**/\* CS 6413 : Compiler Design & Construction Spring 2015 Instructed By : Dr. R. M. Siegfried**

**TERM PROJECT : PART 1 :**

**GROUP MEMBERS :- N #**

**(1) SUMITRA KANNAN N 12952143**

**(2) MRUNAL PATEKAR N 14789187**

**(3) VINAY N 10423043**

**AIM OF THE PROJECT :** Write a scanner that recognizes a function gettoken() that recognizes the lexemes of the language and returns a token corresponding to that lexeme. We will Generate Tokens for given program.

**ASSUMPTIONS :** Since we do not yet have a symbol table, we use the following temporary tokens :

(i) All words (a letter followed by zero or more alphanumeric characters) have the temporary token : tokword,

(ii) All numbers have the temporary token : toknumber and

(iii) All others have the temporary token : tokop.

**CODE :**

**/\* Inclusion of import java libraries, Main Scanner Class, assigning temporary tokens : toknumber, tokword and tokop to input strings, getToken() : By :- SUMITRA KANNAN :- N12952143 ,MRUNAL PATEKAR :- N 14789187 \*/**

import java.io.\*;

import java.util.\*;

import java.util.regex.\*;

class Scanner {

public static Vector Num\_token= new Vector(5,2);

public static Enumeration Num\_Enum = Num\_token.elements();

public static Vector Word\_token= new Vector(5,2);

public static Enumeration Word\_Enum = Word\_token.elements();

public static Vector Op\_token= new Vector(5,2);

public static Enumeration Op\_Enum = Op\_token.elements();

public static String NUM\_TOK = "toknumber";

public static String WORD\_TOK = "tokword";

public static String OP\_TOK = "tokop";

public void getToken(String tokens) //this function breaks string in tokens

{

StringTokenizer newtokens = new StringTokenizer(tokens);

while (newtokens.hasMoreTokens()) //checks if another token is available

{

String token = newtokens.nextToken() ;

String tokenType = getTokenType(token) ;

System.out.println(token + "\t\t" + tokenType); //the type of token is returned

}

}

**/\* Inclusion of getTokenType(), pattern matching of lexems with temporary tokens assigned : By :- MRUNAL PATEKAR :- N14789187 , VINAY :- N10423043 \*/**

private String getTokenType(String token) {

if(token != null) {

if (Pattern.matches("[\\d]+", token)) { //checks if it is a number token

return NUM\_TOK;

} else if (Pattern.matches("[\\w]+", token)) { //checks if it is a word token

return WORD\_TOK;

} else {

return OP\_TOK; // checks value operator token

}

}

return null; // if none of the above found , than null will be returned

}

**/\* Elaboration of Main class, getting input from user, pattern matching with input lexemes : By :- MRUNAL PATEKAR :- N 14789187 , VINAY :- N 10423043 \*/**

public static void main(String s[]) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); //create Buffered reader using system.in to read

int i, strlen;

Vector vtok = new Vector(5, 2);

Enumeration vEnum = vtok.elements();

String getinput,getinput1; // to get input string from user and store it

Scanner sc = new Scanner();

System.out.println("Input your string with appropriate spaces and write OVER after the code to get appropriate tokens: ");

do {

getinput = br.readLine();

if (getinput.endsWith("over"))

{

strlen = getinput.length();

strlen -= 4;

getinput = getinput.substring(0,strlen);

getinput1 = ("over"); // whenever END is typed in the end, inputs wont be taken

vtok.addElement(new String(getinput));

vtok.addElement(new String(getinput1));

break;

}

else vtok.addElement(new String(getinput));

}

while (!getinput.equalsIgnoreCase("over"));

getinput = "";

while (!getinput.equalsIgnoreCase("over"))

{

while (vEnum.hasMoreElements()) {

getinput = (String) vEnum.nextElement();

if (getinput.endsWith("over"))

{

strlen = getinput.length();

strlen -= 4;

getinput1 = getinput.substring(0,strlen);

sc.getTokens(getinput1);

break;

}

else

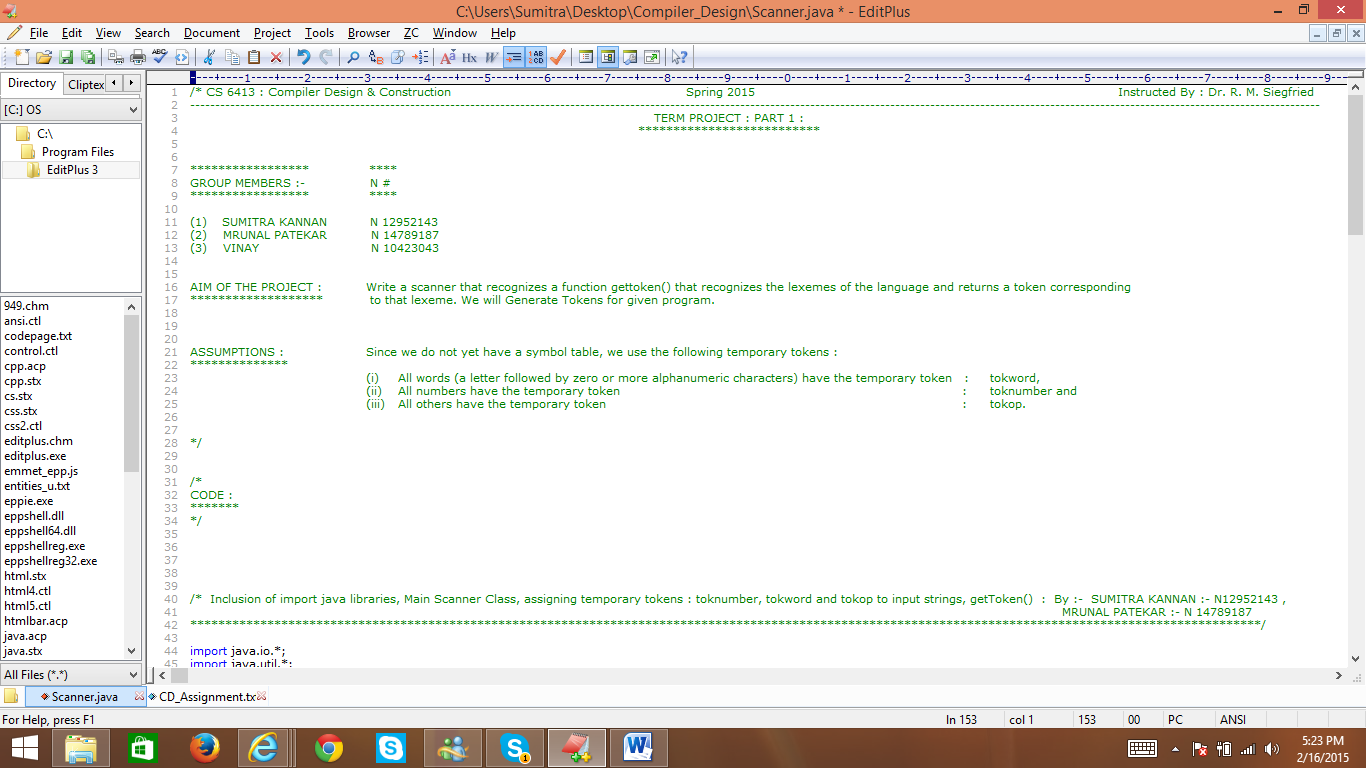
sc.getToken(getinput); // call to function getToken() to identify tokens

}

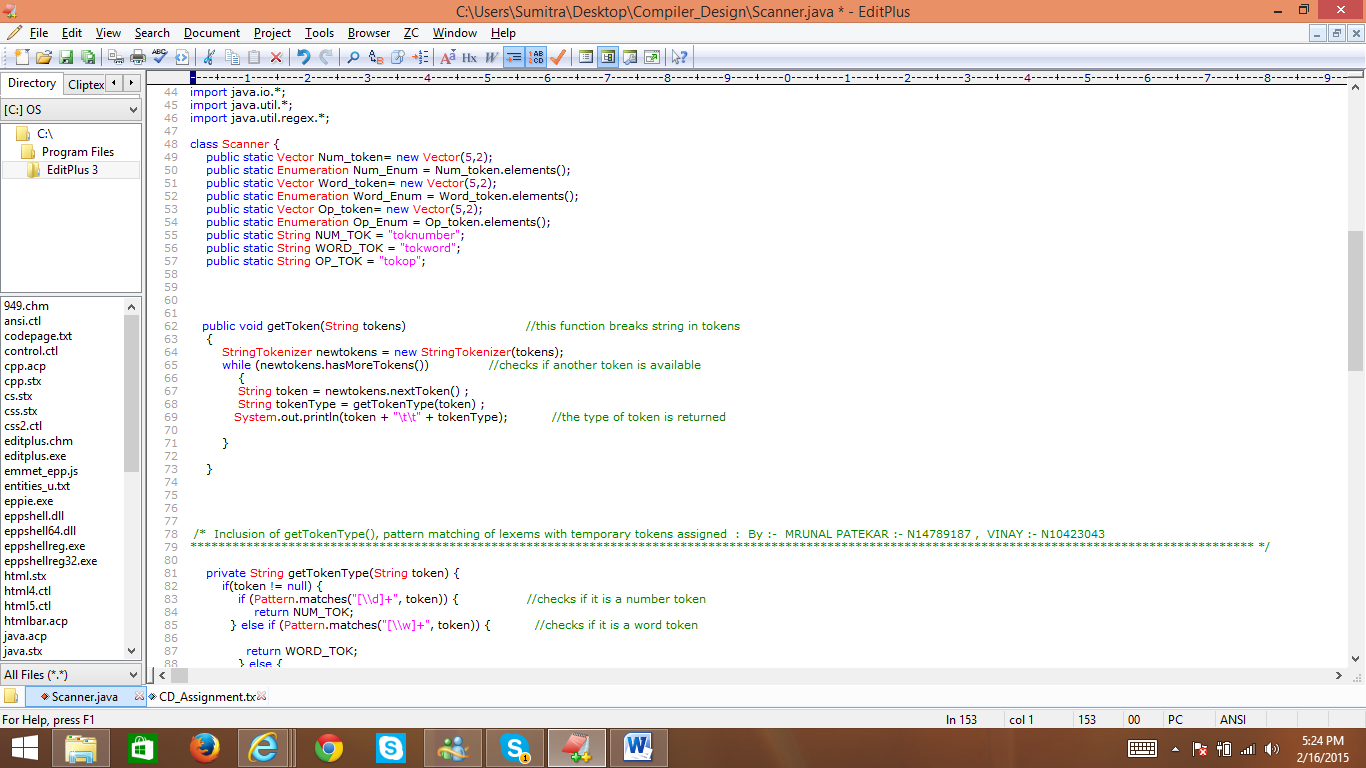
}

}

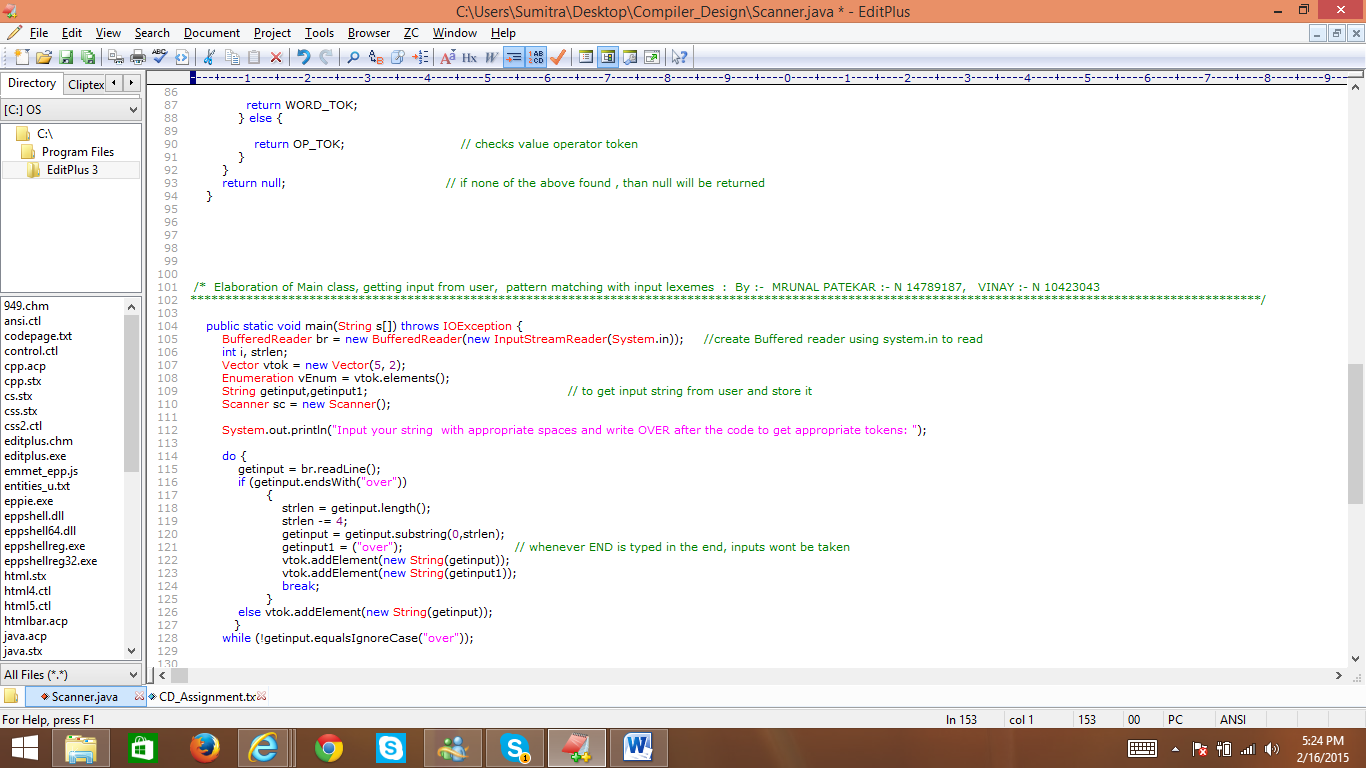
}



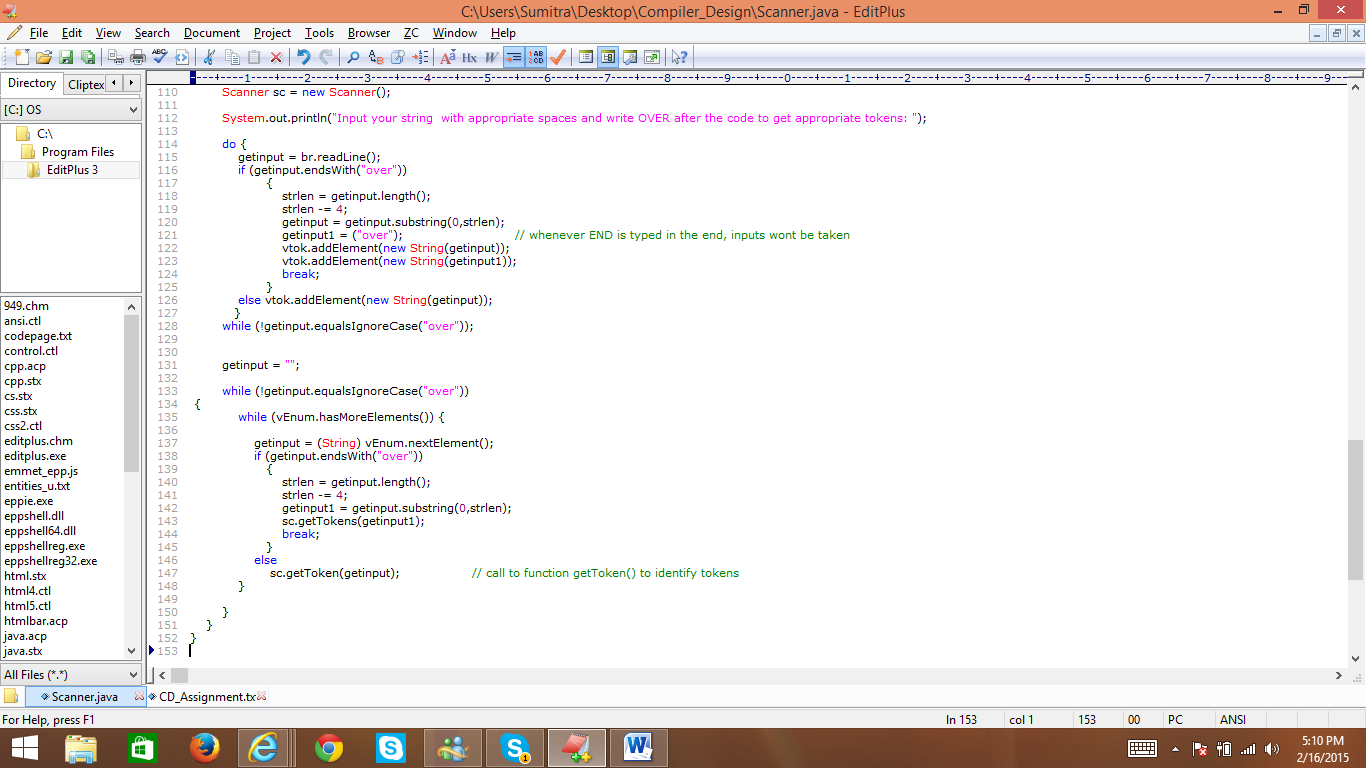
**Fig 1 : Screenshot of Scanner Class Code**



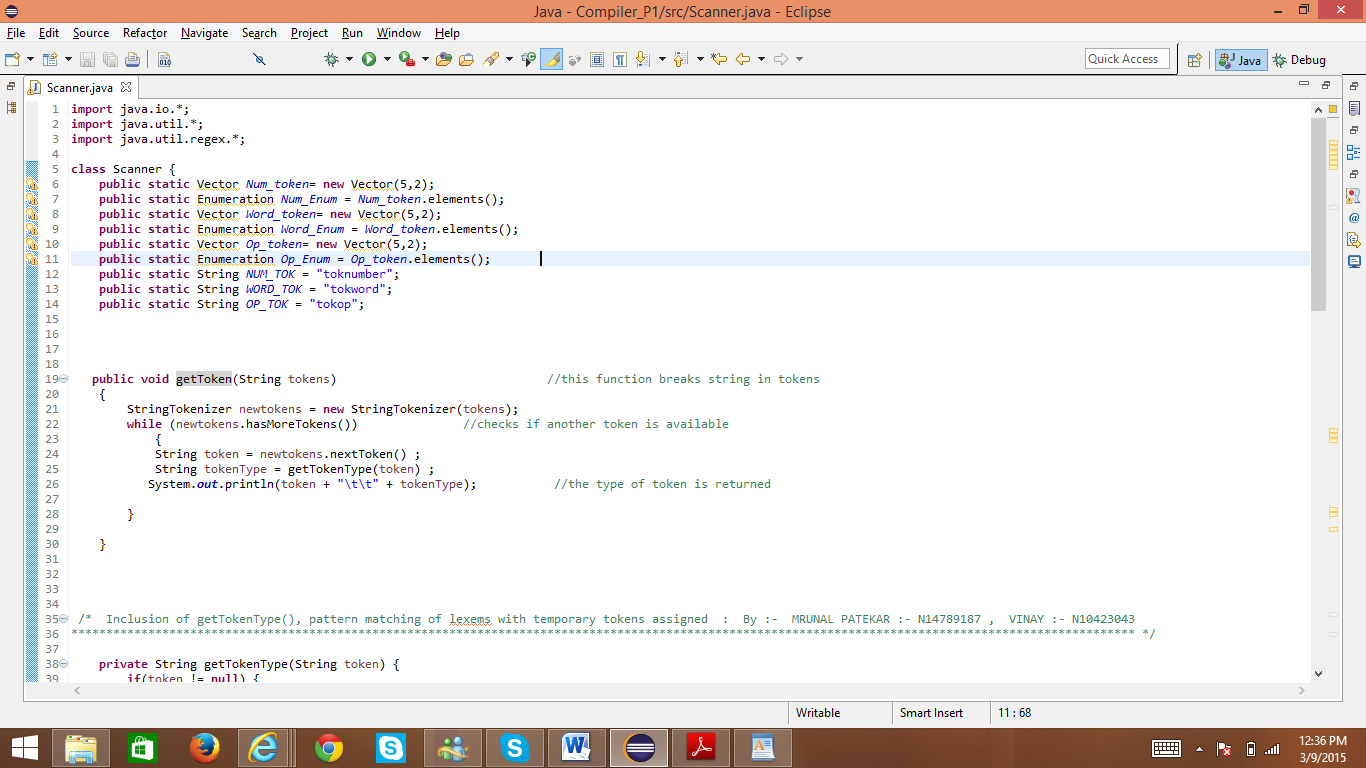
**Fig 2 : Screenshot of Scanner Class Code**



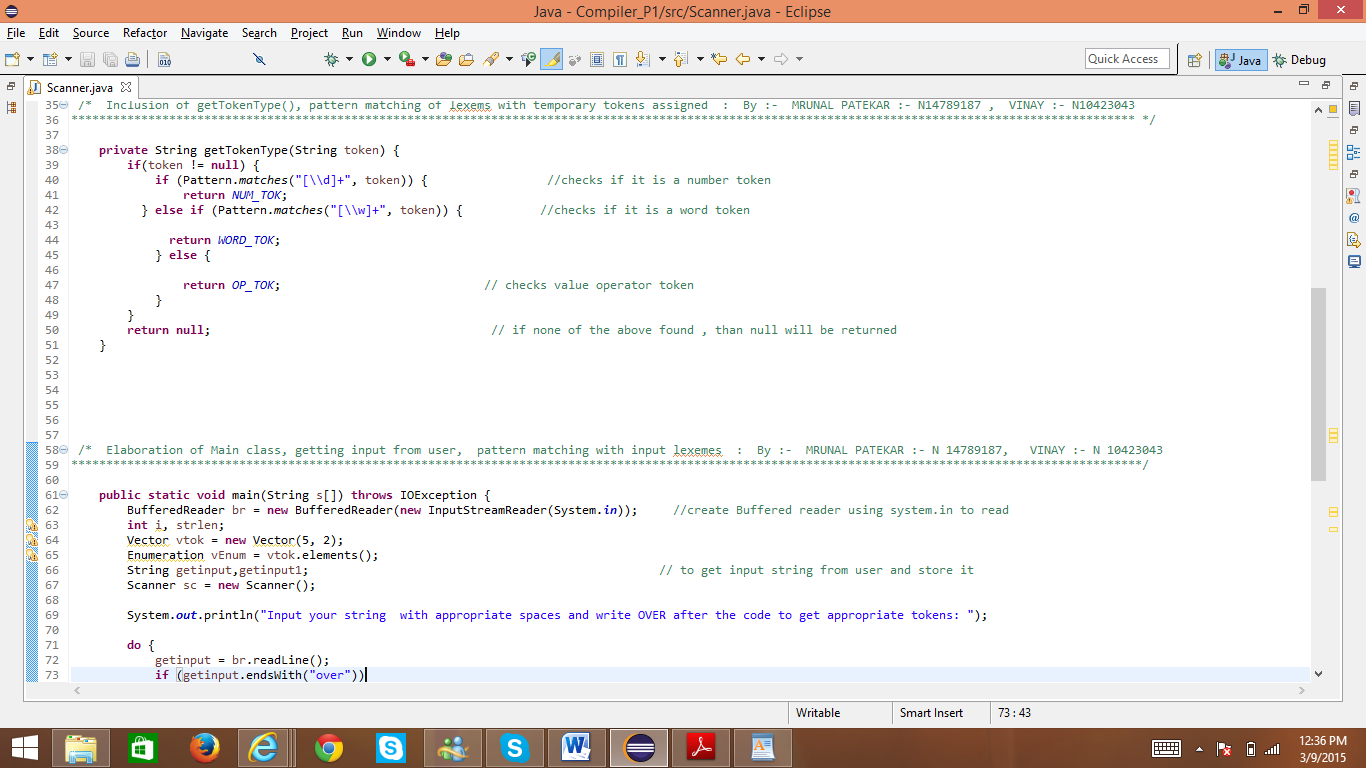
**Fig 3 : Screenshot of Scanner Class Code**



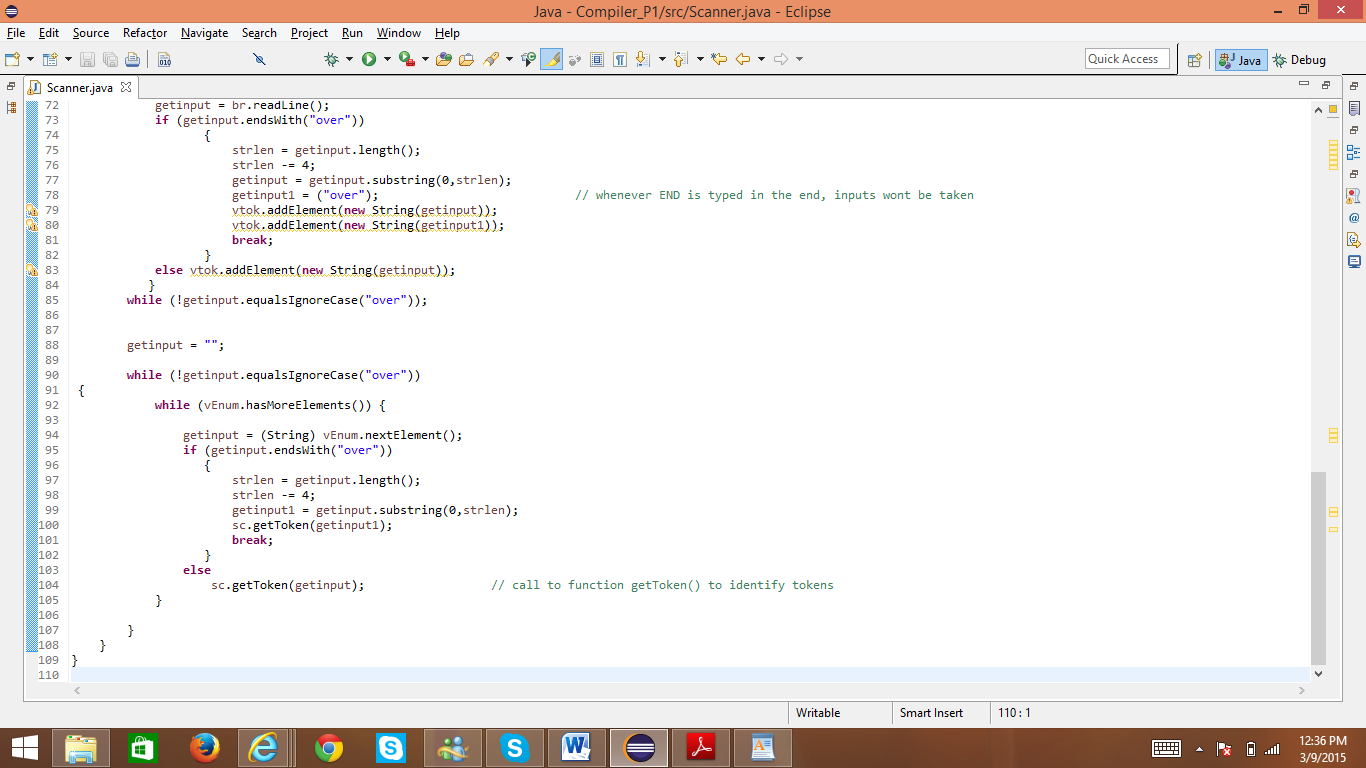
**Fig 4 : Screenshot of Scanner Class Code**



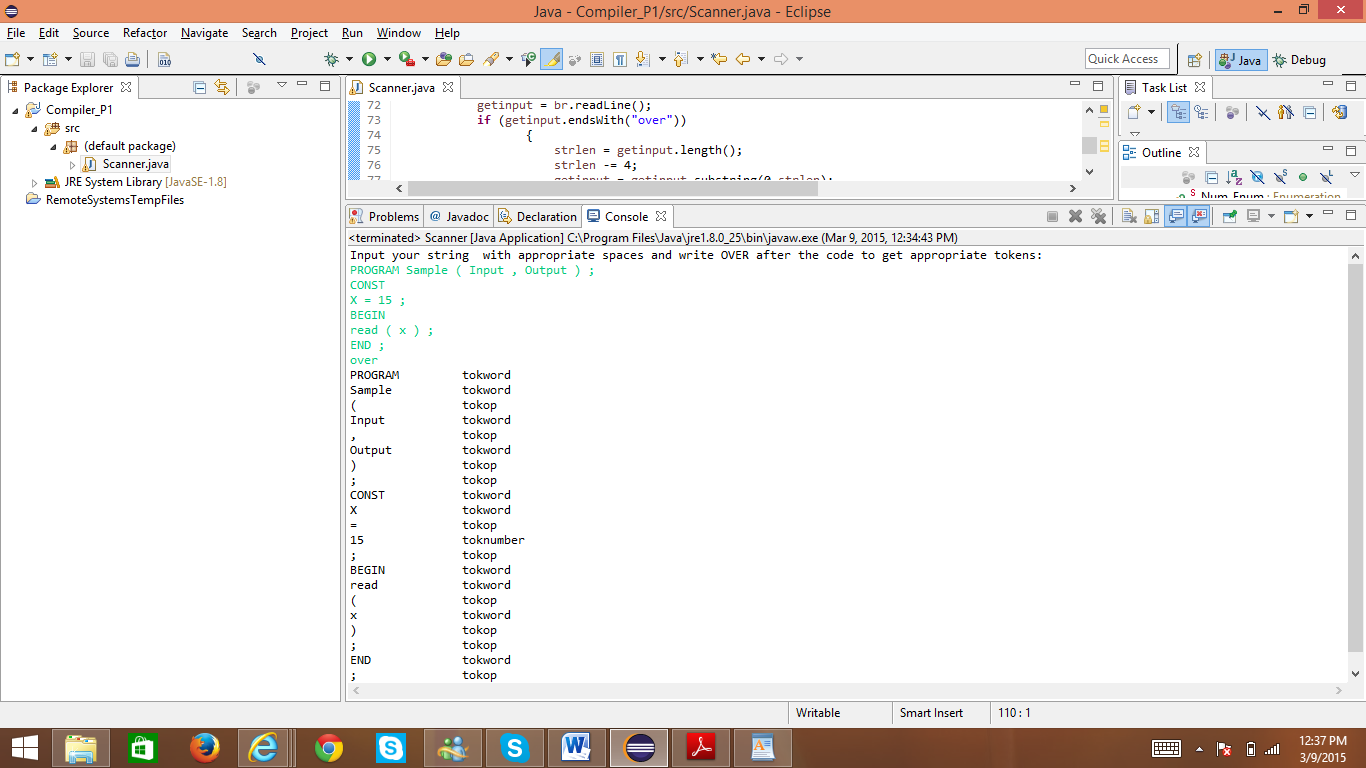
**Fig 5 : Screenshot of Scanner Class Code Execution in Eclipse Java Compiler**



**Fig 6 : Screenshot of Scanner Class Code Execution in Eclipse Java Compiler**



**Fig 7 : Screenshot of Scanner Class Code Execution in Eclipse Java Compiler**



**Fig 8 : Screenshot of Output of Scanner Class Code Execution in Eclipse Java Compiler**

**Input for the scanner :**

PROGRAM Sample ( Input , Output ) ;

CONST

X = 15 ;

BEGIN

read ( x ) ;

END ;

**Output for the scanner :**

Start typing your code:

When you finish writing enter STOP:

PROGRAM Sample ( Input , Output ) ;

CONST

X = 15 ;

BEGIN

read ( x ) ;

END ;

stop

PROGRAM tokWord

Sample tokWord

( tokOp

Input tokWord

, tokOp

Output tokWord

) tokOp

; tokOp

CONST tokWord

X tokWord

= tokOp

15 tokNumber

; tokOp

BEGIN tokWord

read tokWord

( tokOp

x tokWord

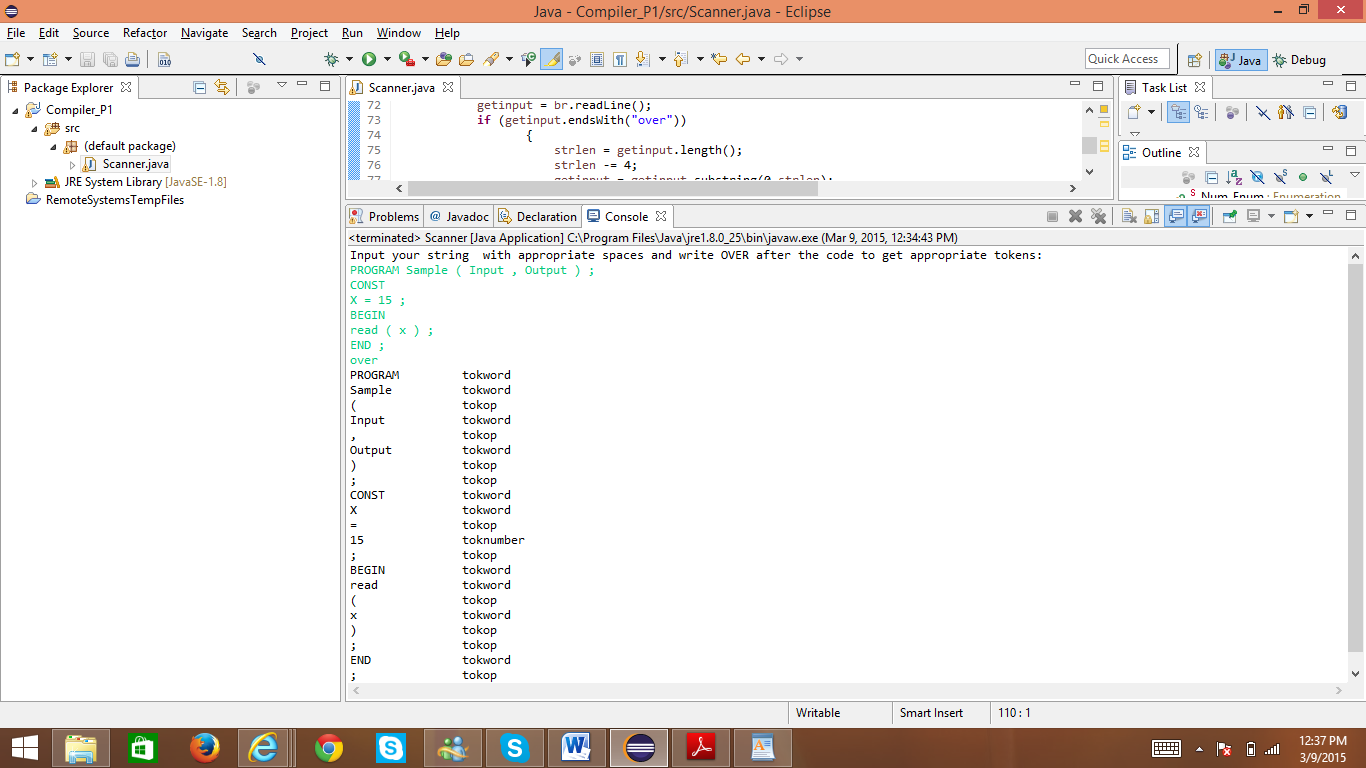
) tokOp

; tokOp

END tokWord

; tokOp

Over



**Fig 9: Screenshot of Input & Output of Scanner Class in Eclipse Java Compiler**