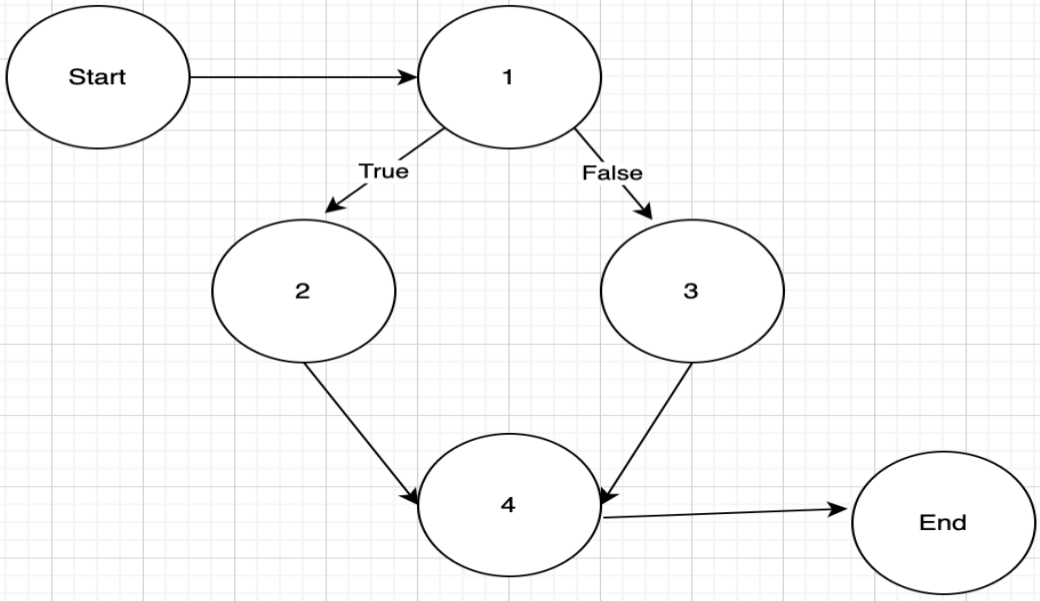
Control Flow Graphs (CFGs)

CFGs along with the corresponding source code, and the basic block table that identifies each basic block. The code in the catch clauses can be skipped, i.e., these clauses do not need to be represented in the CFG.

1. open_chracter_stream

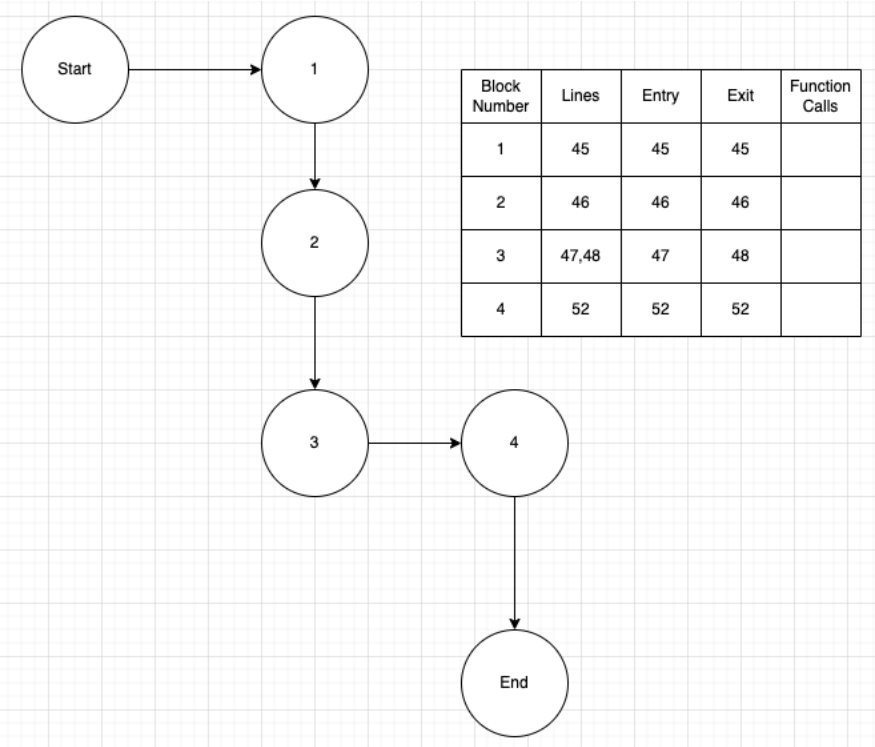
```
22    BufferedReader open_character_stream(String fname) {
23        BufferedReader br = null;
24        if (fname == null) {
25            br = new BufferedReader(new InputStreamReader(System.in));
26        } else {
27            try {
28                FileReader fr = new FileReader(fname);
29                br = new BufferedReader(fr);
30            } catch (FileNotFoundException e) {
31                System.out.print("The file " + fname +" doesn't exists\n")
32                e.printStackTrace();
33            }
34        }
35
36        return br;
37    }
```



Block Number	Lines	Entry	Exit	Function Calls
1	23, 24	23	24	
2	25	25	25	
3	28, 29	28	29	
4	36	36	36	

2. get_char

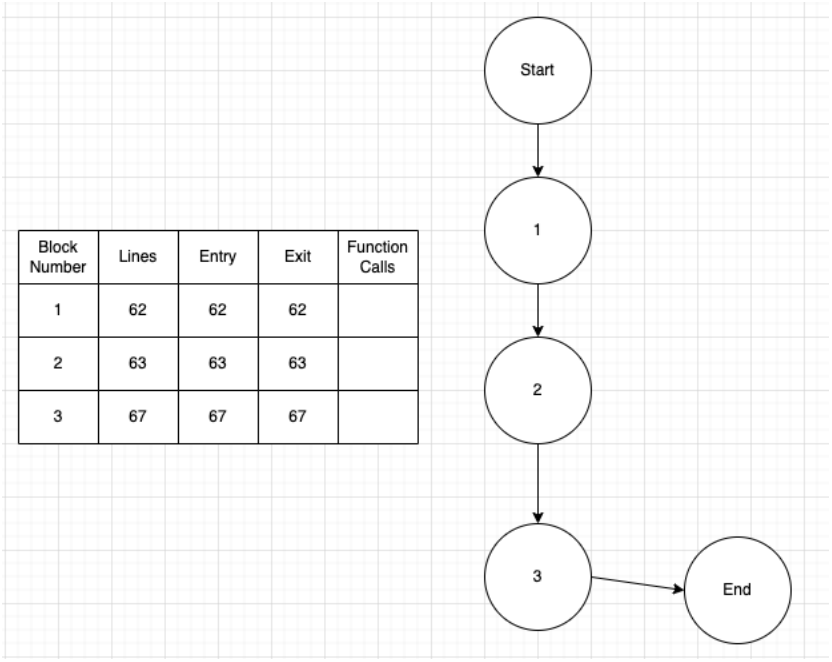
```
44    int get_char(BufferedReader br){
45        int ch = 0;
46        try {
47            br.mark(4);
48            ch= br.read();
49        } catch (IOException e) {
50            e.printStackTrace();
51        }
52        return ch;
53    }
```



Block Number	Lines	Entry	Exit	Function Calls
1	45	45	45	
2	46	46	46	
3	47,48	47	48	
4	52	52	52	

3. unget_char

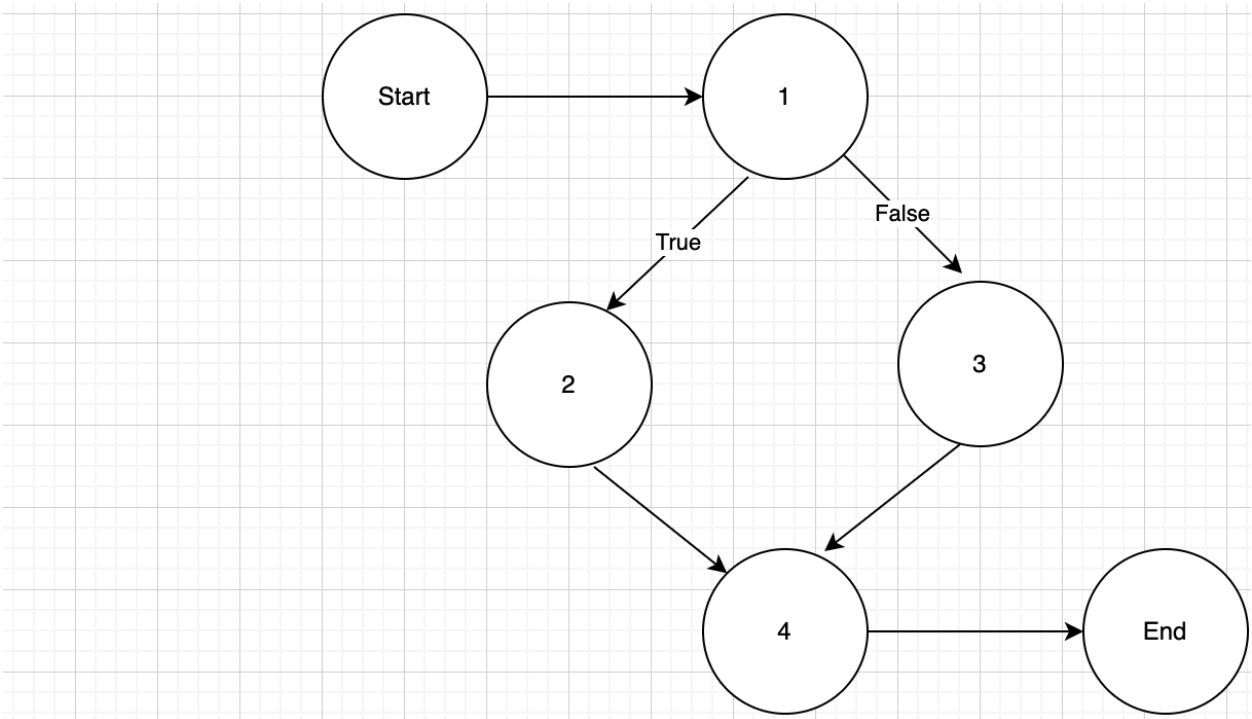
```
61    char unget_char (int ch,BufferedReader br) {
62      try {
63        br.reset();
64    } catch (IOException e) {
65        e.printStackTrace();
66    }
67         return 0;
68    }
```



Block Number	Lines	Entry	Exit	Function Calls
1	62	62	62	
2	63	63	63	
3	67	67	67	

4. open_token_stream

```
77    BufferedReader open_token_stream(String fname)
78    {
79        BufferedReader br;
80    if(fname==null || fname.equals(""))
81        br=open_character_stream(null);
82     else
83        br=open_character_stream(fname);
84     return br;
85    }
```

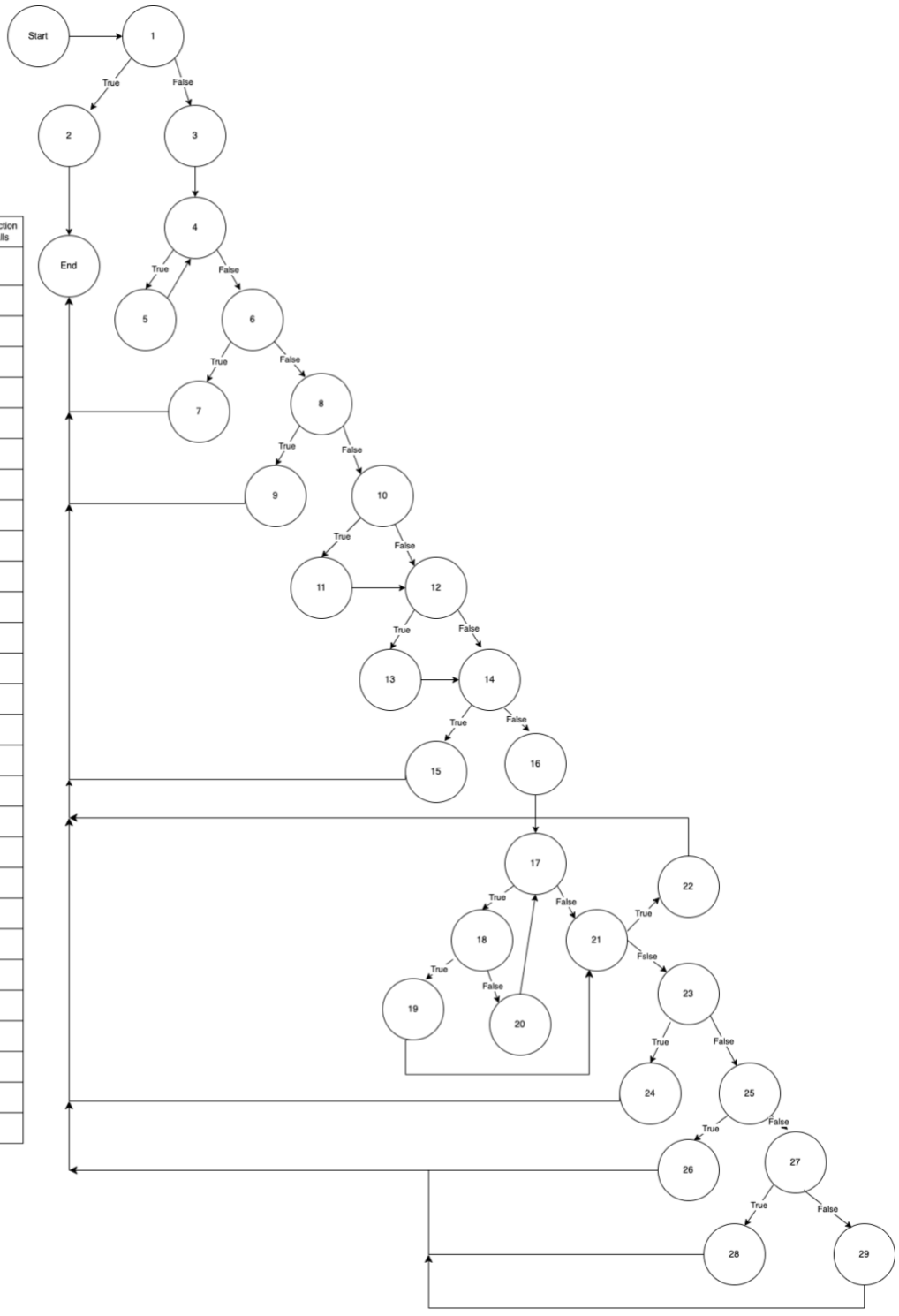


Block Number	Lines	Entry	Exit	Function Calls
1	79, 80	79	80	
2	81	81	81	open_character_stream
3	83	83	83	open_character_stream
4	84	84	84	

5. get_token

```
94    String get_token(BufferedReader br)
95    {
96      int i=0,j;
97      int id=0;
98      int res = 0;
99      char ch = '\0';
100
101     StringBuilder sb = new StringBuilder();
102
103      try {
104          res = get_char(br);
105          if (res == -1) {
106              return null;
107          }
108          ch = (char)res;
109         while(ch==' '||ch=='\n' || ch == '\r')
110         {
111           res = get_char(br);
112           ch = (char)res;
113         }
114
115      if(res == -1)return null;
116      sb.append(ch);
117      if(is_spec_symbol(ch)==true)return sb.toString();
118      if(ch =='"')id=2;     /* prepare for string */
119      if(ch ==59)id=1;     /* prepare for comment */
120
121      res = get_char(br);
122      if (res == -1) {
123          unget_char(ch,br);
124          return sb.toString();
125      }
126      ch = (char)res;
127
128      while (is_token_end(id,res) == false)/* until meet
129      {
130          sb.append(ch);
131          br.mark(4);
132          res = get_char(br);
133          if (res == -1) {
134              break;
135          }
136          ch = (char)res;
137      }
139      if(res == -1)       /* if end
140         { unget_char(ch,br);
141           return sb.toString();
142         }
143
144      if(is_spec_symbol(ch)==true)
145         { unget_char(ch,br);
146           return sb.toString();
147         }
148      if(id==1)                   /*
149        {
150          if (ch == '"') {
151              sb.append(ch);
152          }
153          return sb.toString();
154        }
155      if(id==0 && ch==59)
156
157        { unget_char(ch,br);       /
158          return sb.toString();
159        }
160   } catch (IOException e) {
161       e.printStackTrace();
162   }
163
164      return sb.toString();
165   }
166
```

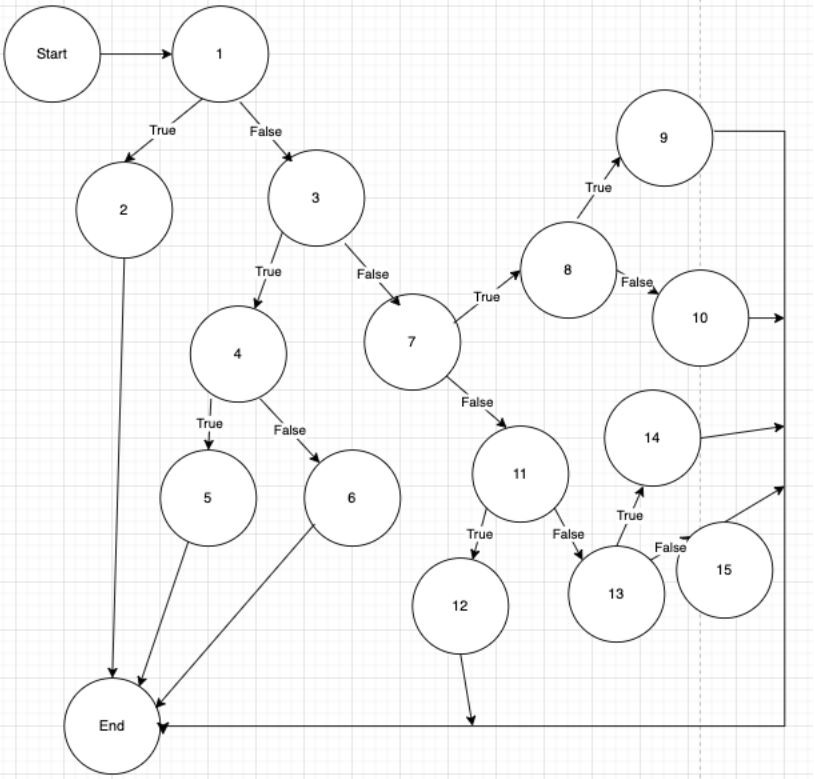
Block Number	Lines	Entry	Exit	Function calls
1	96, 97, 98, 99, 101, 104, 105	96	105	
2	106	106	106	
3	108	108	108	
4	109	109	109	
5	111, 112	111	112	
6	115a	115a	115a	
7	115b	115b	115b	
8	116, 117a	116	117a	
9	117b	117b	117b	
10	118a	118a	118a	
11	118b	118b	118b	
12	119a	119a	119a	
13	119b	119b	119b	
14	121, 122	121	122	
15	123, 124	123	124	
16	126	126	126	
17	128	128	128	
18	130, 131, 132, 133	130	133	
19	134	134	134	
20	136	136	136	
21	139	139	139	
22	140, 141	140	141	
23	144	144	144	
24	145, 146	145	146	
25	148	148	148	
26	150, 151, 153	150	153	
27	155	155	155	
28	157, 158	157	158	
29	164	164	164	



6. is_token_end

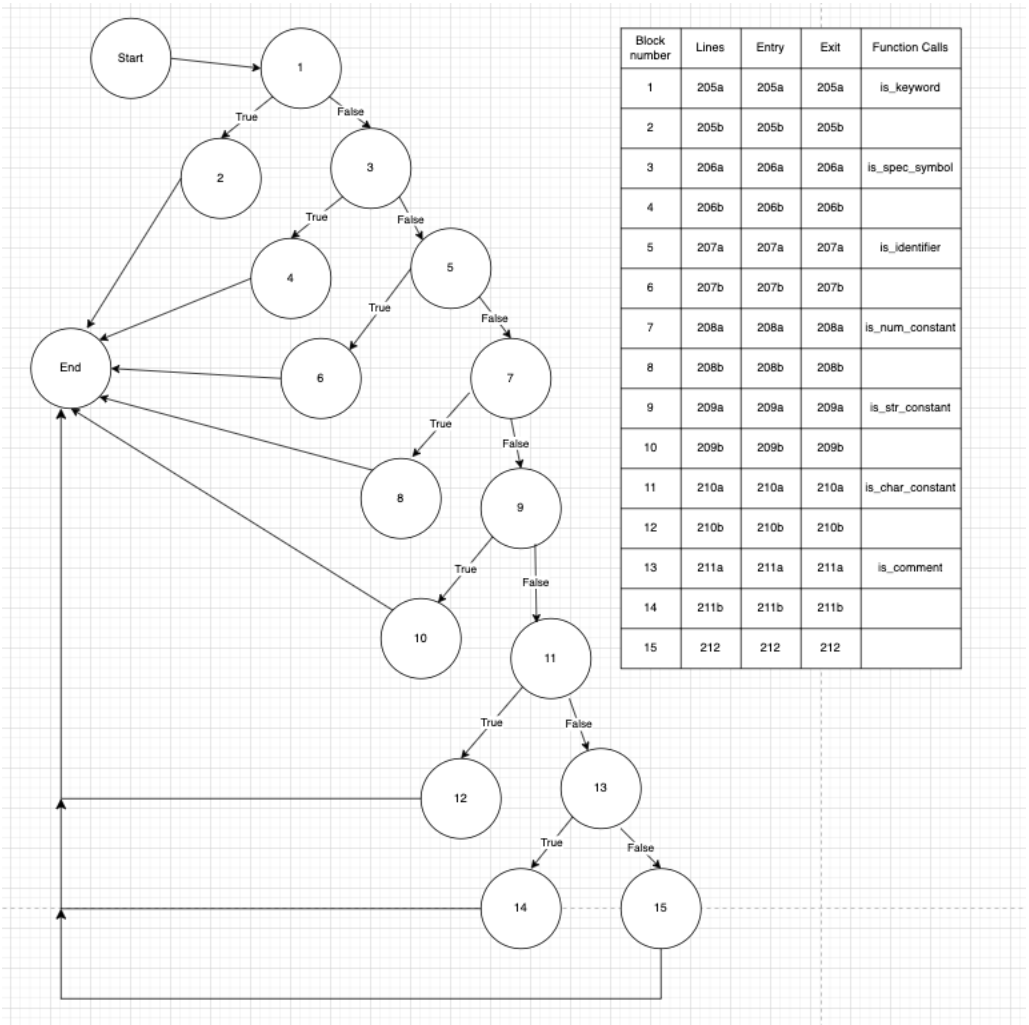
```
172    static boolean is_token_end(int str_com_id, int res)
173    {
174     if(res==-1)return(true); /* is eof token? */
175     char ch = (char)res;
176     if(str_com_id==1)          /* is string token */
177        { if(ch=='"' || ch=='\n' || ch == '\r' || ch=='\t')   /*
178             return true;
179          else
180             return false;
181        }
182
183     if(str_com_id==2)    /* is comment token */
184       { if(ch=='\n' || ch == '\r' || ch=='\t')     /* for comme
185            return true;
186          else
187            return false;
188       }
189
190     if(is_spec_symbol(ch)==true) return true; /* is special_sym
191     if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59) return true;
192                                  /* others until meet blank or
193     return false;               /* other case,return FALSE */
194    }
```

Block Number	Lines	Entry	Exit	Function Calls
1	174a	174a	174a	
2	174b	174b	174b	
3	175, 176	175	176	
4	177	177	177	
5	178	178	178	
6	180	180	180	
7	183	183	183	
8	184	184	184	
9	185	185	185	
10	187	187	187	
11	190a	190a	190a	is_spec_symbol
12	190b	190b	190b	is_spec_symbol
13	191a	191a	191a	
14	191b	191b	191b	
15	193	193	193	



7. token_type

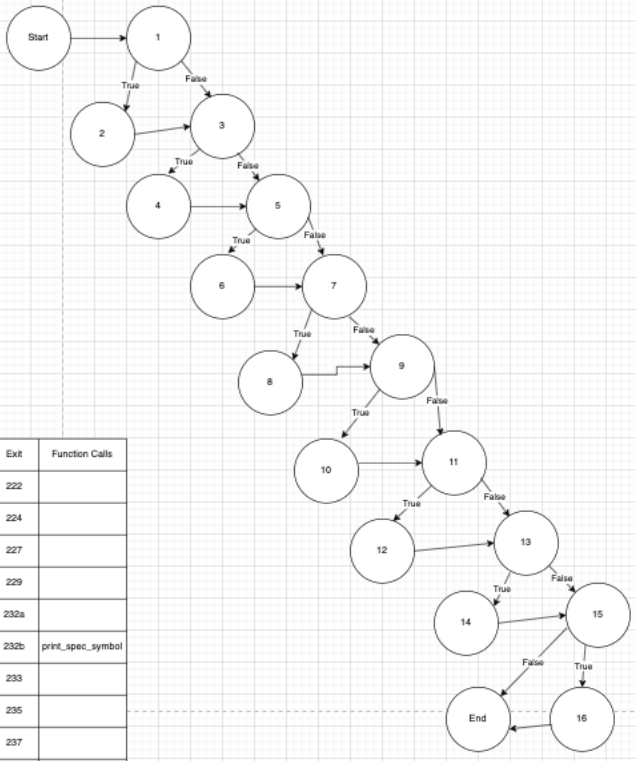
```
203    static int token_type(String tok)
204    {
205     if(is_keyword(tok))return(keyword);
206     if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
207     if(is_identifier(tok))return(identifier);
208     if(is_num_constant(tok))return(num_constant);
209     if(is_str_constant(tok))return(str_constant);
210     if(is_char_constant(tok))return(char_constant);
211     if(is_comment(tok))return(comment);
212     return(error);                    /* else look as erro
213    }
```



Block number	Lines	Entry	Exit	Function Calls
1	205a	205a	205a	is_keyword
2	205b	205b	205b	
3	206a	206a	206a	is_spec_symbol
4	206b	206b	206b	
5	207a	207a	207a	is_identifier
6	207b	207b	207b	
7	208a	208a	208a	is_num_constant
8	208b	208b	208b	
9	209a	209a	209a	is_str_constant
10	209b	209b	209b	
11	210a	210a	210a	is_char_constant
12	210b	210b	210b	
13	211a	211a	211a	is_comment
14	211b	211b	211b	
15	212	212	212	

8. print_token

```
219    void print_token(String tok)
220    { int type;
221      type=token_type(tok);
222     if(type==error)
223       {
224        System.out.print("error,\"" + tok + "\".\n");
225       }
226
227     if(type==keyword)
228       {
229       System.out.print("keyword,\"" + tok + "\".\n");
230       }
231
232     if(type==spec_symbol)print_spec_symbol(tok);
233     if(type==identifier)
234       {
235       System.out.print("identifier,\"" + tok + "\".\n");
236       }
237     if(type==num_constant)
238       {
239       System.out.print("numeric," + tok + ".\n");
240       }
241     if(type==str_constant)
242       {
243       System.out.print("string," + tok + ".\n");
244       }
245     if(type==char_constant)
246       {
247        System.out.print("character,\"" + tok.charAt(1) + "\".\n");
248       }
249     if(type==comment)
250       {
251        System.out.print("comment,\"" + tok + "\".\n");
252       }
253   }
```

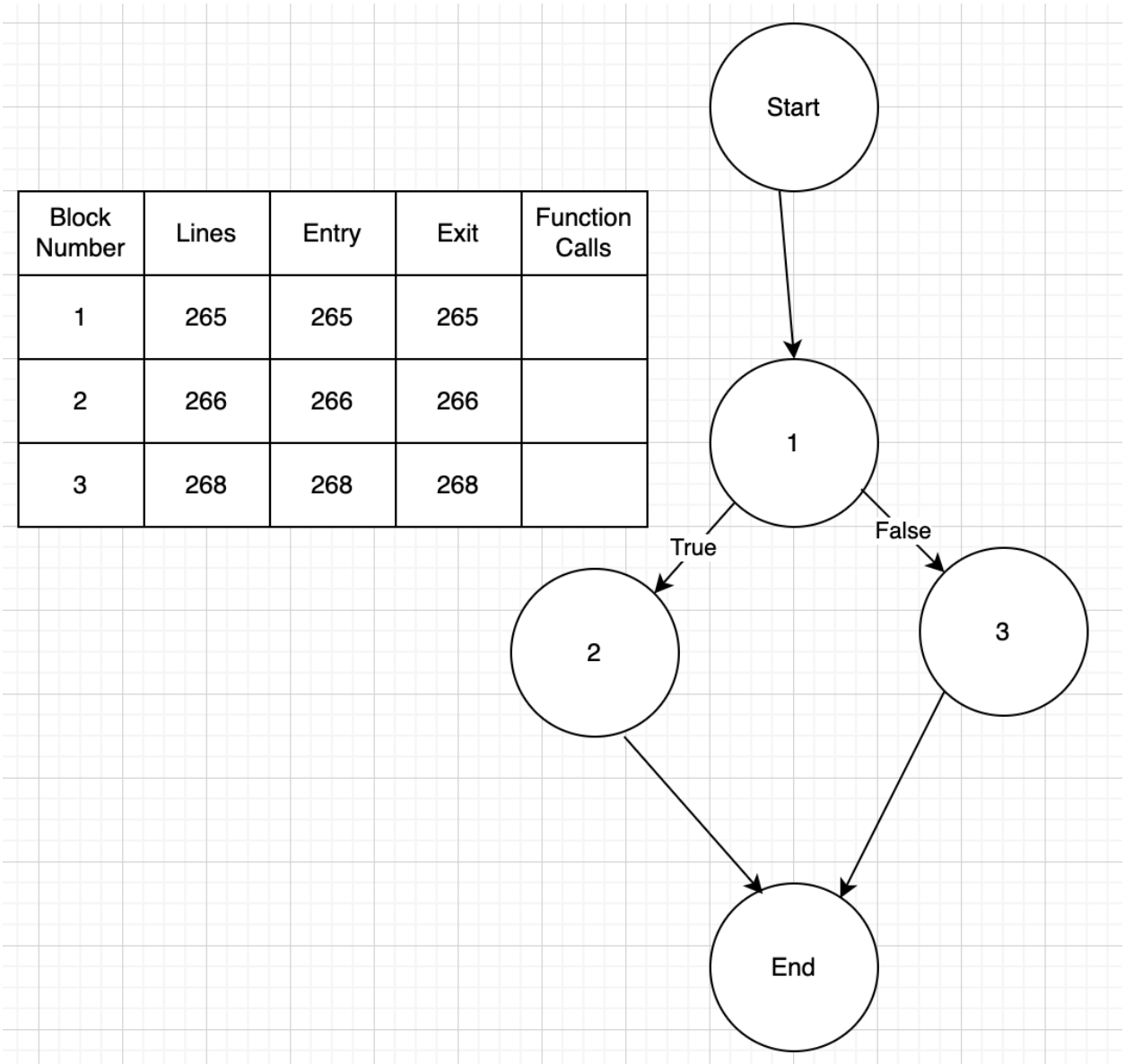


Block Number	Lines	Entry	Exit	Function Calls
1	220, 221, 222	220	222	
2	224	224	224	
3	227	227	227	
4	229	229	229	
5	232a	232a	232a	
6	232b	232b	232b	print_spec_symbol
7	233	233	233	
8	235	235	235	
9	237	237	237	
10	239	239	239	
11	241	241	241	
12	243	243	243	
13	245	245	245	
14	247	247	247	
15	249	249	249	
16	251	251	251	

9. is_comment

```
263    static boolean is_comment(String ident)
264    {
265      if( ident.charAt(0) ==59 )   /* the char is 59   */
266         return true;
267      else
268         return false;
269    }
270
271    /************************************/
272    /* NAME:    is_keyword           */
273    /* INPUT:   a token */
274    /* OUTPUT:      a BOOLEAN value      */
275    /************************************/
276    static boolean is_keyword(String str)
277    {
278     if (str.equals("and") || str.equals("or") || str.equals("if") ||
279            str.equals("xor")||str.equals("lambda")||str.equals("=>"))
280          return true;
281      else
282          return false;
283    }
```

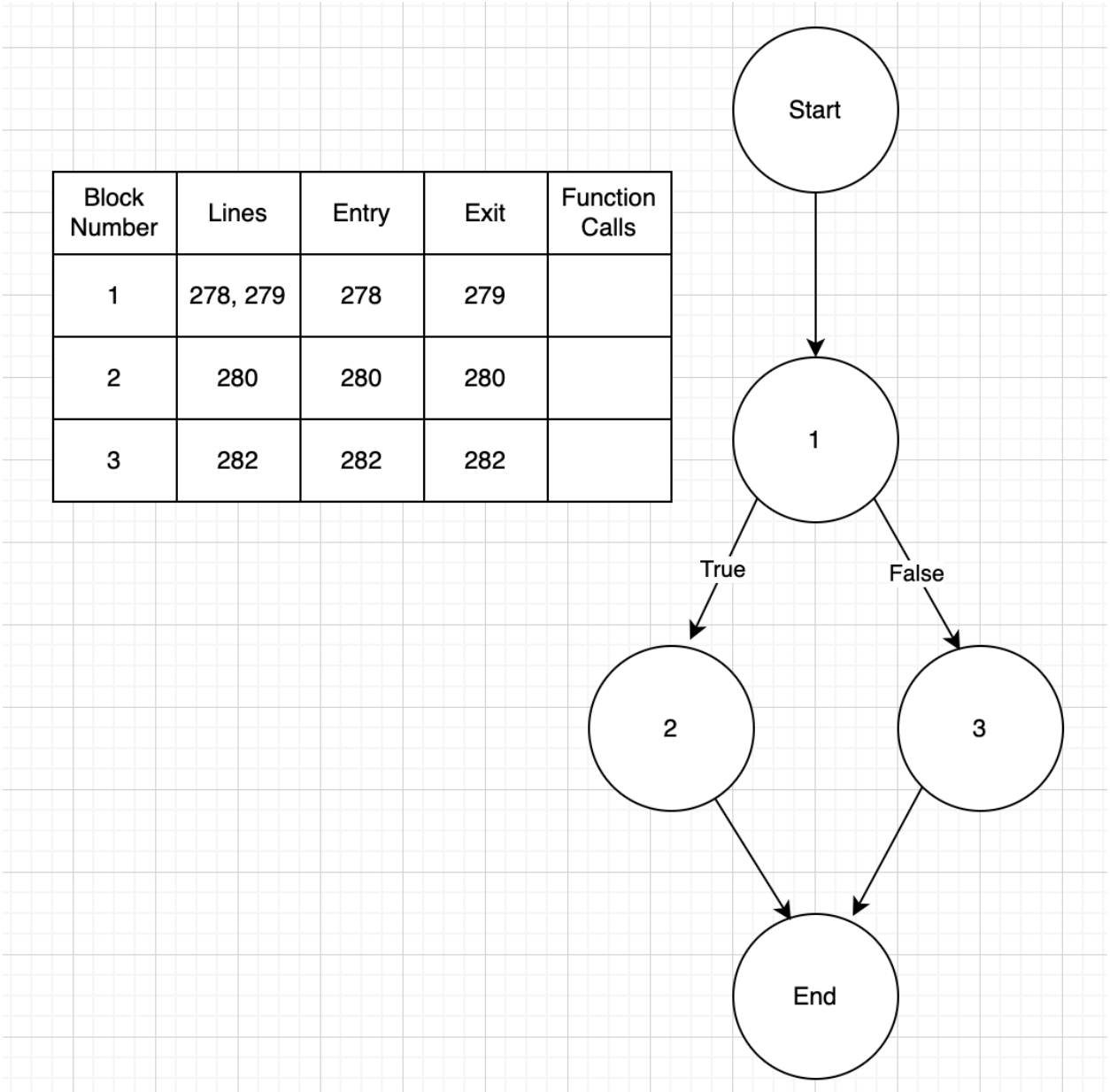
Block Number	Lines	Entry	Exit	Function Calls
1	265	265	265	
2	266	266	266	
3	268	268	268	



10. is_keyword

```
276    static boolean is_keyword(String str)
277    {
278     if (str.equals("and") || str.equals("or") || str.equals("if") ||
279             str.equals("xor")||str.equals("lambda")||str.equals("=>"))
280          return true;
281       else
282          return false;
283    }
284
```

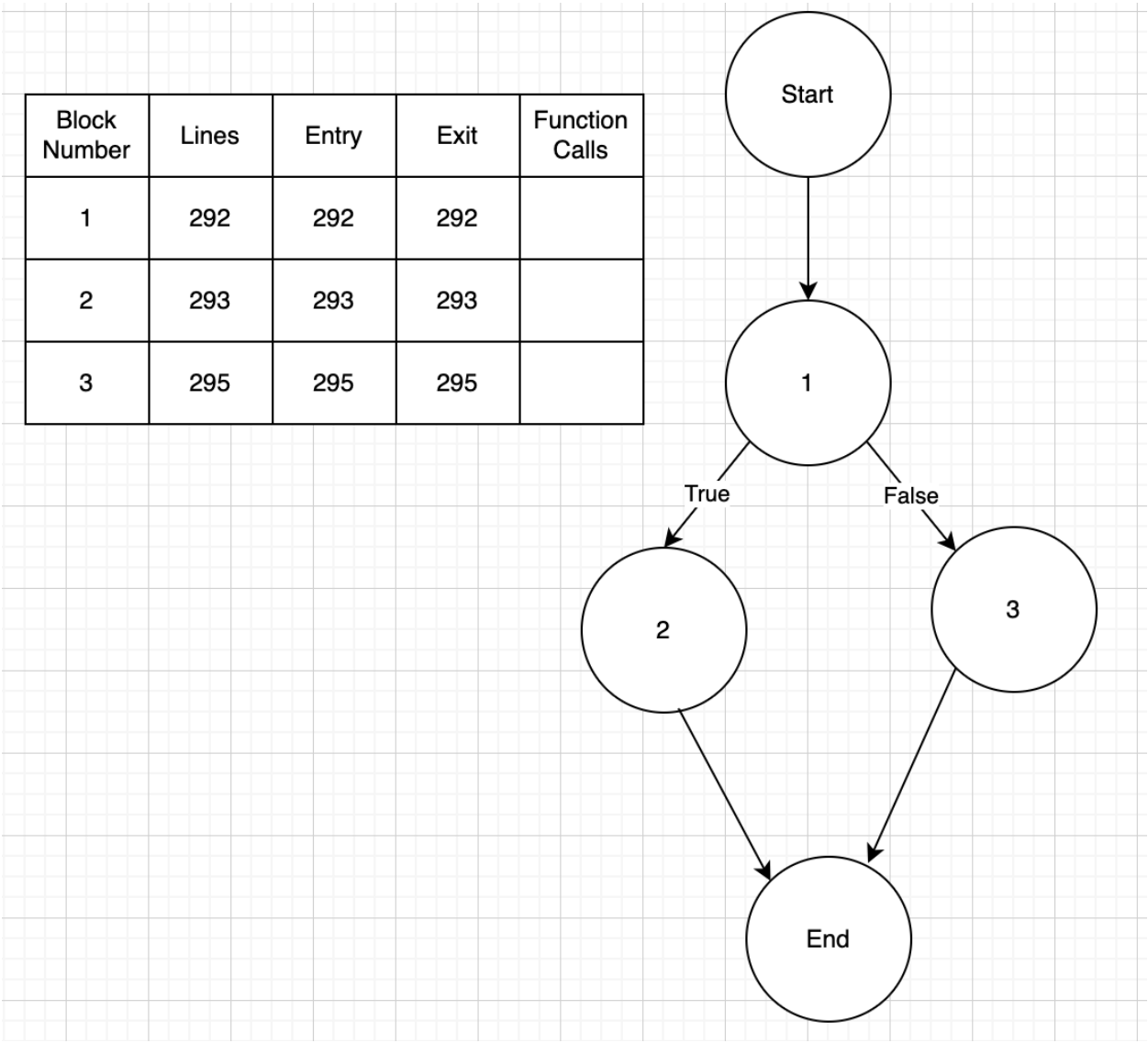
Block Number	Lines	Entry	Exit	Function Calls
1	278, 279	278	279	
2	280	280	280	
3	282	282	282	



11. is_char_constant

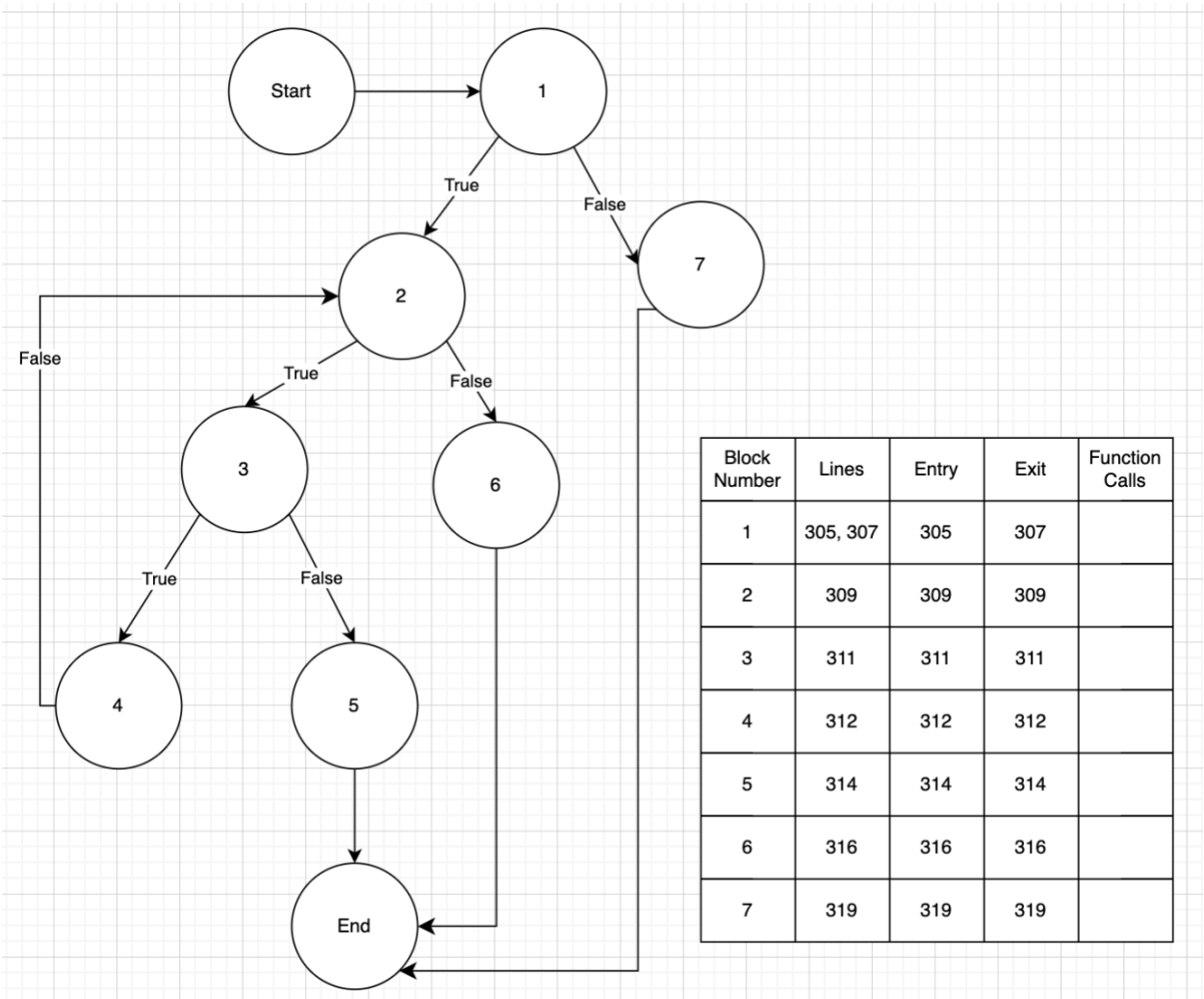
```
290    static boolean is_char_constant(String str)
291    {
292      if (str.length() > 2 || str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
293         return true;
294      else
295         return false;
296    }
```

Block Number	Lines	Entry	Exit	Function Calls
1	292	292	292	
2	293	293	293	
3	295	295	295	



12. is_num_constant

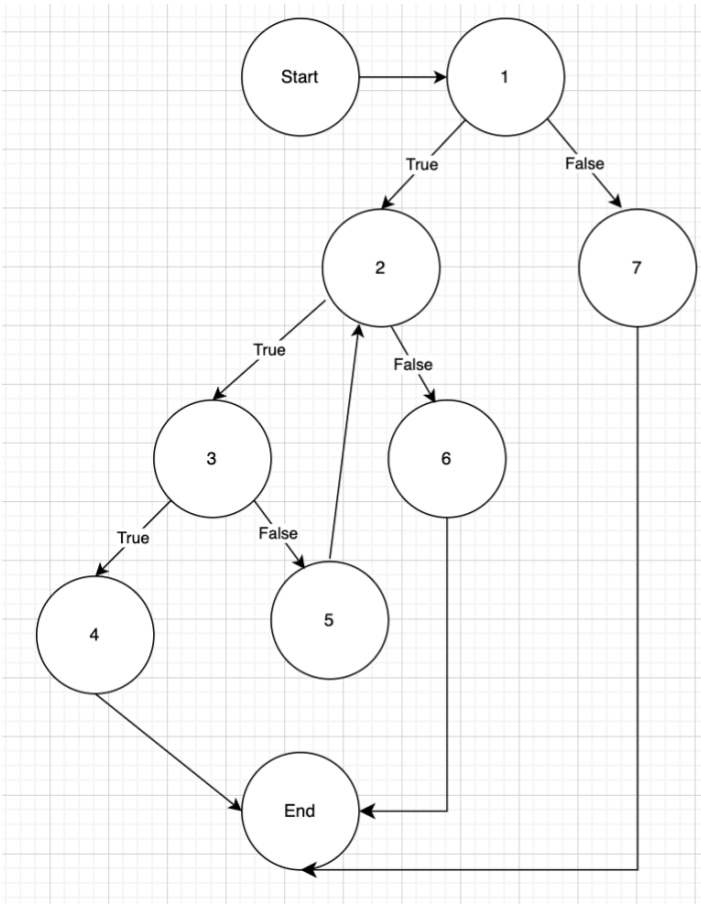
```
303    static boolean is_num_constant(String str)
304    {
305      int i=1;
306
307      if ( Character.isDigit(str.charAt(0)))
308        {
309        while ( i < str.length() && str.charAt(i) != '\0' )
310          {
311           if(Character.isDigit(str.charAt(i+1)))
312             i++;
313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



Block Number	Lines	Entry	Exit	Function Calls
1	305, 307	305	307	
2	309	309	309	
3	311	311	311	
4	312	312	312	
5	314	314	314	
6	316	316	316	
7	319	319	319	

13. is_str_constant

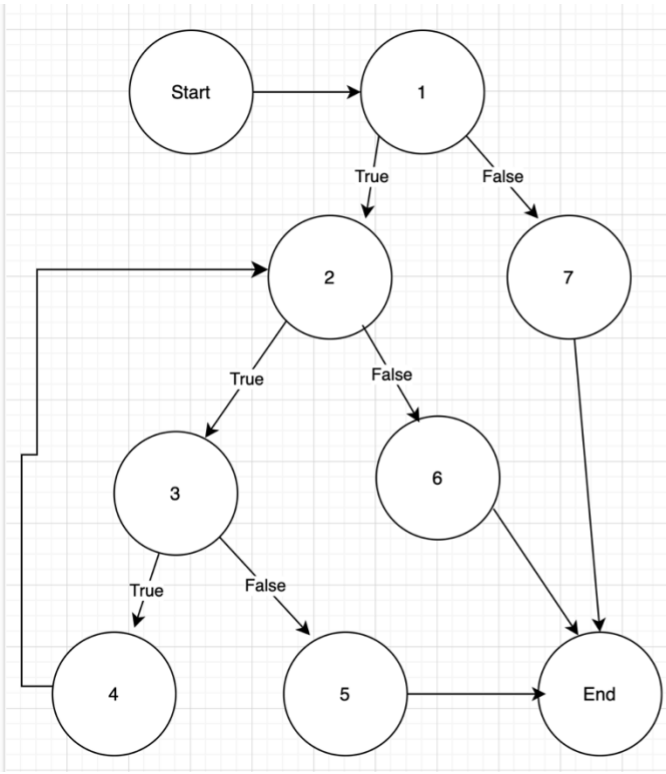
```
327    static boolean is_str_constant(String str)
328    {
329      int i=1;
330
331      if ( str.charAt(0) =='"')
332         { while (i < str.length() && str.charAt(i)!='\0')
333             { if(str.charAt(i)=='"' )
334                 return true;        /* meet the second '"
335           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



Block Number	Lines	Entry	Exit	Function Calls
1	329, 331	329	331	
2	332	332	332	
3	333	333	333	
4	334	334	334	
5	336	336	336	
6	338	338	338	
7	341	341	341	

14. is_identifier

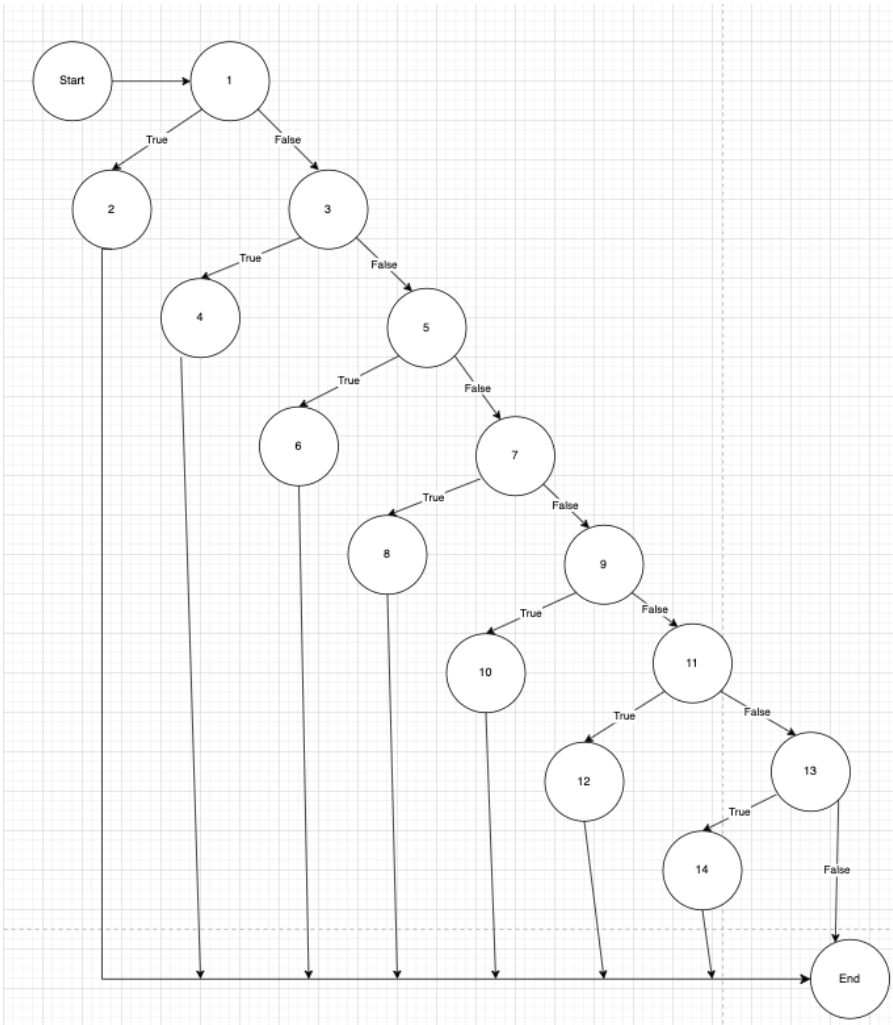
```
349    static boolean is_identifier(String str)
350    {
351      int i=1;
352
353      if ( Character.isLetter(str.charAt(0)) )
354         {
355            while(i < str.length() && str.charAt(i) !='\0' )    /* unti meet the end token s
356               {
357                if(Character.isLetter(str.charAt(i))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



Block Number	Lines	Entry	Exit	Function Calls
1	351, 353	351	353	
2	355	355	355	
3	357	357	357	
4	358	358	358	
5	360	360	360	
6	362	362	362	
7	365	365	365	

15. print_spec_symbol

```
376    static void print_spec_symbol(String str)
377    {
378        if      (str.equals(")"))
379        {
380
381                System.out.print("lparen.\n");
382                return;
383        }
384        if (str.equals(")"))
385        {
386
387                System.out.print("rparen.\n");
388                return;
389        }
390        if (str.equals("["))
391        {
392                System.out.print("lsquare.\n");
393                return;
394        }
395        if (str.equals("]"))
396        {
397
398                System.out.print("rsquare.\n");
399                return;
400        }
401        if (str.equals("'"))
402        {
403                System.out.print("quote.\n");
404                return;
405        }
406        if (str.equals("`"))
407        {
408
409                System.out.print("bquote.\n");
410                return;
411        }
412
413        if (str.equals(","))
414        {
415                System.out.print("comma.\n");
416                return;
417        }
418    }
```

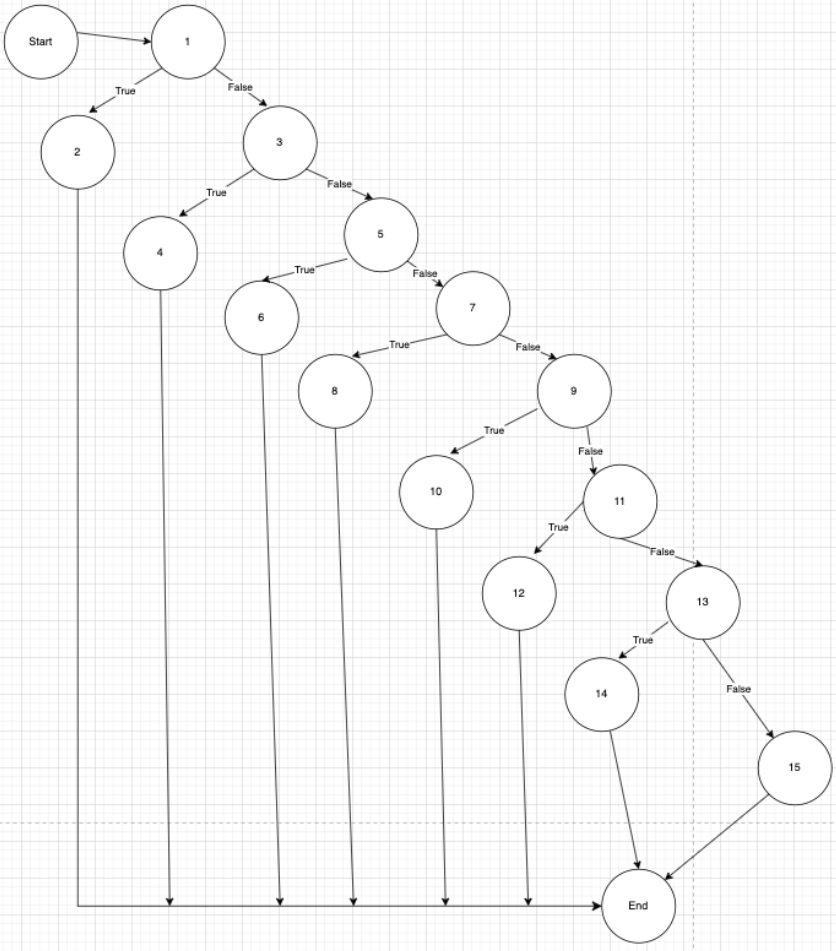


Block Number	Lines	Entry	Exit	Function Calls
1	378	378	378	
2	381, 382	381	382	
3	384	384	384	
4	387, 388	387	388	
5	390	390	390	
6	392, 393	392	393	
7	395	395	395	
8	398, 399	398	399	
9	401	401	401	
10	403, 404	403	404	
11	406	406	406	
12	409, 410	409	410	
13	413	413	413	
14	415, 416	415	416	

16. is_spec_symbol

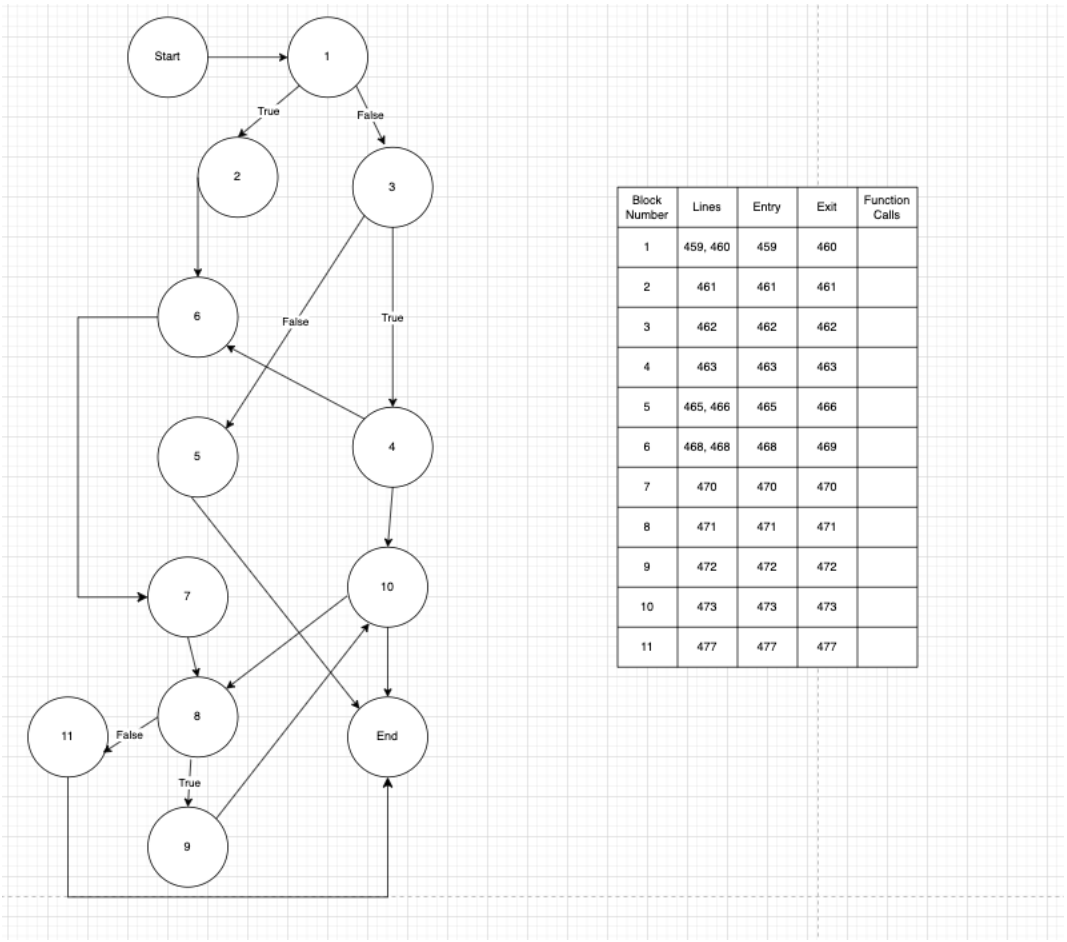
```
425    static boolean is_spec_symbol(char c)
426    {
427        if (c == '(')
428        {
429            return true;
430        }
431        if (c == ')')
432        {
433            return true;
434        }
435        if (c == '[')
436        {
437            return true;
438        }
439  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

Block Number	Lines	Entry	Exit	Function Calls
1	427	427	427	
2	429	429	429	
3	431	431	431	
4	433	433	433	
5	435	435	435	
6	437	437	437	
7	439	439	439	
8	441	441	441	
9	443	443	443	
10	445	445	445	
11	447	447	447	
12	449	449	449	
13	451	451	451	
14	453	453	453	
15	455	455	455	



17. main

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



Block Number	Lines	Entry	Exit	Function Calls
1	459, 460	459	460	
2	461	461	461	
3	462	462	462	
4	463	463	463	
5	465, 466	465	466	
6	468, 468	468	469	
7	470	470	470	
8	471	471	471	
9	472	472	472	
10	473	473	473	
11	477	477	477	