "height", "12", "UP"},
{ "form", "action","method", "accept", "13", "P"},
{ "input", "type","name", "value", "placeholder" ,"14", "P"},
{ "select", "name","autofocus", "size", "15", "P"},
{ "option", "value","selected", "disabled", "16", "P"},
{ "table", "border","border-color", "cellspacing", "cellpadding", "rules", "17", "P"},
{ "tr", "border","align", "bgcolor", "18", "P"},
{ "td", "border","align", "bgcolor", "19", "P"},
{ "ol", "start","", "reversed", "20", "P"},
{ "li", "value","", "", "21", "P"},
{ "a", "href","rel", "target", "22", "P"},
{ "menu", "type","contextmenu", "", "23", "P"},
{ "menuitem", "","", "", "24", "P"},
{ "frameset", "cols","rows", "target", "25", "P"},
{ "div", "class","height", "width", "26", "P"},
{ "caption", "align","", "", "27", "P"},
{ "area", "target","alt", "media", "28", "P"},
{ "audio", "src","controls", "loop", "29", "P"},
{ "strike", "","", "", "30", "P"},
{ "th", "border","align", "colspan", "31", "P"}};
```

Output.java:

This class defines the format in which output should look after the generation of tokens takes place

Output.java (~/CD) - gedit

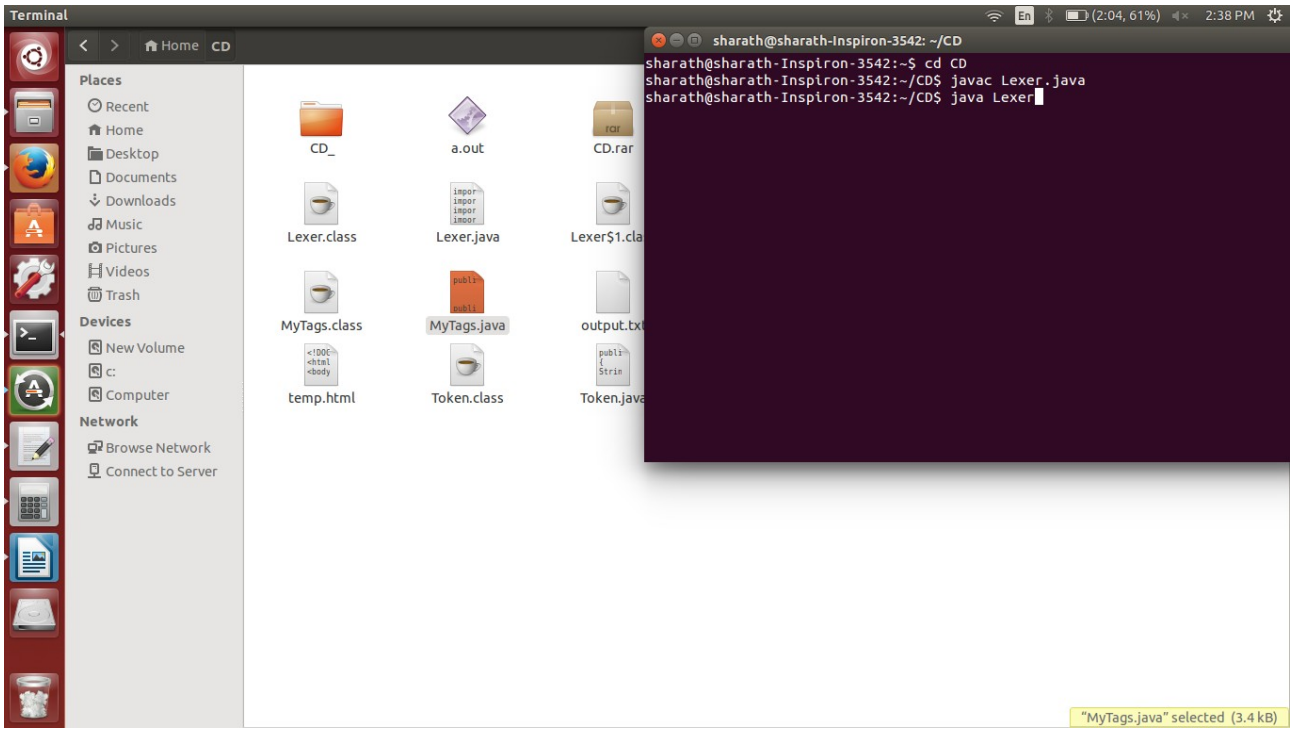
Lexer.java | MyTags.java | Output.java | Token.java

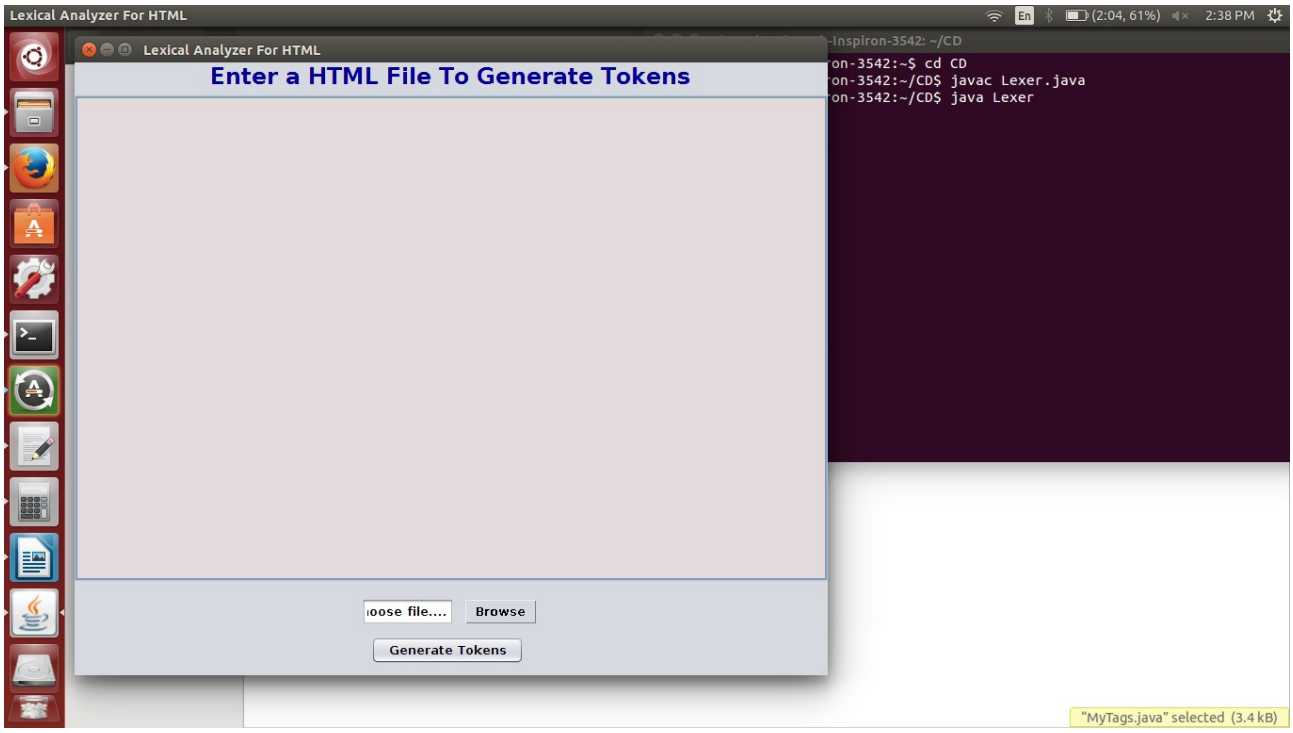
```java
                        tmp = "{TOKEN NO. , TOKEN , NAME , TYPE/VALUE , PARENT TAG ,LINE NO. }" + "\n";
                        tmp += "=====================================================" + "\n" ;
                }
                else {tmp += "-------Empty file---------\n" ;error="None";}
        while((s = in.readLine()) != null) {
            ArrayList<Token> a = getToken(s);
                        line_counter+=1;
                        generateTokens(a,line_counter);
        }
                        bw.write(tmp);
                        bw.write("\nError Messages:-\n");
                        bw.write(error);
                        bw.write("\n^ "+errorCounter+" Errors");
                        bw.close();
        try
            {
                 FileReader reader = new FileReader(file);
                    BufferedReader br = new BufferedReader(reader);
                    jTextArea1.read( br, null );
                    br.close();
                    jTextArea1.requestFocus();

            } catch (FileNotFoundException ex) {
                Logger.getLogger(Lexer.class.getName()).log(Level.SEVERE, null, ex);
            } catch (IOException ex) {
                Logger.getLogger(Lexer.class.getName()).log(Level.SEVERE, null, ex);
            }
    }
        catch(Exception e) {
            e.printStackTrace();
                //System.out.print("-------File Not Found---------");
    }


}
```

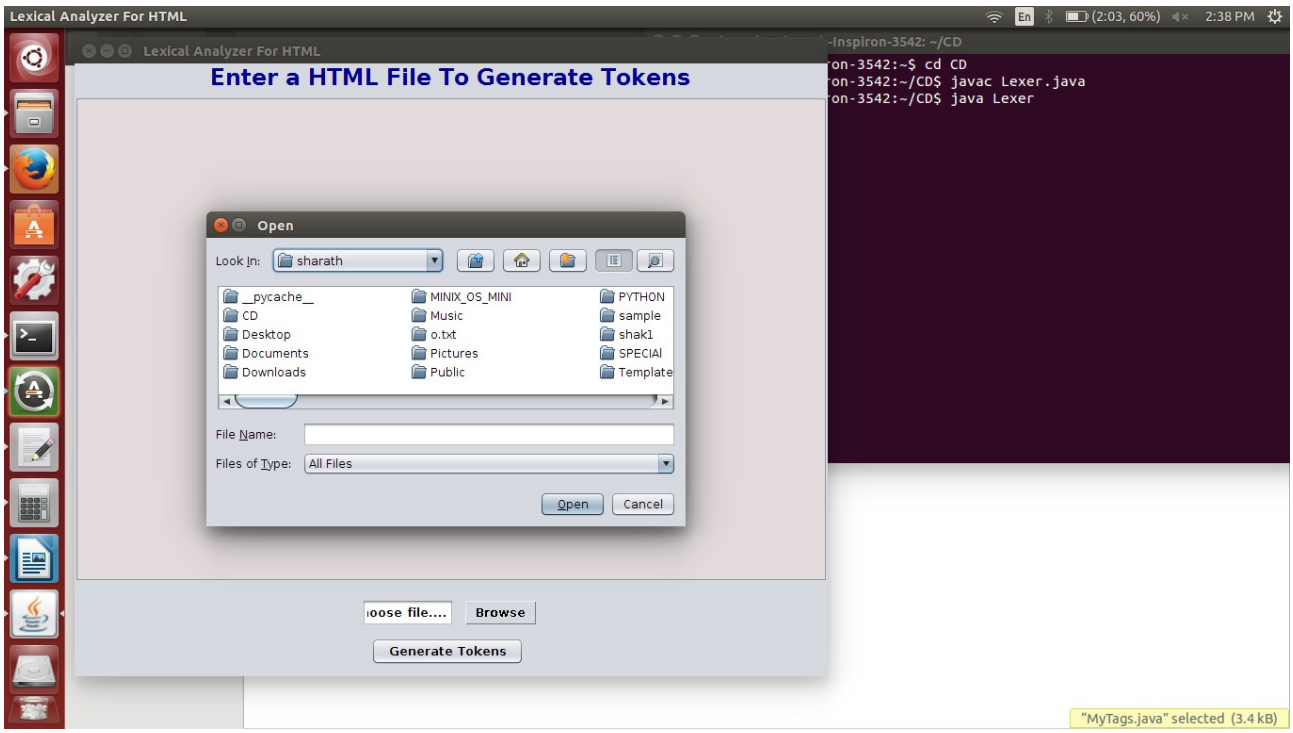
Java | Tab Width: 8 | Ln 19, Col 4 | INS

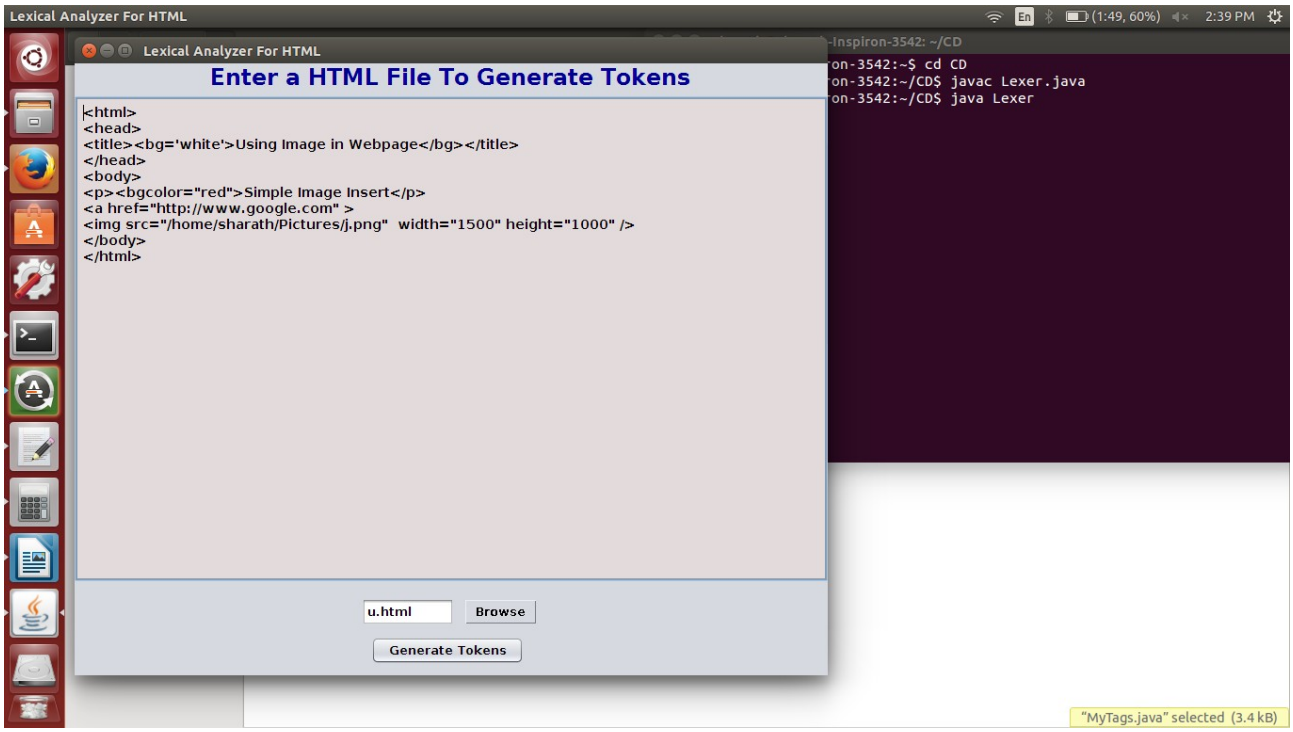
THE OUTPUT

1. Compile the lexer.java program
2. Run the same

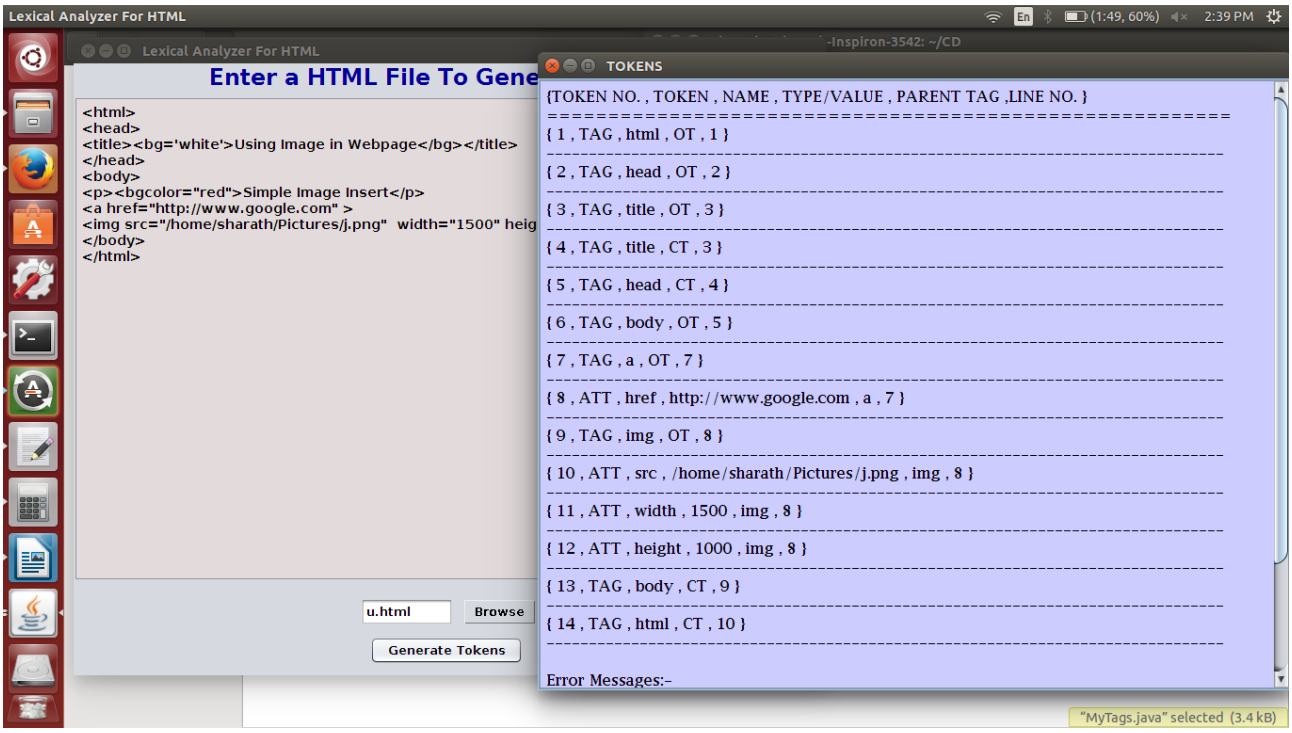


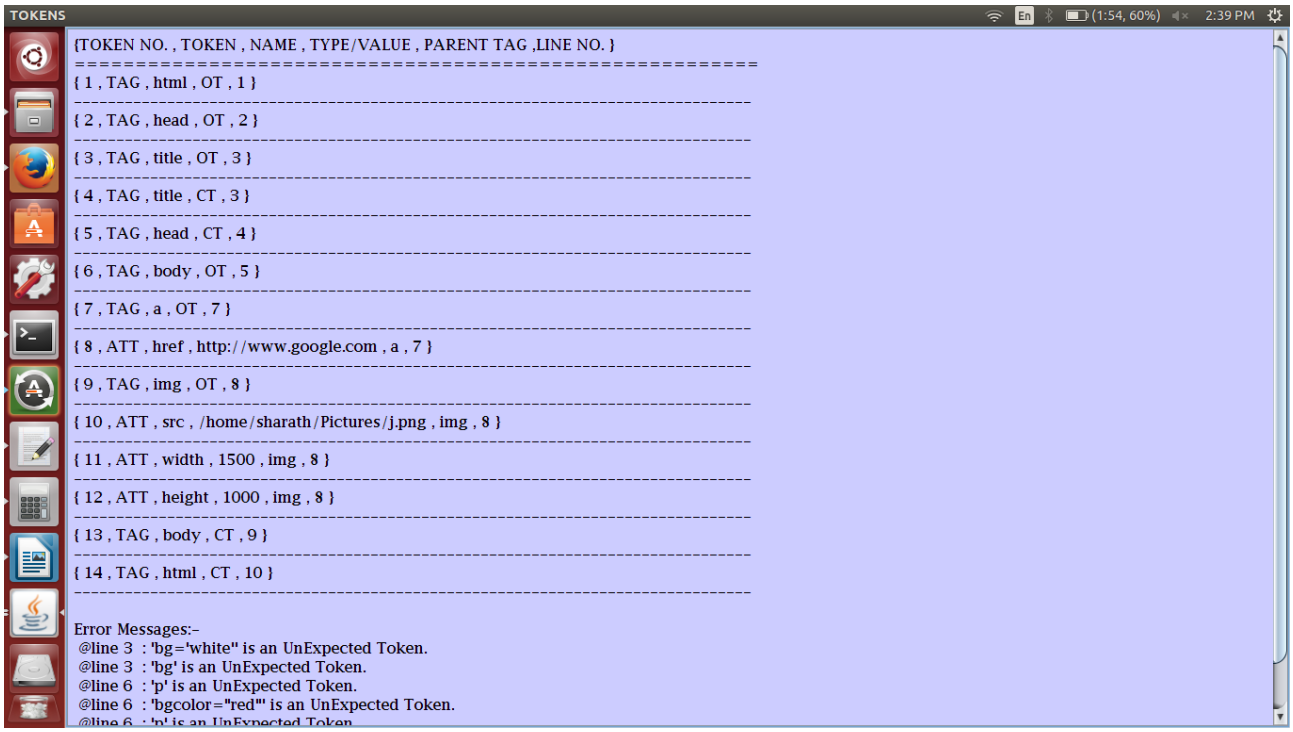



Lexical Analyzer For HTML
Enter a HTML File To Generate Tokens
Open
Look In: sharath
__pycache__
CD
Desktop
Documents
Downloads
MINIX_OS_MINI
Music
o.txt
Pictures
Public
PYTHON
sample
shak1
SPECIAl
Template
File Name:
Files of Type: All Files
Open
Cancel
oose file....
Browse
Generate Tokens
on-3542:~$ cd CD
on-3542:~/CD$ javac Lexer.java
on-3542:~/CD$ java Lexer
"MyTags.java" selected (3.4 kB)


Lexical Analyzer For HTML
Enter a HTML File To Generate Tokens
<html>
<head>
<title><bg='white'>Using Image in Webpage</bg></title>
</head>
<body>
<bgcolor="red">Simple Image Insert
<a href="http://www.google.com" >

</body>
</html>
u.html
Browse
Generate Tokens
on-3542:~$ cd CD
on-3542:~/CD$ javac Lexer.java
on-3542:~/CD$ java Lexer
"MyTags.java" selected (3.4 kB)


Lexical Analyzer For HTML
Lexical Analyzer For HTML
Enter a HTML File To Gene
<html>
<head>
<title><bg='white'>Using Image in Webpage</bg></title>
</head>
<body>
<bgcolor="red">Simple Image Insert
<a href="http://www.google.com" >
<img src="/home/sharath/Pictures/j.png" width="1500" heig
</body>
</html>
u.html
Browse
Generate Tokens
TOKENS
{TOKEN NO. , TOKEN , NAME , TYPE/VALUE , PARENT TAG ,LINE NO. }
{ 1 , TAG , html , OT , 1 }
{ 2 , TAG , head , OT , 2 }
{ 3 , TAG , title , OT , 3 }
{ 4 , TAG , title , CT , 3 }
{ 5 , TAG , head , CT , 4 }
{ 6 , TAG , body , OT , 5 }
{ 7 , TAG , a , OT , 7 }
{ 8 , ATT , href , http://www.google.com , a , 7 }
{ 9 , TAG , img , OT , 8 }
{ 10 , ATT , src , /home/sharath/Pictures/j.png , img , 8 }
{ 11 , ATT , width , 1500 , img , 8 }
{ 12 , ATT , height , 1000 , img , 8 }
{ 13 , TAG , body , CT , 9 }
{ 14 , TAG , html , CT , 10 }
Error Messages:-
"MyTags.java" selected (3.4 kB)


TOKENS
{TOKEN NO. , TOKEN , NAME , TYPE/VALUE , PARENT TAG ,LINE NO. }
{ 1 , TAG , html , OT , 1 }
{ 2 , TAG , head , OT , 2 }
{ 3 , TAG , title , OT , 3 }
{ 4 , TAG , title , CT , 3 }
{ 5 , TAG , head , CT , 4 }
{ 6 , TAG , body , OT , 5 }
{ 7 , TAG , a , OT , 7 }
{ 8 , ATT , href , http://www.google.com , a , 7 }
{ 9 , TAG , img , OT , 8 }
{ 10 , ATT , src , /home/sharath/Pictures/j.png , img , 8 }
{ 11 , ATT , width , 1500 , img , 8 }
{ 12 , ATT , height , 1000 , img , 8 }
{ 13 , TAG , body , CT , 9 }
{ 14 , TAG , html , CT , 10 }
Error Messages:-
@line 3 : 'bg='white" is an UnExpected Token.
@line 3 : 'bg' is an UnExpected Token.
@line 6 : 'p' is an UnExpected Token.
@line 6 : 'bgcolor="red"' is an UnExpected Token.

ERROR RECOVERY: BY

Panic Mode Recovery

Panic mode recovery is an error recovery method that can be used in any kind of parsing, because error recovery depends somewhat on the type of parsing technique used. In panic mode recovery, a parser discards input symbols until a statement delimiter , such as a semicolon or an end, is encountered . The parser then deletes stack entries until it finds an entry that will allow it to continue parsing, given the synchronizing token on the input. This method is simple to implement, and it never gets into an infinite loop.