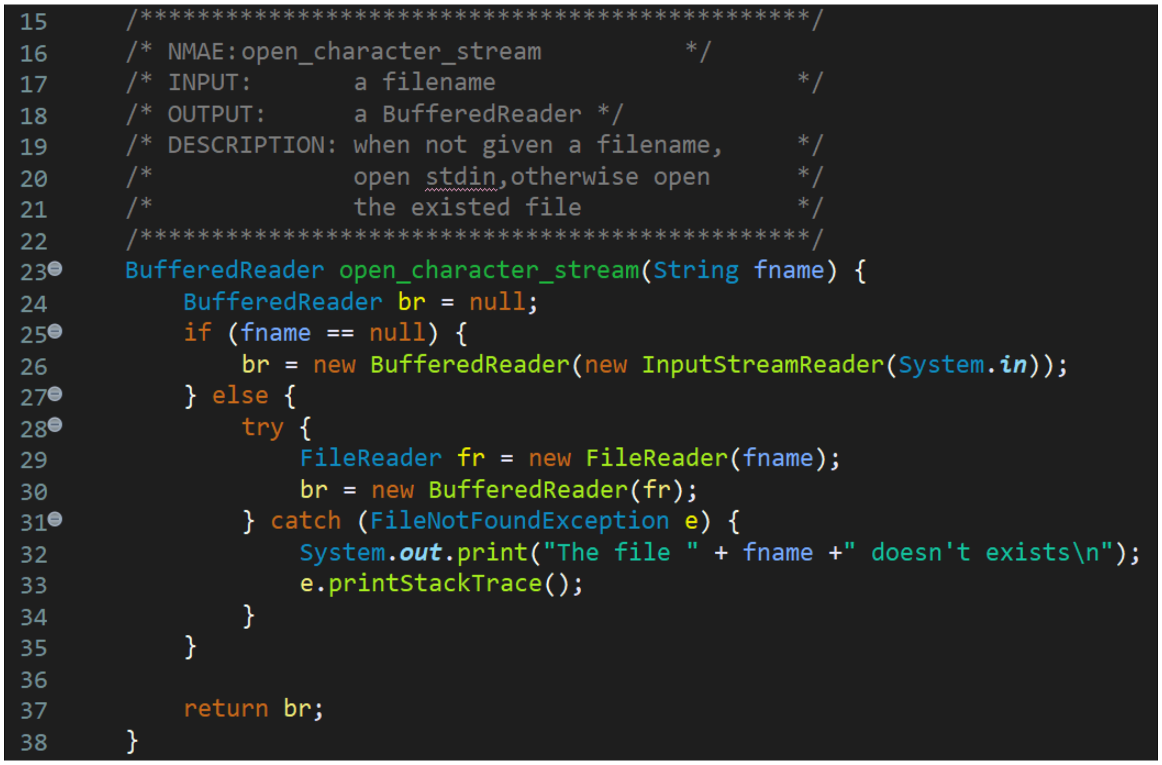
Testing-team-04
Mason Berry
Adalynn Everly
CFG.pdf

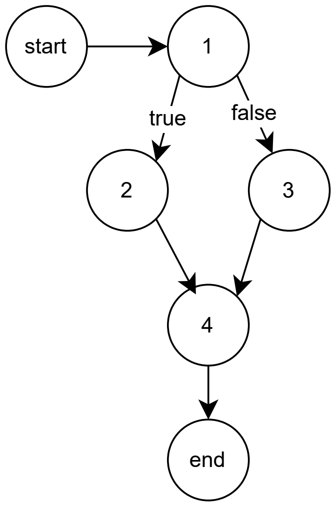
page number	Funciton Name	Line Numbers
1	open_character_stream	23
2	get_char	45
3	unget_char	62
4	open_token_stream	78
5, 6, 7	get_token	95
8	is_token_end	173
9	token_type	204
10, 11	print_token	220
12	is_comment	264
13	is_keyword	277
14	is_char_constant	291
15	is_num_constant	304
16	is_str_constant	328
17	is_identifier	350
18	print_spec_symbol	377
19	is_spec_symbol	426
20	main	459

- End nodes may drawn multiple times to improve graph readability.
- All end nodes represent the same funciton exit node.
- Code screenshots are from before faults were corrected.
- Drawn and exported from draw.io

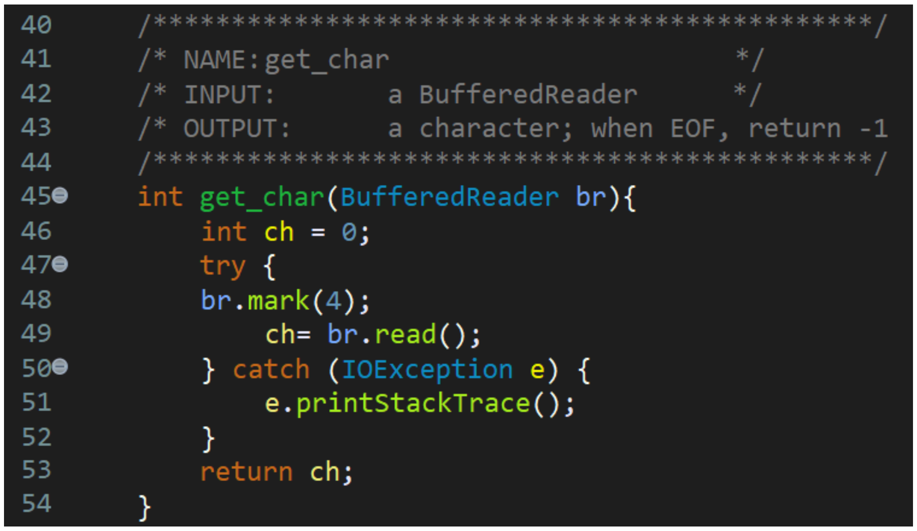


```java
15      /******************************************/
16      /* NMAE:open_character_stream          */
17      /* INPUT:       a filename                */
18      /* OUTPUT:      a BufferedReader */
19      /* DESCRIPTION: when not given a filename,     */
20      /*              open stdin,otherwise open      */
21      /*              the existed file               */
22      /******************************************/
23      BufferedReader open_character_stream(String fname) {
24          BufferedReader br = null;
25          if (fname == null) {
26              br = new BufferedReader(new InputStreamReader(System.in));
27          } else {
28              try {
29                  FileReader fr = new FileReader(fname);
30                  br = new BufferedReader(fr);
31              } catch (FileNotFoundException e) {
32                  System.out.print("The file " + fname +" doesn't exists\n");
33                  e.printStackTrace();
34              }
35          }
36
37          return br;
38      }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	24,25	24	25	
2	26	26	26	
3	29, 30	29	30	
4	37	37	37	



1

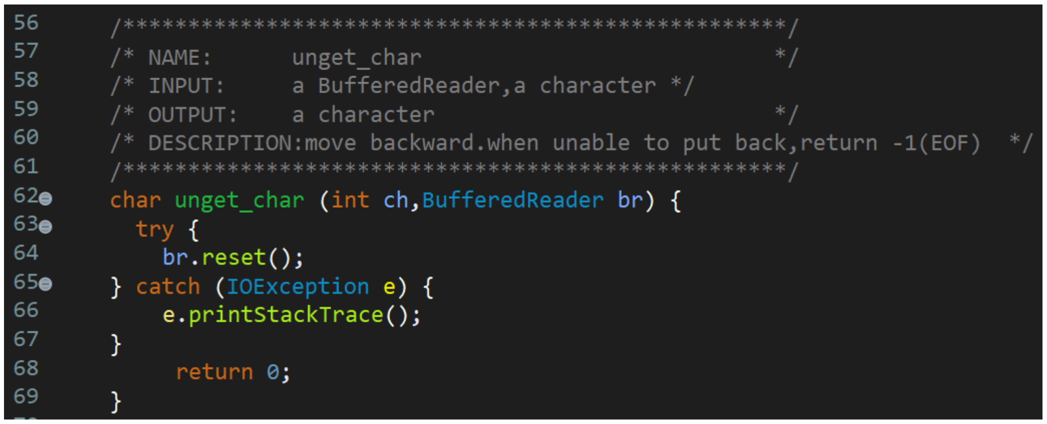


```
40      /*******************************************/
41      /* NAME:get_char                          */
42      /* INPUT:        a BufferedReader         */
43      /* OUTPUT:       a character; when EOF, return -1
44      /*******************************************/
45      int get_char(BufferedReader br){
46          int ch = 0;
47          try {
48          br.mark(4);
49              ch= br.read();
50          } catch (IOException e) {
51              e.printStackTrace();
52          }
53          return ch;
54      }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	46, 48, 49, 53	46	53	

$$\begin{tikzcd}
\text{start} \arrow[r] & 1 \arrow[d] \\
 & \text{end}
\end{tikzcd}$$

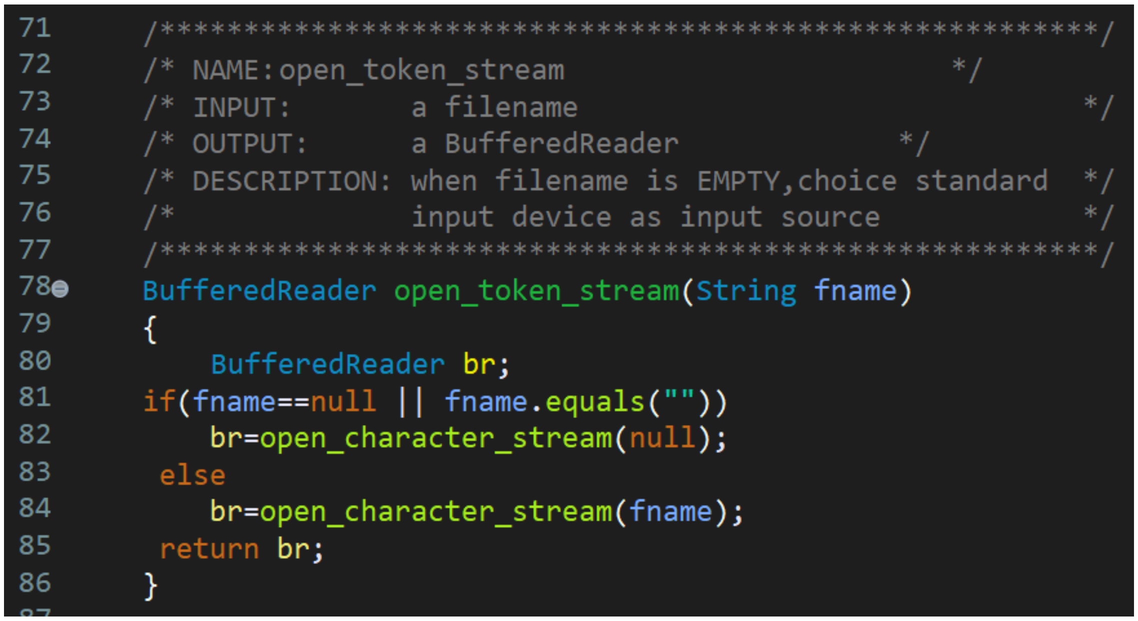
2



```
56      /***************************************************/
57      /* NAME:      unget_char                          */
58      /* INPUT:     a BufferedReader,a character */
59      /* OUTPUT:    a character                         */
60      /* DESCRIPTION:move backward.when unable to put back,return -1(EOF)  */
61      /***************************************************/
62      char unget_char (int ch,BufferedReader br) {
63        try {
64          br.reset();
65      } catch (IOException e) {
66          e.printStackTrace();
67      }
68           return 0;
69      }
```

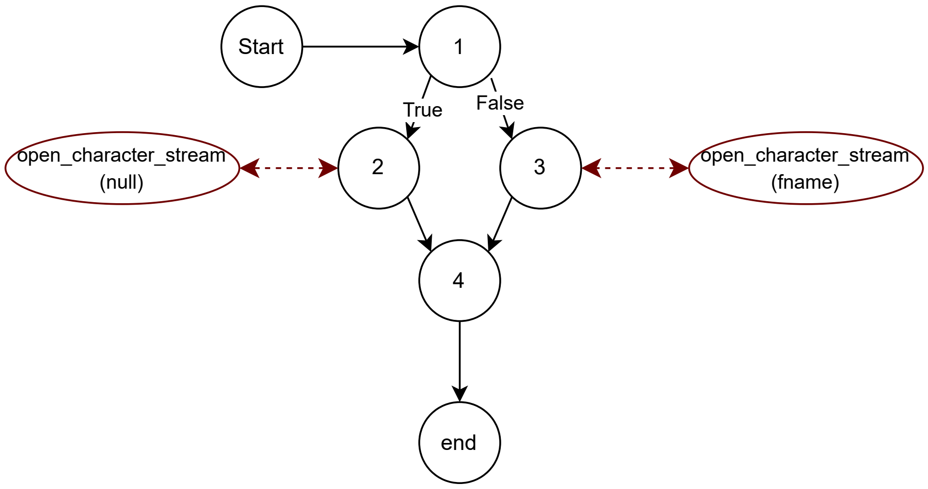
Block Number	Line numbers	Entry	Exit	Function Calls
1	64, 68	64	68	

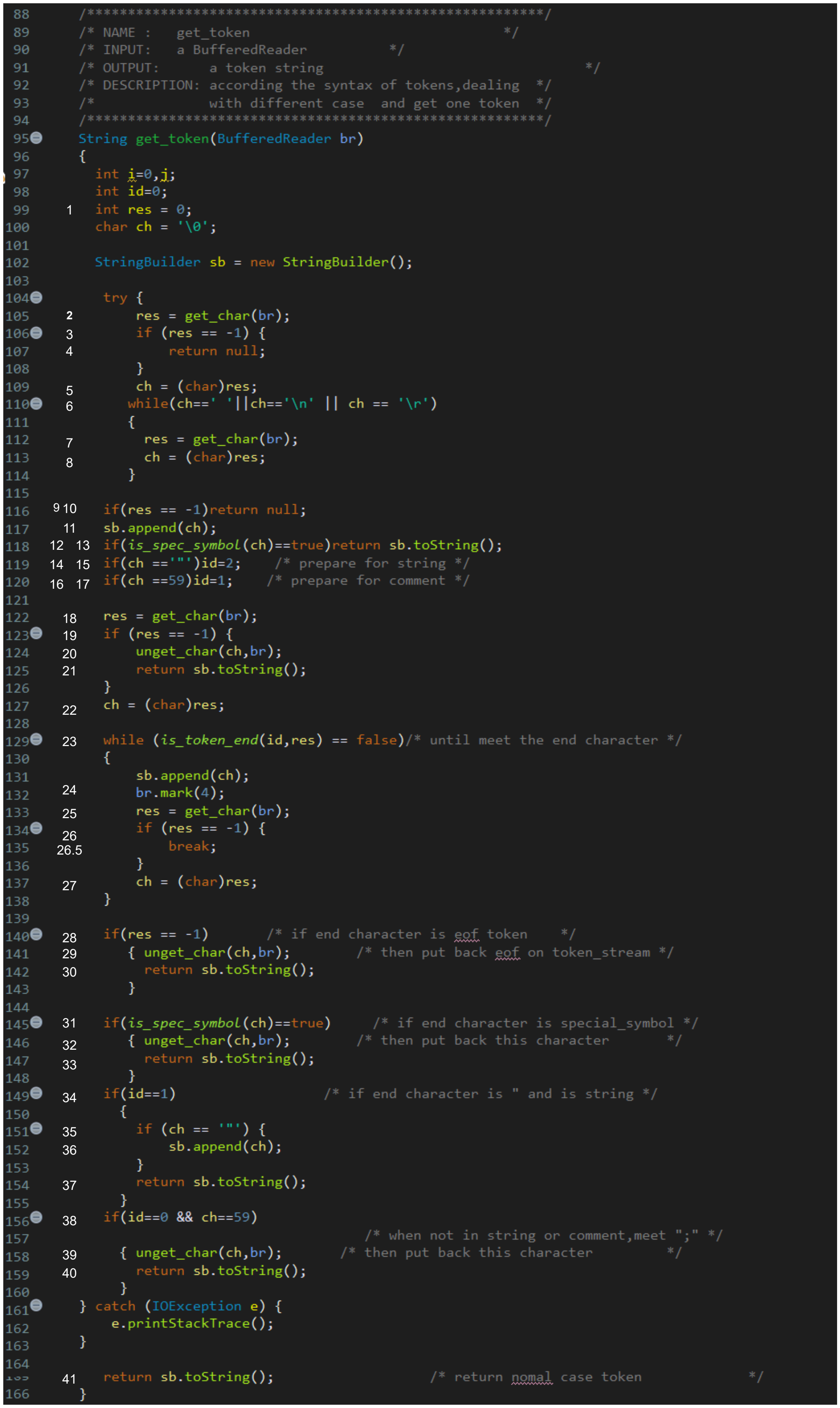
$$\begin{tikzcd}
\text{start} \arrow[r] & 1 \arrow[d] \\
& \text{end}
\end{tikzcd}$$



```
71      /**************************************************/
72      /* NAME:open_token_stream                          */
73      /* INPUT:        a filename                            */
74      /* OUTPUT:       a BufferedReader                */
75      /* DESCRIPTION: when filename is EMPTY,choice standard  */
76      /*              input device as input source          */
77      /**************************************************/
78      BufferedReader open_token_stream(String fname)
79      {
80          BufferedReader br;
81      if(fname==null || fname.equals(""))
82          br=open_character_stream(null);
83       else
84          br=open_character_stream(fname);
85       return br;
86      }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	80, 81	80	81	(none)
2	82	82	82	open_character_stream (null)
3	84	84	84	open_character_stream (fnamel)
4	85	85	85	(none)

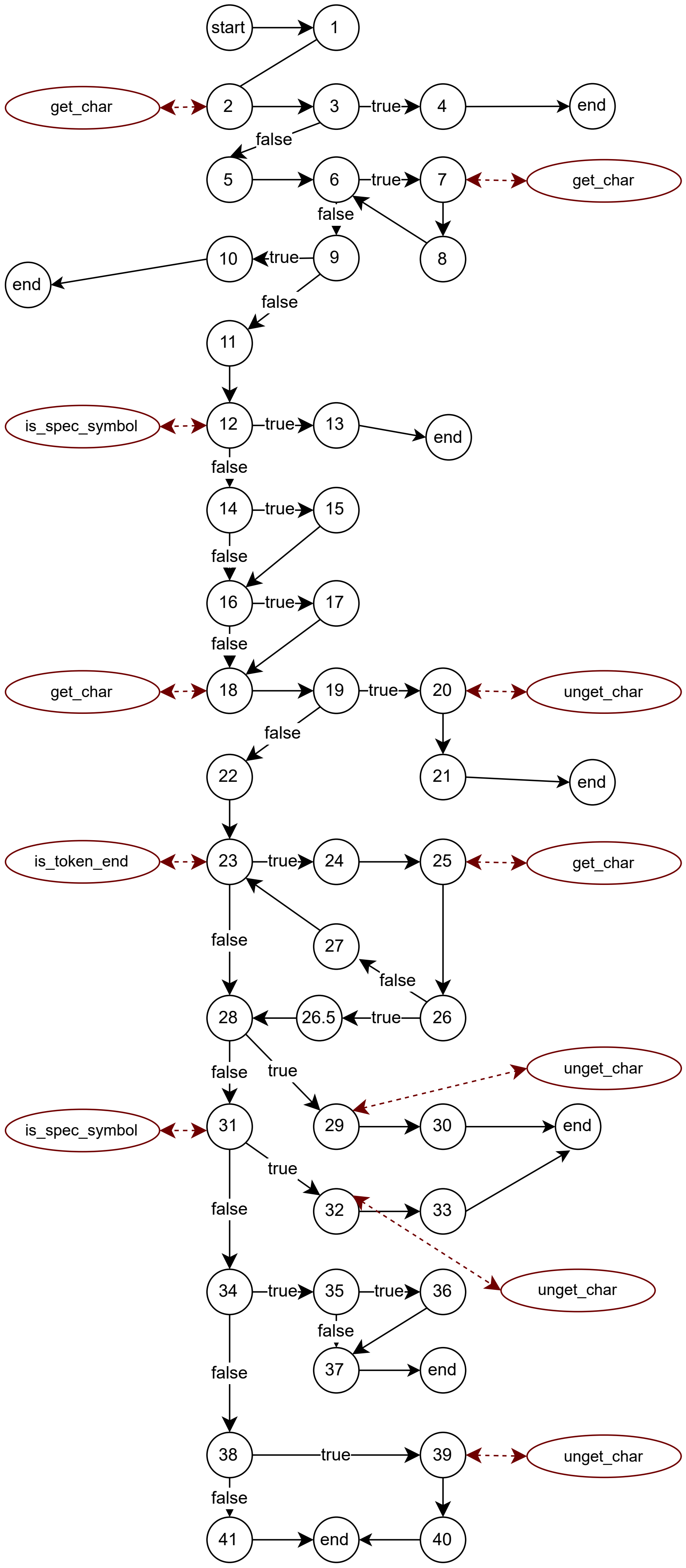


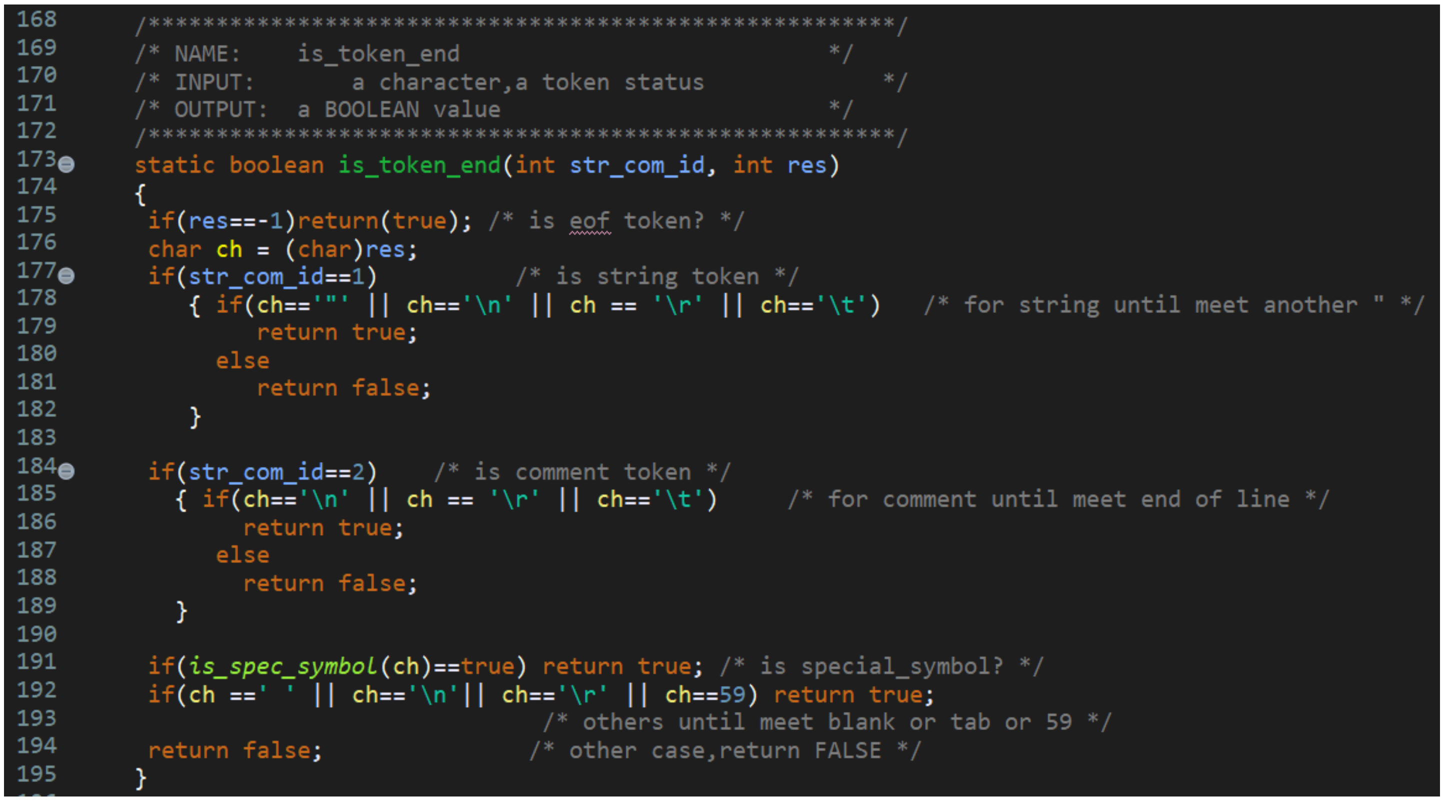


```
        /******************************************************/
        /* NAME :   get_token                           */
        /* INPUT:   a BufferedReader          */
        /* OUTPUT:      a token string                              */
        /* DESCRIPTION: according the syntax of tokens,dealing  */
        /*              with different case  and get one token  */
        /******************************************************/
        String get_token(BufferedReader br)
        {
          int i=0,j;
          int id=0;
1         int res = 0;
          char ch = '\0';

          StringBuilder sb = new StringBuilder();

           try {
2              res = get_char(br);
3              if (res == -1) {
4                  return null;
               }
5              ch = (char)res;
6             while(ch==' '||ch=='\n' || ch == '\r')
              {
7               res = get_char(br);
8               ch = (char)res;
              }

9 10       if(res == -1)return null;
11         sb.append(ch);
12 13      if(is_spec_symbol(ch)==true)return sb.toString();
14 15      if(ch =='"')id=2;    /* prepare for string */
16 17      if(ch ==59)id=1;    /* prepare for comment */

18         res = get_char(br);
19         if (res == -1) {
20             unget_char(ch,br);
21             return sb.toString();
           }
22         ch = (char)res;

23         while (is_token_end(id,res) == false)/* until meet the end character */
           {
               sb.append(ch);
24             br.mark(4);
25             res = get_char(br);
26             if (res == -1) {
26.5               break;
               }
27             ch = (char)res;
           }

28         if(res == -1)        /* if end character is eof token    */
29            { unget_char(ch,br);          /* then put back eof on token_stream */
30              return sb.toString();
              }

31         if(is_spec_symbol(ch)==true)      /* if end character is special_symbol */
32            { unget_char(ch,br);          /* then put back this character       */
33              return sb.toString();
              }
34         if(id==1)                    /* if end character is " and is string */
             {
35             if (ch == '"') {
36                 sb.append(ch);
               }
37             return sb.toString();
             }
38         if(id==0 && ch==59)
                                             /* when not in string or comment,meet ";" */
39           { unget_char(ch,br);         /* then put back this character         */
40             return sb.toString();
             }
        } catch (IOException e) {
            e.printStackTrace();
        }

41         return sb.toString();                 /* return nomal case token              */
        }
```

Block Number	Line numbers	Entry	Exit	Function Calls / return statements
1	97-100,102	97	102	
2	105	105	105	get_char
3	106	106	106	
4	107	107	107	return null
5	109	109	109	
6	110	110	110	
7	112	112	112	get_char
8	113	113	113	
9	116a	116a	116a	
10	116b	116b	116b	return null
11	117	117	117	
12	118a	118a	118a	is_spec_symbol
13	118b	118b	118b	return sb.toString
14	119a	119a	119a	
15	119b	119b	119b	
16	120a	120a	120a	
17	120b	120b	120b	
18	122	122	122	get_char
19	123	123	123	
20	124	124	124	unget_char
21	125	125	125	return sb.toString();
22	127	127	127	
23	129	129	129	is_token_end
24	131, 132	131	132	
25	133	133	133	get_char
26	134, 135	134	135	

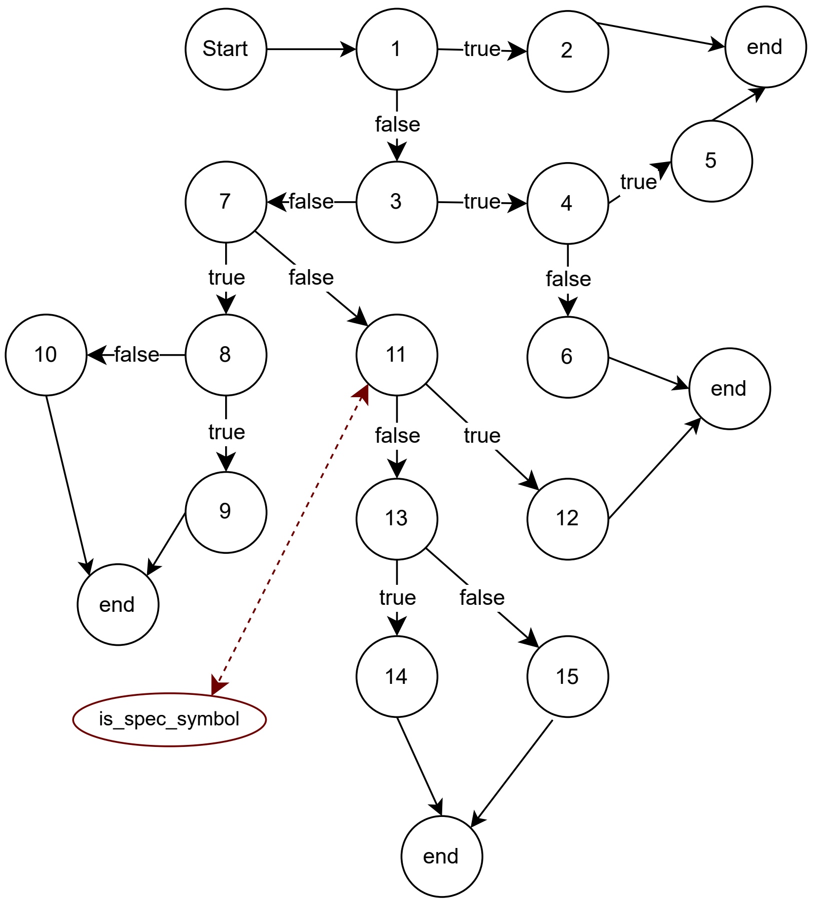
Block Number	Line numbers	Entry	Exit	Function Calls
26.5	135	135	135	return
27	137	137	137	
28	140	140	140	
29	141	141	141	unget_char
30	142	142	142	return sb.toString();
31	145	145	145	is_spec_symbol
32	146	146	146	unget_char
33	147	147	147	return sb.toString();
34	149	149	149	
35	151	151	151	
36	152	152	152	
37	154	154	154	return sb.toString();
38	156	156	156	
39	158	158	158	unget_char
40	159	159	159	return sb.toString();
41	165	165	165	return sb.toString();

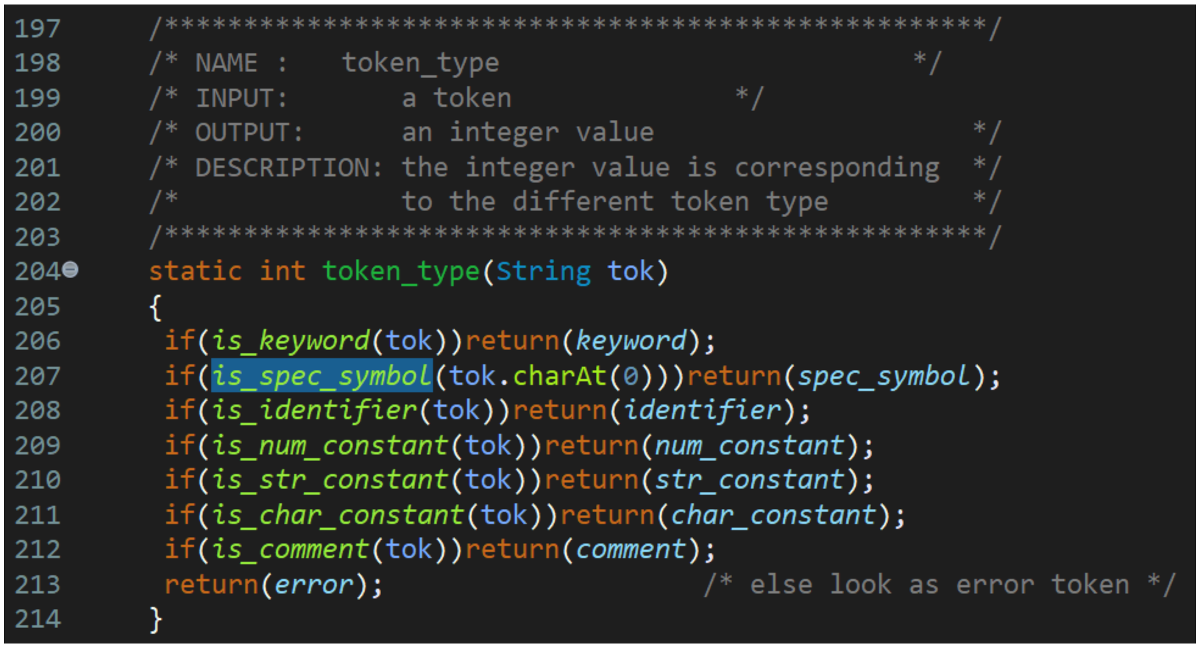

start
1
get_char
2
3
true
4
end
false
5
6
true
7
get_char
false
8
9
true
10
end
false
11
is_spec_symbol
12
true
13
end
false
14
true
15
false
16
true
17
false
get_char
18
19
true
20
unget_char
false
22
21
end
is_token_end
23
true
24
25
get_char
false
27
false
28
26.5
true
26
false
true
unget_char
is_spec_symbol
31
29
30
end
true
false
32
33
unget_char
34
true
35
true
36
false
false
37
end
38
true
39
unget_char
false
41
end
40



```
168  /*****************************************************/
169  /* NAME:    is_token_end                            */
170  /* INPUT:       a character,a token status              */
171  /* OUTPUT:  a BOOLEAN value                         */
172  /*****************************************************/
173  static boolean is_token_end(int str_com_id, int res)
174  {
175   if(res==-1)return(true); /* is eof token? */
176   char ch = (char)res;
177   if(str_com_id==1)          /* is string token */
178      { if(ch=='"' || ch=='\n' || ch == '\r' || ch=='\t')   /* for string until meet another " */
179           return true;
180        else
181           return false;
182      }
183
184   if(str_com_id==2)     /* is comment token */
185     { if(ch=='\n' || ch == '\r' || ch=='\t')      /* for comment until meet end of line */
186          return true;
187        else
188          return false;
189     }
190
191   if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
192   if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59) return true;
193                               /* others until meet blank or tab or 59 */
194   return false;              /* other case,return FALSE */
195  }
```

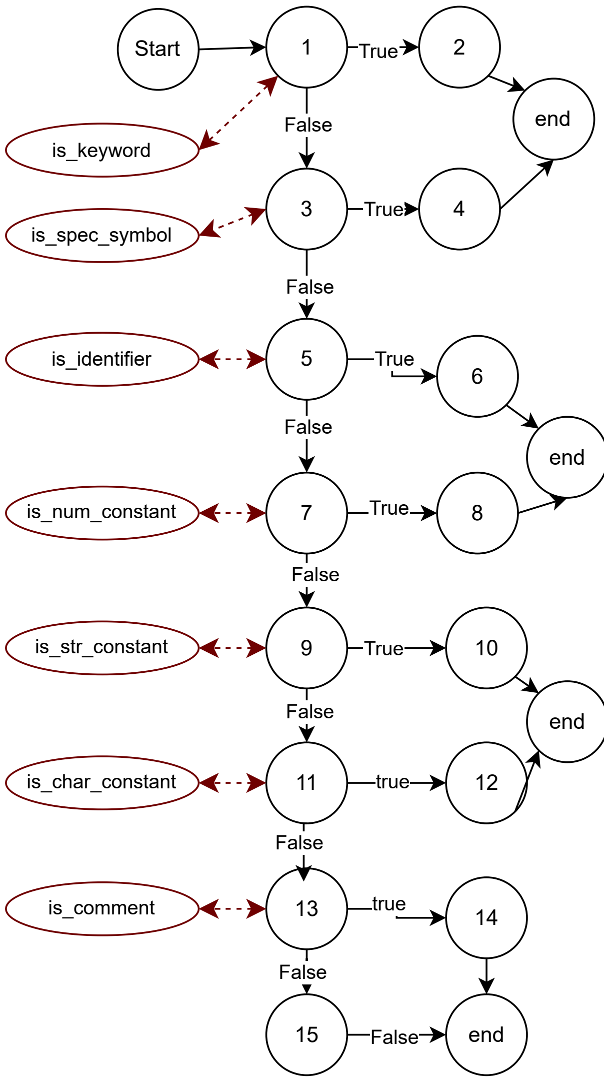
Block Number	Line numbers	Entry	Exit	Function Calls
1	175a	175a	175a	(none)
2	175b	175b	175b	
3	176, 177	176	177	(none)
4	178	178	178	(none)
5	179	179	179	return true;
6	181	181	181	return false;
7	184	184	184	(none)
8	185	185	185	(none)
9	186	186	186	return true;
10	188	188	188	(none)
11	191a	191a	191a	is_spec_symbol()
12	191b	191b	191b	return true;
13	192a	192a	192a	(none)
14	192b	192b	192b	return true;
15	194	194	194	return false;

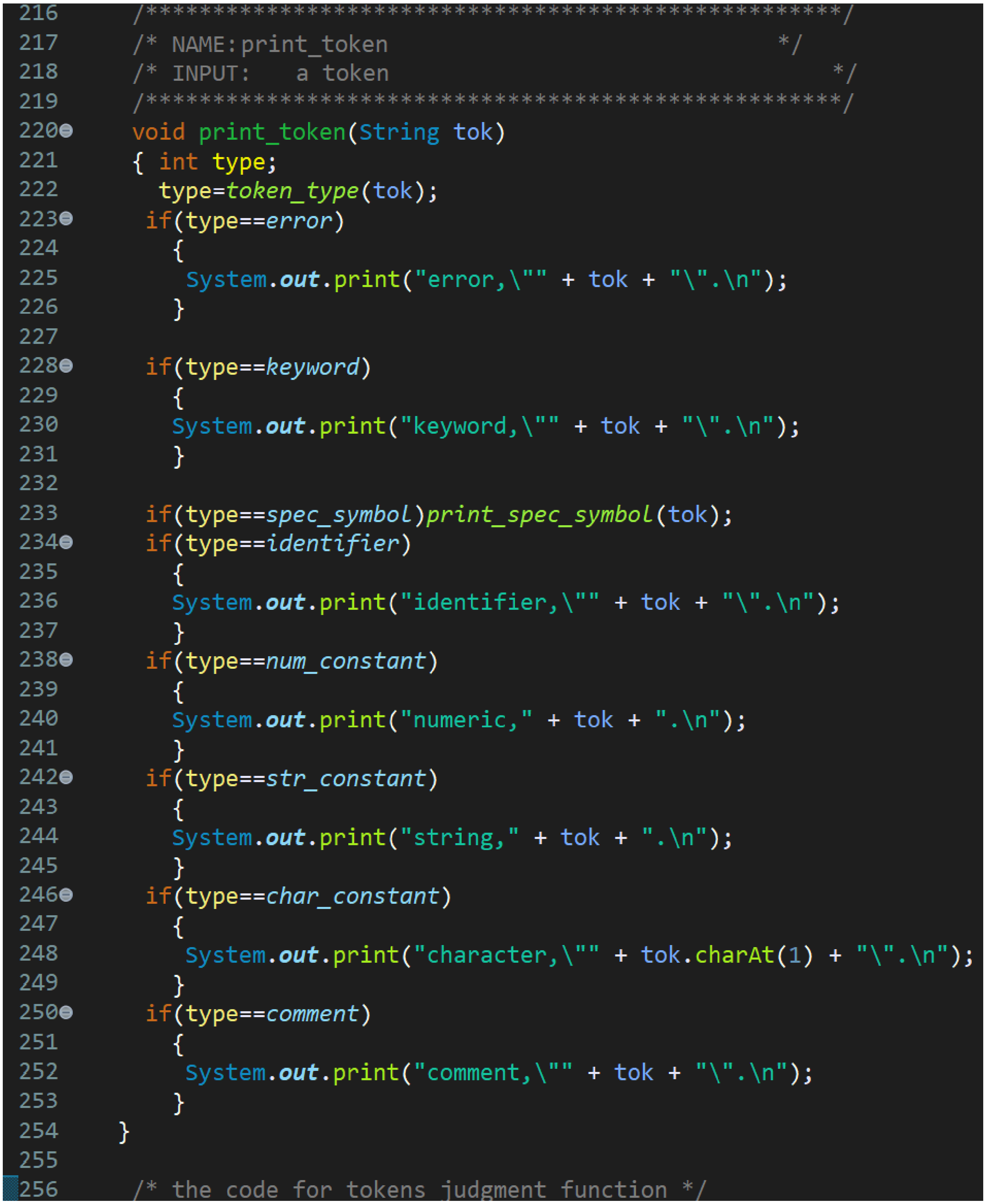




```
197     /*********************************************/
198     /* NAME :   token_type                      */
199     /* INPUT:       a token             */
200     /* OUTPUT:      an integer value                */
201     /* DESCRIPTION: the integer value is corresponding */
202     /*              to the different token type    */
203     /*********************************************/
204     static int token_type(String tok)
205     {
206      if(is_keyword(tok))return(keyword);
207      if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
208      if(is_identifier(tok))return(identifier);
209      if(is_num_constant(tok))return(num_constant);
210      if(is_str_constant(tok))return(str_constant);
211      if(is_char_constant(tok))return(char_constant);
212      if(is_comment(tok))return(comment);
213      return(error);                    /* else look as error token */
214     }
```

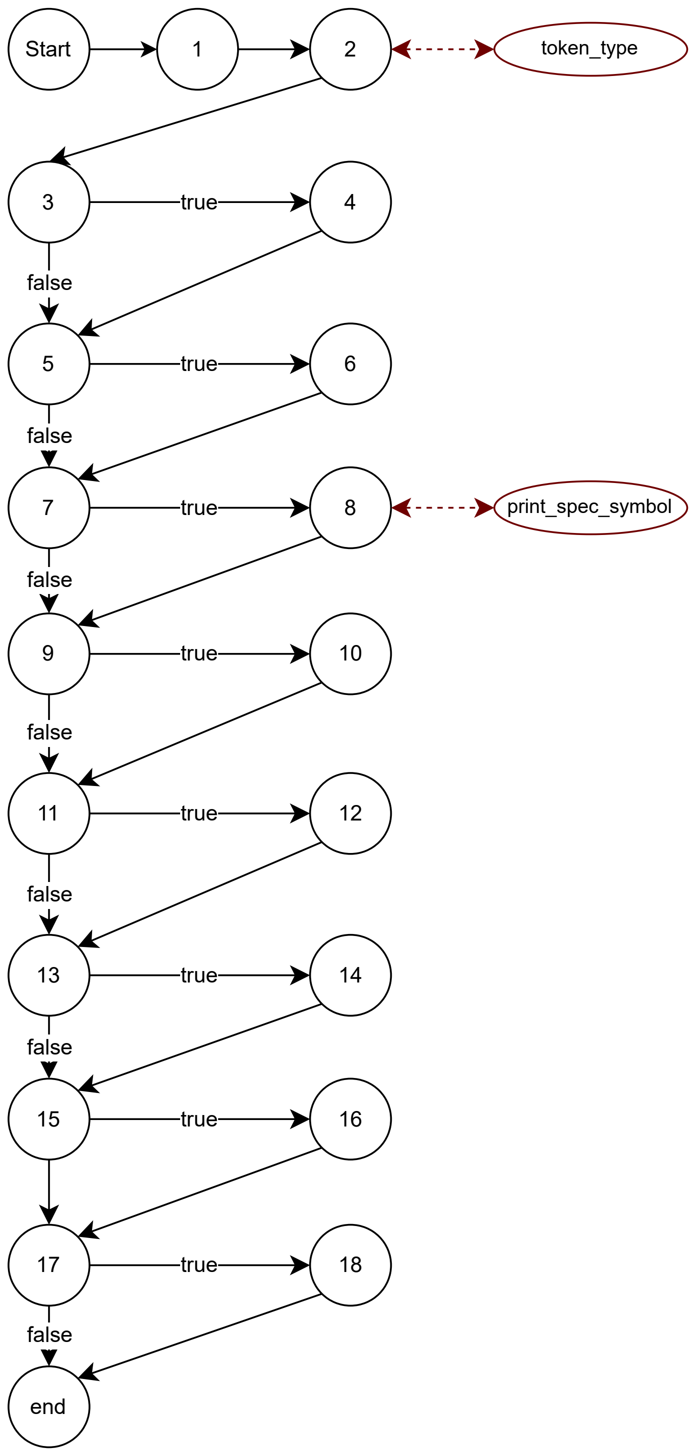
Block Number	Line numbers	Entry	Exit	Function Calls
1	206a	206a	206a	is_keyword(tok)
2	206b	206b	206b	return(keyword);
3	207a	207a	207a	is_spec_symbol(tok.charAt(0))
4	207b	207b	207b	return(spec_symbol);
5	208a	208a	208a	is_identifier(tok)
6	208b	208b	208b	return(identifier);
7	209a	209a	209a	is_num_constant(tok)
8	209b	209b	209b	return(num_constant);
9	210a	210a	210a	is_str_constant(tok)
10	210b	210b	210b	return(str_constant);
11	211a	211a	211a	is_char_constant(tok)
12	211b	211b	211b	return(char_constant);
13	212a	212a	212a	is_comment(tok)
14	212b	212b	212b	return(comment);
15	213	213	213	return(error);

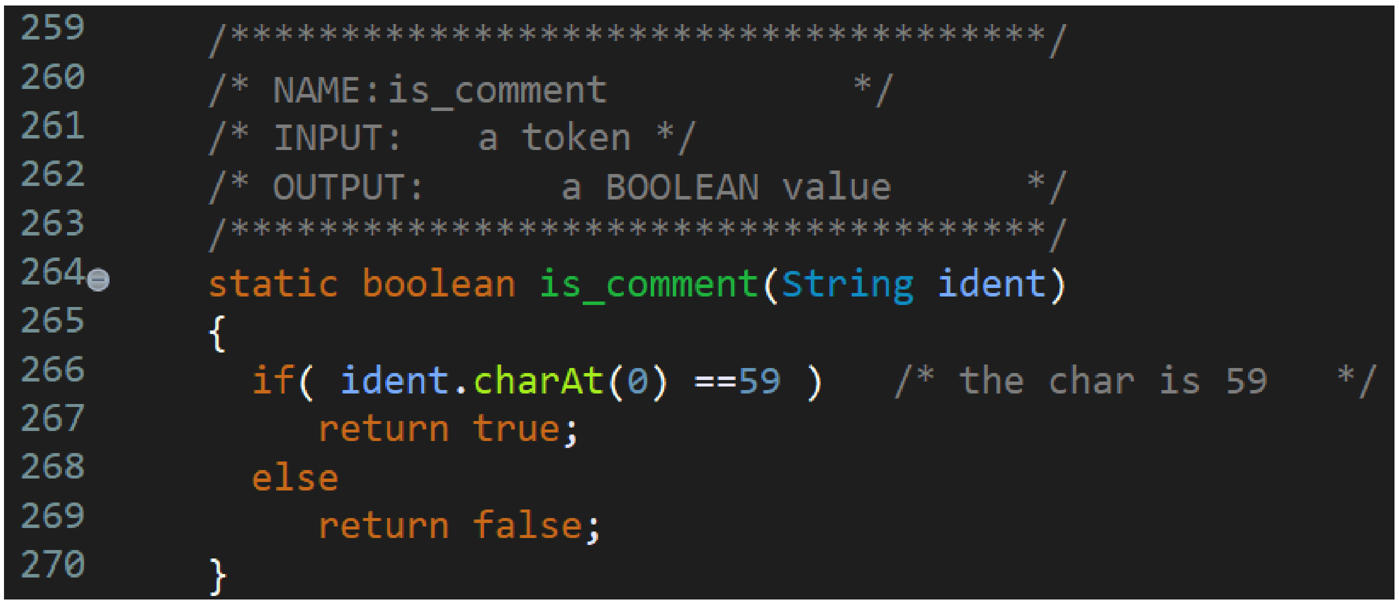




```
 /****************************************************/
 /* NAME:print_token                             */
 /* INPUT:   a token                                 */
 /****************************************************/
 void print_token(String tok)
 { int type;
   type=token_type(tok);
  if(type==error)
    {
     System.out.print("error,\"" + tok + "\".\n");
    }

  if(type==keyword)
    {
    System.out.print("keyword,\"" + tok + "\".\n");
    }

  if(type==spec_symbol)print_spec_symbol(tok);
  if(type==identifier)
    {
    System.out.print("identifier,\"" + tok + "\".\n");
    }
  if(type==num_constant)
    {
    System.out.print("numeric," + tok + ".\n");
    }
  if(type==str_constant)
    {
    System.out.print("string," + tok + ".\n");
    }
  if(type==char_constant)
    {
     System.out.print("character,\"" + tok.charAt(1) + "\".\n");
    }
  if(type==comment)
    {
     System.out.print("comment,\"" + tok + "\".\n");
    }
}

 /* the code for tokens judgment function */
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	221	221	221	(none)
2	222	222	222	token_type(tok)
3	223	223	223	(none)
4	225	225	225	(none)
5	228	228	228	(none)
6	230	230	230	(none)
7	233a	233a	233a	(none)
8	233b	233b	233b	print_spec_symbol(tok)
9	234	234	234	(none)
10	236	236	236	(none)
11	238	238	238	(none)
12	240	240	240	(none)
13	242	242	242	(none)
14	244	244	244	(none)
15	246	246	246	(none)
16	248	248	248	(none)
17	250	250	250	(none)
18	252	252	252	(none)

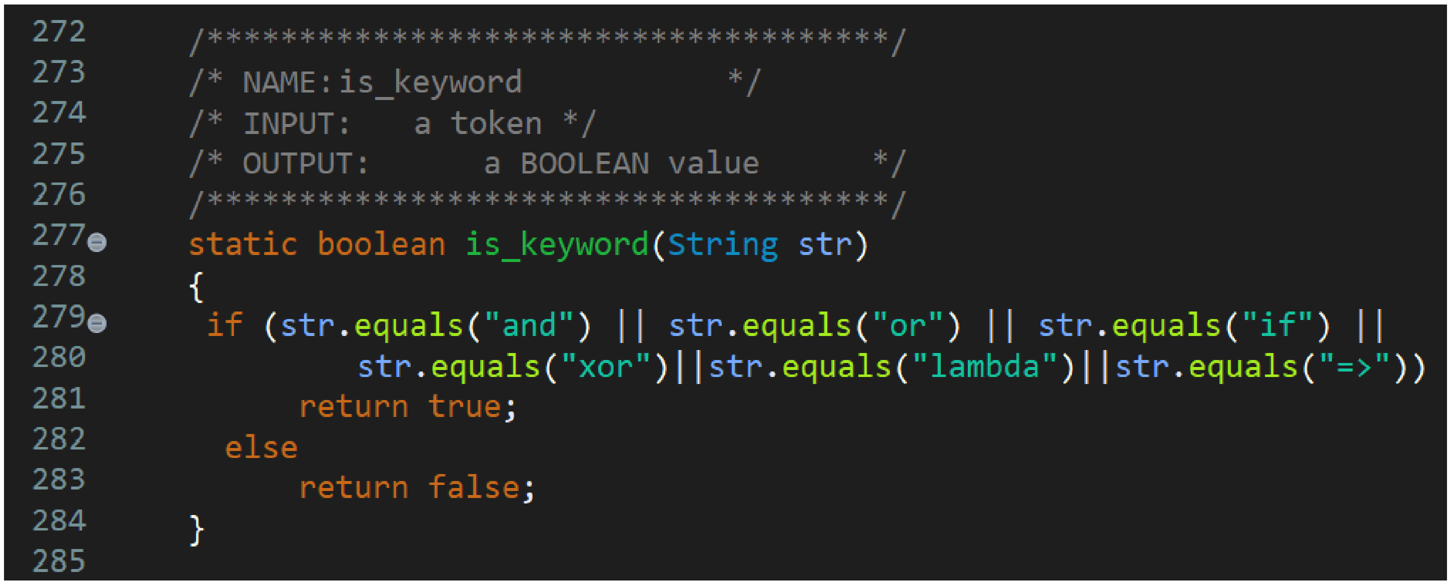




```
259         /*************************************/
260         /* NAME:is_comment           */
261         /* INPUT:    a token */
262         /* OUTPUT:      a BOOLEAN value      */
263         /*************************************/
264         static boolean is_comment(String ident)
265         {
266           if( ident.charAt(0) ==59 )    /* the char is 59   */
267              return true;
268           else
269              return false;
270         }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	266	266	266	(none)
2	267	267	267	return true;
3	269	269	269	return false;

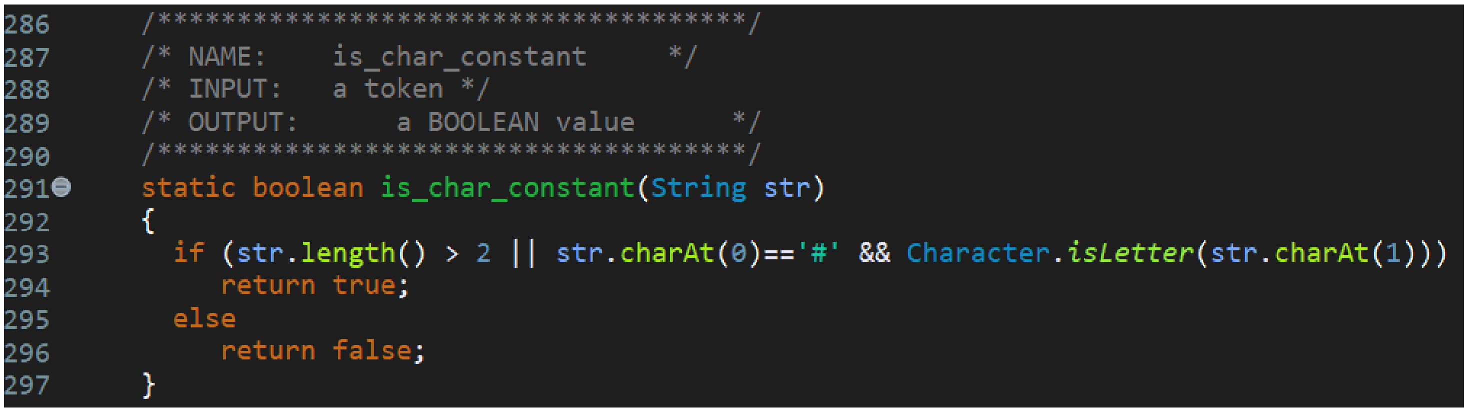
$$\begin{tikzcd}
\text{Start} \arrow[rr] & & 1 \arrow[dl, "\text{True}"'] \arrow[dr, "\text{False}"] & \\
& 2 \arrow[dr] & & 3 \arrow[dl] \\
& & \text{End} &
\end{tikzcd}$$



```
272     /***********************************/
273     /* NAME:is_keyword           */
274     /* INPUT:    a token */
275     /* OUTPUT:       a BOOLEAN value      */
276     /***********************************/
277     static boolean is_keyword(String str)
278     {
279      if (str.equals("and") || str.equals("or") || str.equals("if") ||
280              str.equals("xor")||str.equals("lambda")||str.equals("=>"))
281            return true;
282        else
283            return false;
284     }
285
```

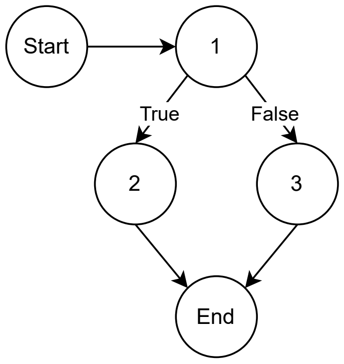
Block Number	Line numbers	Entry	Exit	Function Calls
1	279, 280	279	280	(none)
2	281	281	281	return true;
3	283	283	283	return false;

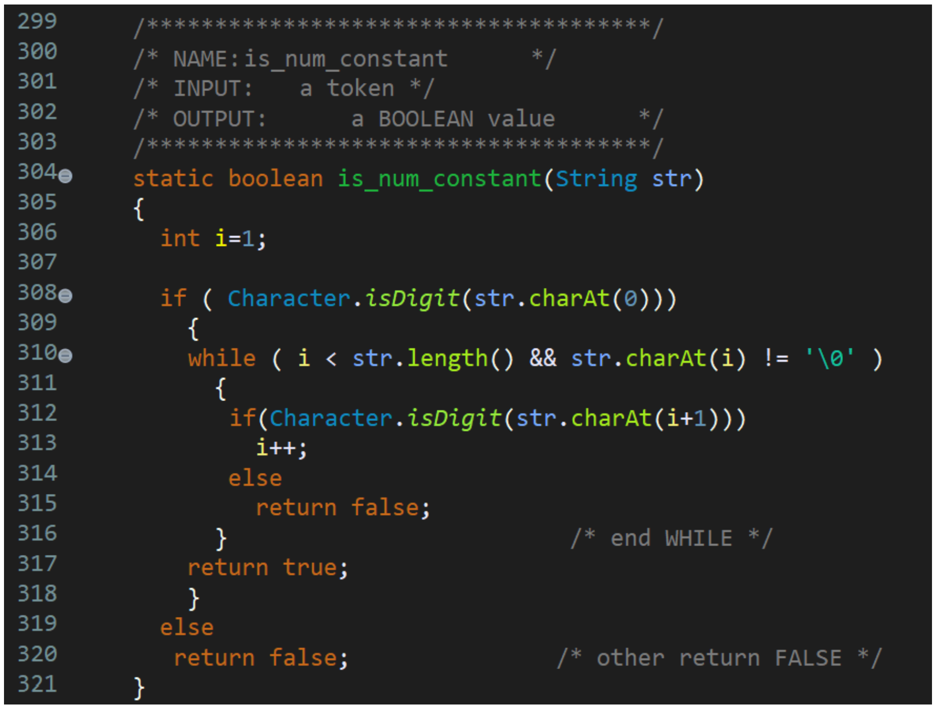
$$\begin{tikzcd}
\text{Start} \arrow[rr] & & 1 \arrow[dl, "\text{True}"'] \arrow[dr, "\text{False}"] & \\
& 2 \arrow[dr] & & 3 \arrow[dl] \\
& & \text{End} &
\end{tikzcd}$$



```
286       /************************************/
287       /* NAME:    is_char_constant     */
288       /* INPUT:   a token */
289       /* OUTPUT:      a BOOLEAN value      */
290       /************************************/
291       static boolean is_char_constant(String str)
292       {
293         if (str.length() > 2 || str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
294            return true;
295         else
296            return false;
297       }
```

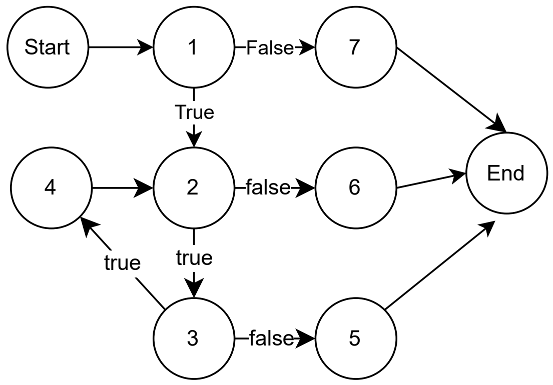
Block Number	Line numbers	Entry	Exit	Function Calls
1	293	293	293	(none)
2	294	294	294	return true;
3	296	296	296	return false;

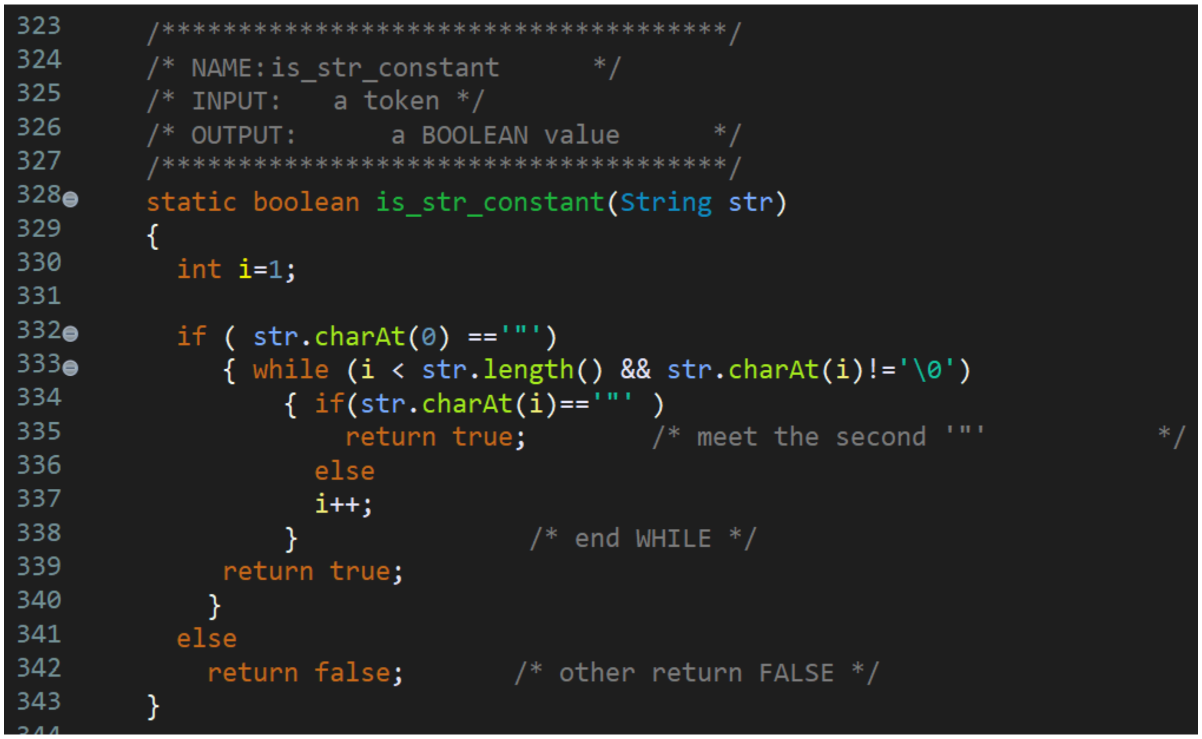




```
299   /*************************************/
300   /* NAME:is_num_constant      */
301   /* INPUT:    a token */
302   /* OUTPUT:       a BOOLEAN value      */
303   /*************************************/
304   static boolean is_num_constant(String str)
305   {
306     int i=1;
307
308     if ( Character.isDigit(str.charAt(0)))
309       {
310       while ( i < str.length() && str.charAt(i) != '\0' )
311         {
312           if(Character.isDigit(str.charAt(i+1)))
313             i++;
314           else
315             return false;
316         }                         /* end WHILE */
317       return true;
318       }
319     else
320      return false;                /* other return FALSE */
321   }
```

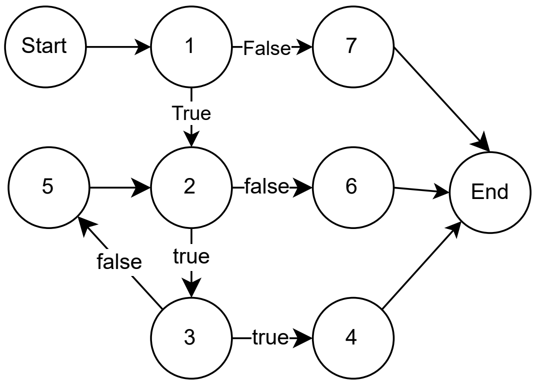
Block Number	Line numbers	Entry	Exit	Function Calls
1	306, 308	306	308	none
2	310	310	310	none
3	312	312	312	none
4	313	313	313	(none)
5	315	315	315	return false;
6	317	317	317	return true;
7	320	320	320	return false;



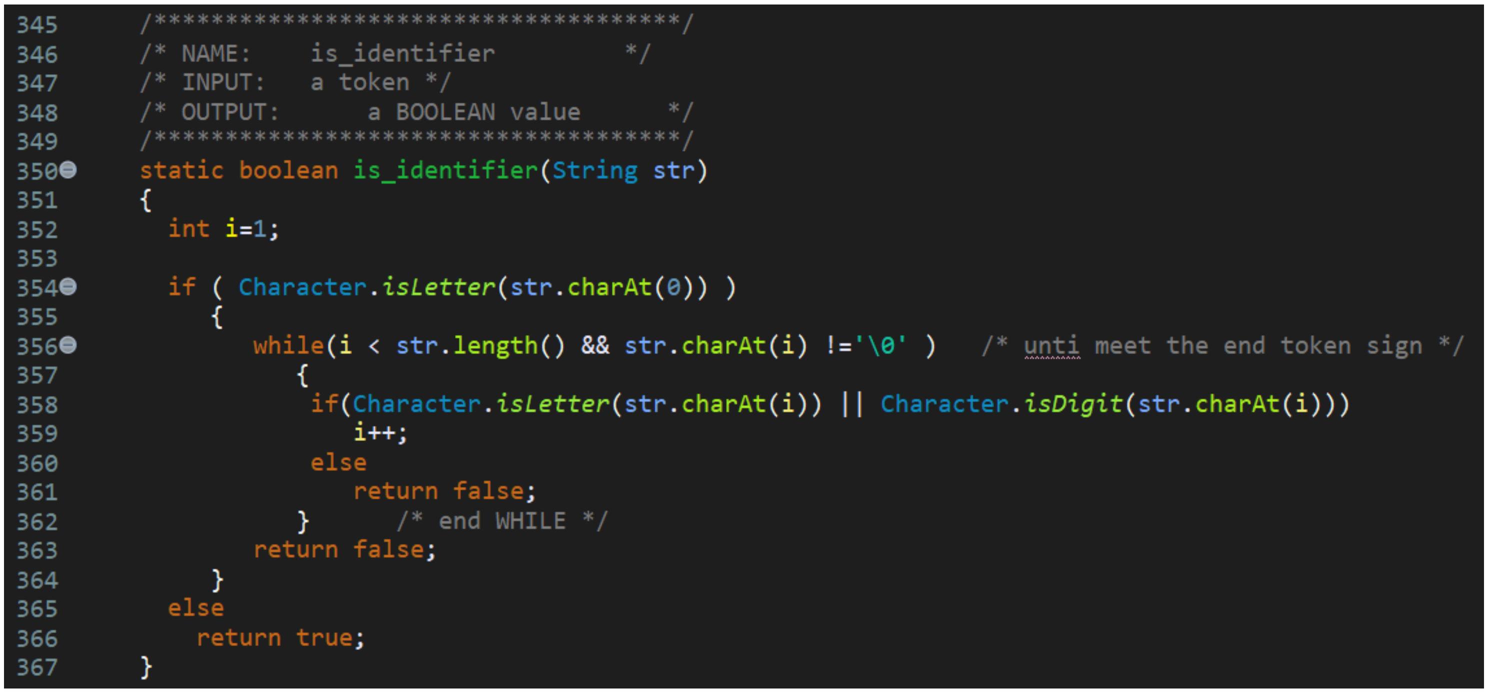


```
323      /************************************/
324      /* NAME:is_str_constant      */
325      /* INPUT:    a token */
326      /* OUTPUT:       a BOOLEAN value      */
327      /************************************/
328      static boolean is_str_constant(String str)
329      {
330        int i=1;
331
332        if ( str.charAt(0) =='"')
333           { while (i < str.length() && str.charAt(i)!='\0')
334               { if(str.charAt(i)=='"' )
335                   return true;        /* meet the second '"'            */
336                 else
337                 i++;
338               }               /* end WHILE */
339           return true;
340          }
341        else
342          return false;        /* other return FALSE */
343      }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	330, 332	330	332	none
2	333	333	333	none
3	334	334	334	none
4	335	335	335	return true;
5	337	337	337	(none)
6	339	339	339	return true;
7	342	342	342	return false;

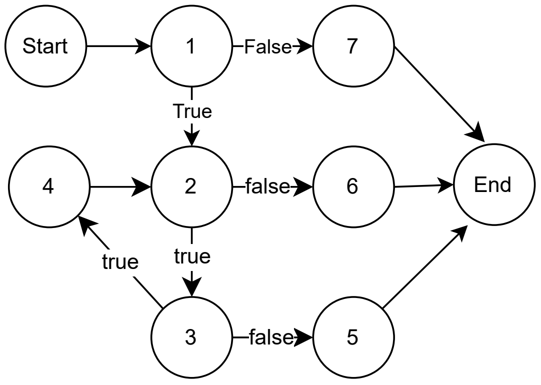


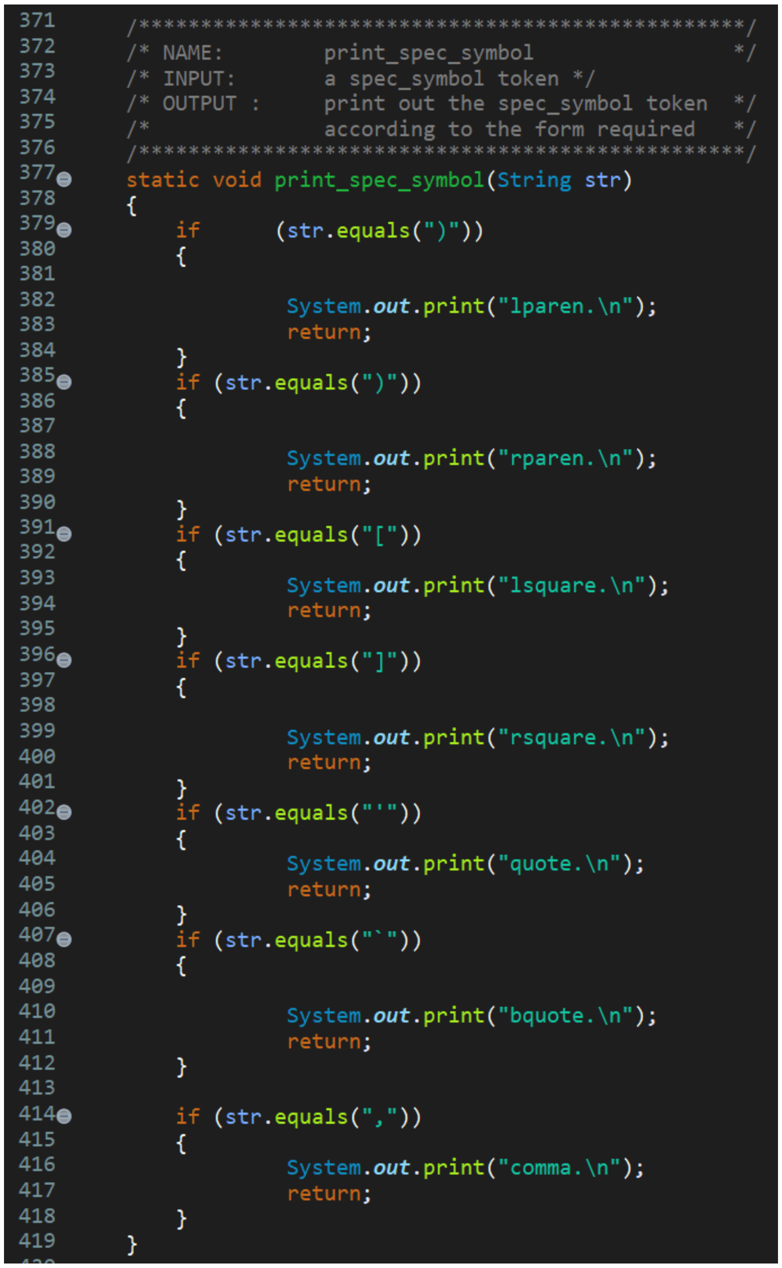
$$\begin{tikzcd}
\text{Start} \arrow[r] & 1 \arrow[r, "\text{False}"] \arrow[d, "\text{True}"] & 7 \arrow[dr] & \\
5 \arrow[r] & 2 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 6 \arrow[r] & \text{End} \\
 & 3 \arrow[ul, "\text{false}"] \arrow[r, "\text{true}"] & 4 \arrow[ur] &
\end{tikzcd}$$



```
345     /**************************************/
346     /* NAME:    is_identifier         */
347     /* INPUT:   a token */
348     /* OUTPUT:      a BOOLEAN value      */
349     /**************************************/
350     static boolean is_identifier(String str)
351     {
352       int i=1;
353
354       if ( Character.isLetter(str.charAt(0)) )
355         {
356           while(i < str.length() && str.charAt(i) !='\0' )    /* unti meet the end token sign */
357             {
358               if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
359                 i++;
360               else
361                 return false;
362             }       /* end WHILE */
363           return false;
364         }
365       else
366         return true;
367     }
```

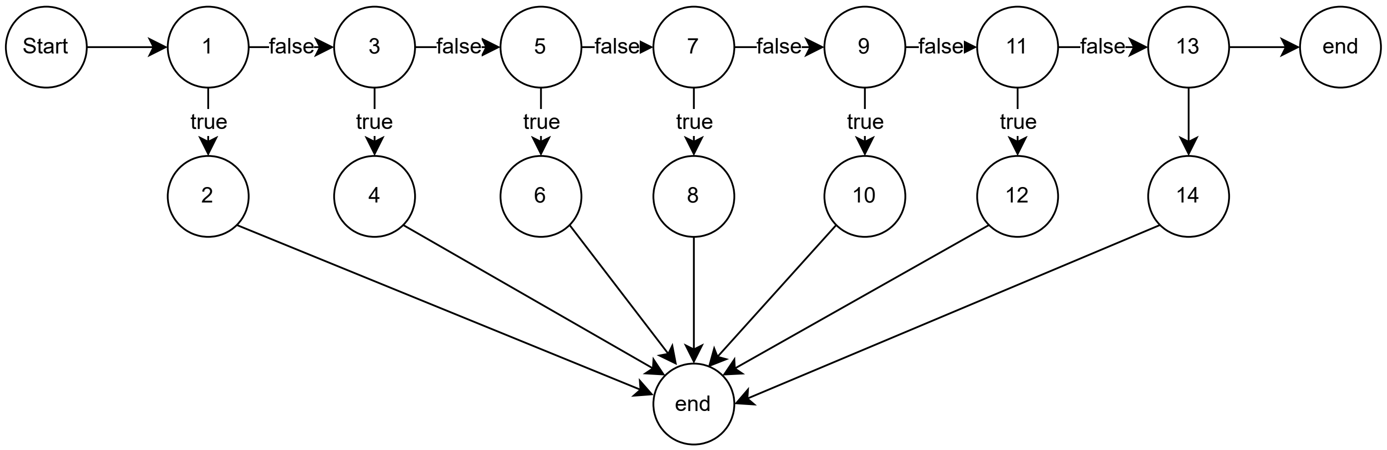
Block Number	Line numbers	Entry	Exit	Function Calls
1	352,354	352	354	(none)
2	356	356	356	(none)
3	358	358	358	(none)
4	359	359	359	(none)
5	361	361	361	return false;
6	363	363	363	return false;
7	366	366	366	return true;



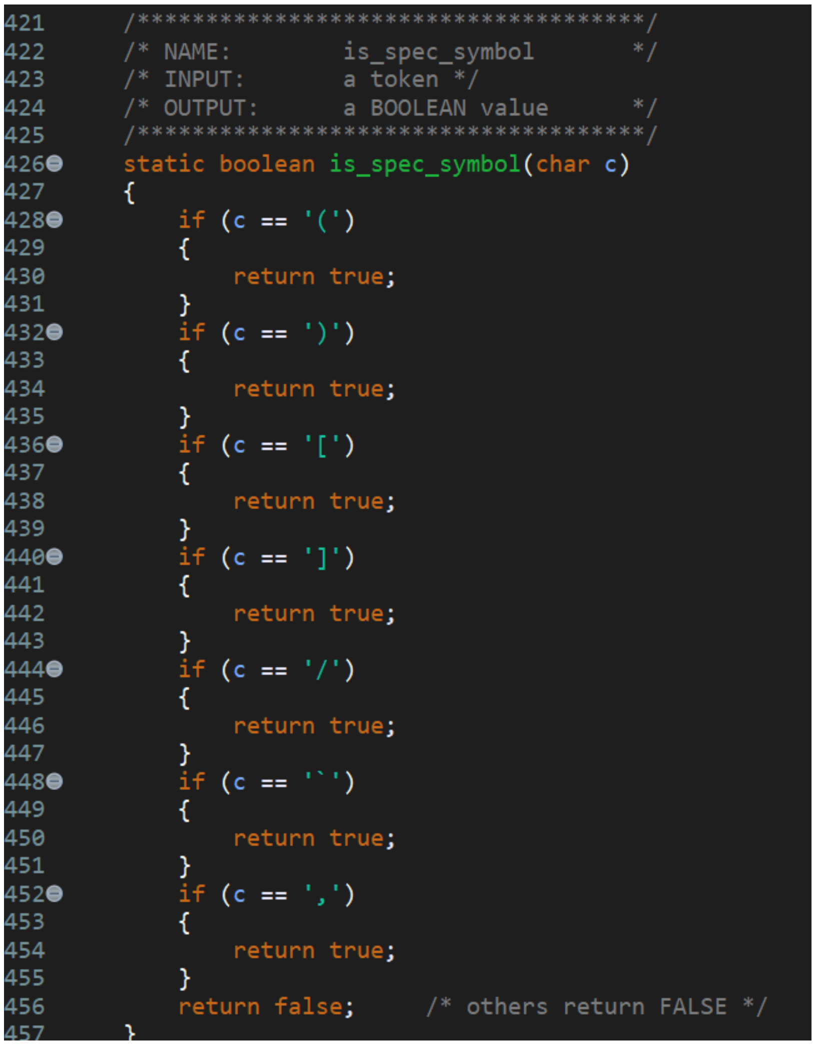


```
371        /************************************************/
372        /* NAME:        print_spec_symbol               */
373        /* INPUT:       a spec_symbol token */
374        /* OUTPUT :     print out the spec_symbol token */
375        /*              according to the form required  */
376        /************************************************/
377        static void print_spec_symbol(String str)
378        {
379            if      (str.equals(")"))
380            {
381
382                     System.out.print("lparen.\n");
383                     return;
384            }
385            if (str.equals(")"))
386            {
387
388                     System.out.print("rparen.\n");
389                     return;
390            }
391            if (str.equals("["))
392            {
393                     System.out.print("lsquare.\n");
394                     return;
395            }
396            if (str.equals("]"))
397            {
398
399                     System.out.print("rsquare.\n");
400                     return;
401            }
402            if (str.equals("'"))
403            {
404                     System.out.print("quote.\n");
405                     return;
406            }
407            if (str.equals("`"))
408            {
409
410                     System.out.print("bquote.\n");
411                     return;
412            }
413
414            if (str.equals(","))
415            {
416                     System.out.print("comma.\n");
417                     return;
418            }
419        }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	379	379	379	none
2	382, 383	382	383	return;
3	385	385	385	none
4	388, 389	388	389	return;
5	391	391	391	(none)
6	393, 394	393	394	return;
7	396	396	396	(none)
8	399, 400	399	400	return;
9	402	402	402	(none)
10	404, 405	404	405	return;
11	407	407	407	(none)
12	410, 411	410	411	return;
13	414	414	414	(none)
14	416, 417	416	417	return;

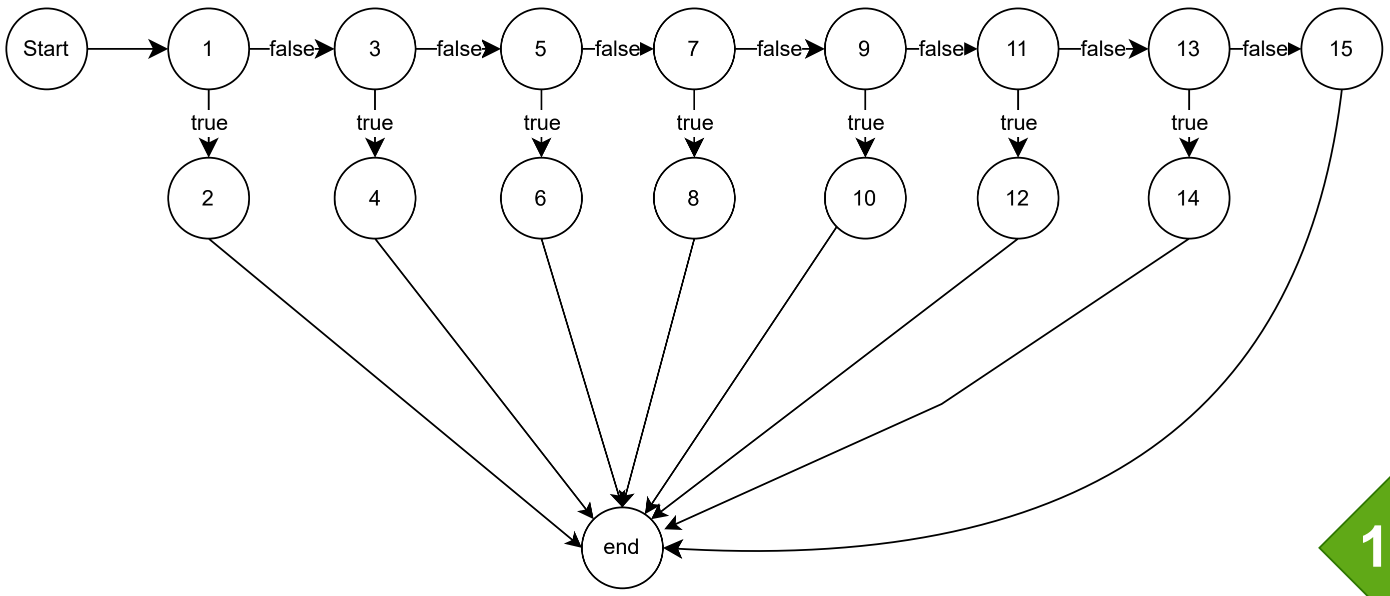


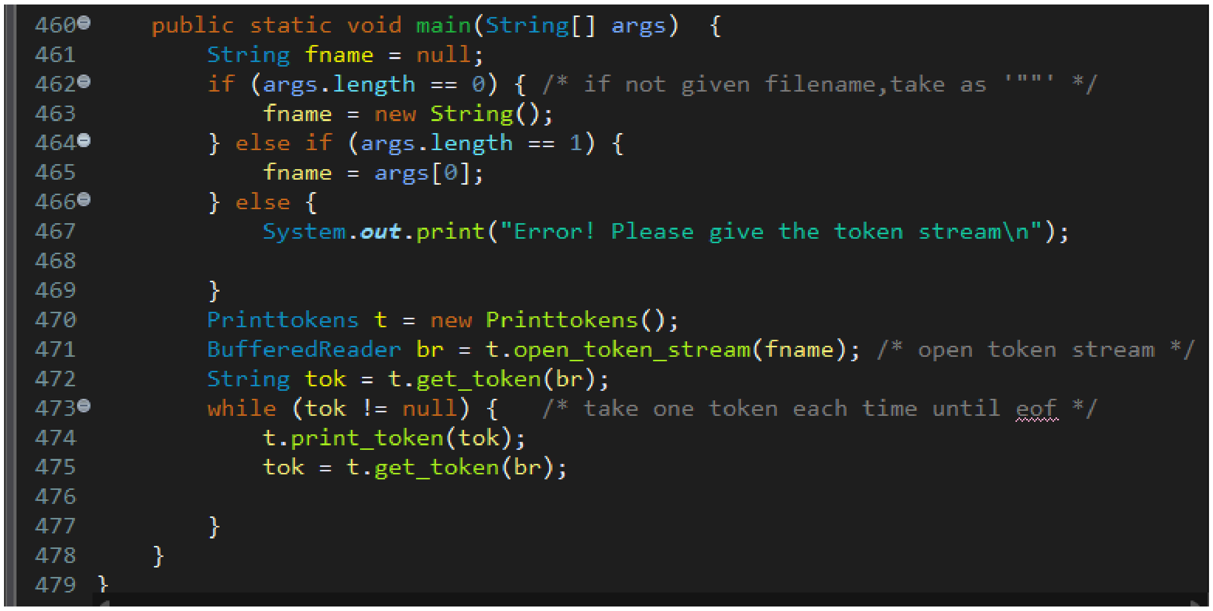
$$\begin{tikzcd}
\text{Start} \arrow[r] & 1 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 3 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 5 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 7 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 9 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 11 \arrow[r, "\text{false}"] \arrow[d, "\text{true}"] & 13 \arrow[r] \arrow[d] & \text{end} \\
 & 2 \arrow[drrr] & 4 \arrow[drr] & 6 \arrow[dr] & 8 \arrow[d] & 10 \arrow[dl] & 12 \arrow[dll] & 14 \arrow[dlll] & \\
 & & & & \text{end} & & & &
\end{tikzcd}$$



```
421     /************************************/
422     /* NAME:         is_spec_symbol       */
423     /* INPUT:        a token */
424     /* OUTPUT:       a BOOLEAN value      */
425     /************************************/
426     static boolean is_spec_symbol(char c)
427     {
428         if (c == '(')
429         {
430             return true;
431         }
432         if (c == ')')
433         {
434             return true;
435         }
436         if (c == '[')
437         {
438             return true;
439         }
440         if (c == ']')
441         {
442             return true;
443         }
444         if (c == '/')
445         {
446             return true;
447         }
448         if (c == '`')
449         {
450             return true;
451         }
452         if (c == ',')
453         {
454             return true;
455         }
456         return false;     /* others return FALSE */
457     }
```

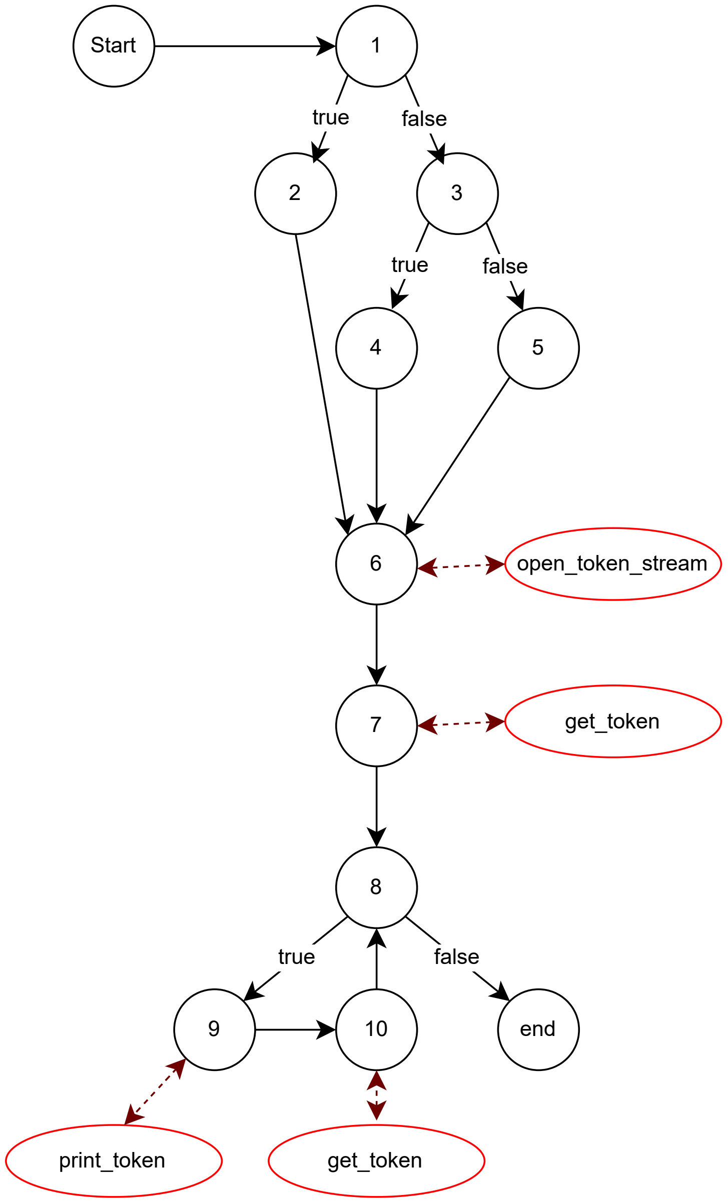
Block Number	Line numbers	Entry	Exit	Function Calls
1	428	428	428	(none)
2	430	430	430	return true;
3	432	432	432	(none)
4	434	434	434	return true;
5	436	436	436	(none)
6	438	438	438	return true;
7	440	440	440	(none)
8	442	442	442	return true;
9	444	444	444	(none)
10	446	446	446	return true;
11	448	448	448	(none)
12	450	450	450	return true;
13	452	452	452	(none)
14	454	454	454	return true;
15	456	456	456	return false;





```
460     public static void main(String[] args)  {
461         String fname = null;
462         if (args.length == 0) { /* if not given filename,take as '""' */
463             fname = new String();
464         } else if (args.length == 1) {
465             fname = args[0];
466         } else {
467             System.out.print("Error! Please give the token stream\n");
468 
469         }
470         Printtokens t = new Printtokens();
471         BufferedReader br = t.open_token_stream(fname); /* open token stream */
472         String tok = t.get_token(br);
473         while (tok != null) {   /* take one token each time until eof */
474             t.print_token(tok);
475             tok = t.get_token(br);
476 
477         }
478     }
479 }
```

Block Number	Line numbers	Entry	Exit	Function Calls
1	460,461	460	461	
2	462	462	462	
3	463	463	463	
4	464	464	464	
5	466	466	466	
6	468, 469	468	469	open_token _stream
7	470	470	470	get_token
8	471	471	471	
9	472	472	472	print_token
10	473	473	473	get_token



Start
1
true
false
2
3
true
false
4
5
6
open_token_stream
7
get_token
8
true
false
9
10
end
print_token
get_token