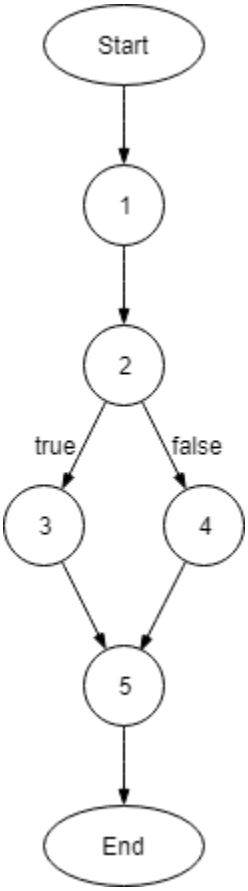
Method 1: open_character_stream

Block	Lines	Entry	Exit
1	27, 28	27	28
2	29	29	29
3	30	30	30
4	32 to 35	32	35
5	41	41	41

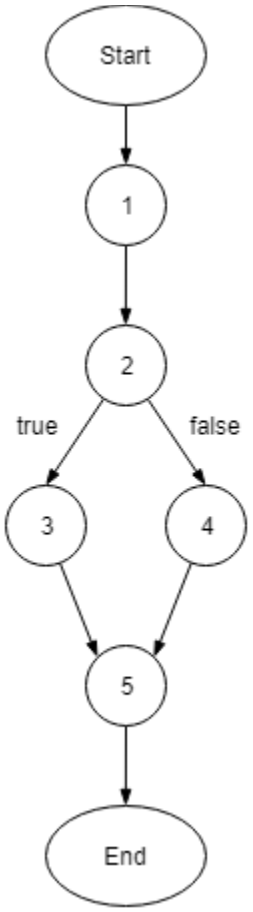


```
27      BufferedReader open_character_stream(String fname) {
28          BufferedReader br = null;
29          if (fname == null) {
30              br = new BufferedReader(new InputStreamReader(System.in));
31          } else {
32              try {
33                  FileReader fr = new FileReader(fname);
34                  br = new BufferedReader(fr);
35              } catch (FileNotFoundException e) {
36                  System.out.print("The file " + fname +" doesn't exists\n");
37                  e.printStackTrace();
38              }
39          }
40
41          return br;
42      }
```

Method 2: get_char → Skipped

Method 3: unget_char → Skipped

Method 4: open_token_stream



Block	Lines	Entry	Exit
1	82, 83, 84	82	84
2	85	85	85
3	86	86	86
4	88	88	88
5	89	89	89

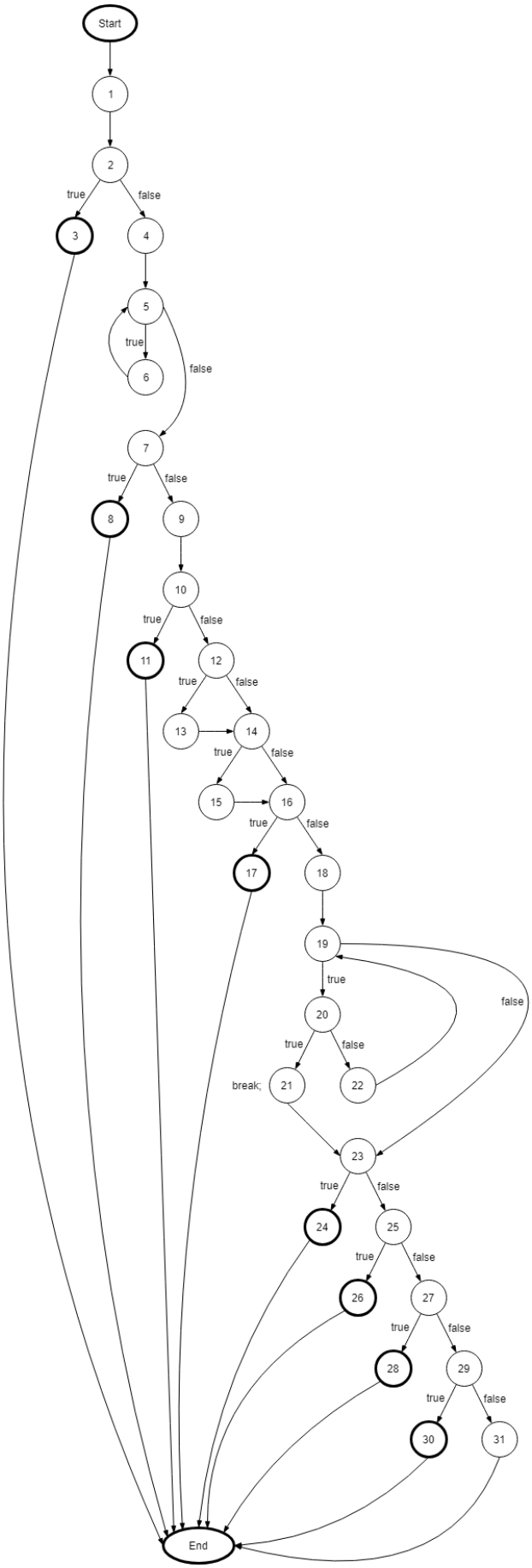
```
82      BufferedReader open_token_stream(String fname)
83      {
84          BufferedReader br;
85       if(fname.equals(null))
86          br=open_character_stream(null);
87       else
88          br=open_character_stream(fname);
89       return br;
90      }
```

Method 5: get_token

Block	Lines	Entry	Exit
1	101 to 104, 106, 109	101	109
2	110	110	110
3	111	111	111
4	113	113	113
5	114	114	114
6	116, 117	116	117
7	120	120	120
8	120	120	120
9	121	121	121
10	122	122	122
11	122	122	122
12	123	123	132
13	123	123	123
14	124	124	124
15	124	124	124
16	126, 127	126	127
17	128, 129	128	129
18	131	131	131
19	133	133	133
20	135 to 138	135	138
21	139	139	139
22	141	141	141
23	144	144	144
24	145, 146	145	146
25	149	149	149
26	150, 151	150	151
27	153	153	153
28	155, 156	155	156
29	158	158	158
30	160, 161	160	161
31	167	167	167

```
99      String get_token(BufferedReader br)
100     {
101       int i=0,j;
102       int id=0;
103       int res = 0;
104       char ch = '\0';
105
106       StringBuilder sb = new StringBuilder();
107
108        try {
109            res = get_char(br);
110            if (res == -1) {
111                return null;
112            }
113            ch = (char)res;
114         while(ch==' '||ch=='\n' || ch == '\r')       /* strip all
115           {
116             res = get_char(br);
117             ch = (char)res;
118           }
119
120        if(res == -1)return null;
121        sb.append(ch);
122        if(is_spec_symbol(ch)==true)return sb.toString();
123        if(ch =='"')id=0;     /* prepare for string */
124        if(ch ==59)id=1;     /* prepare for comment */
125
126        res = get_char(br);
127        if (res == -1) {
128            unget_char(ch,br);
129            return sb.toString();
130        }
131        ch = (char)res;
```

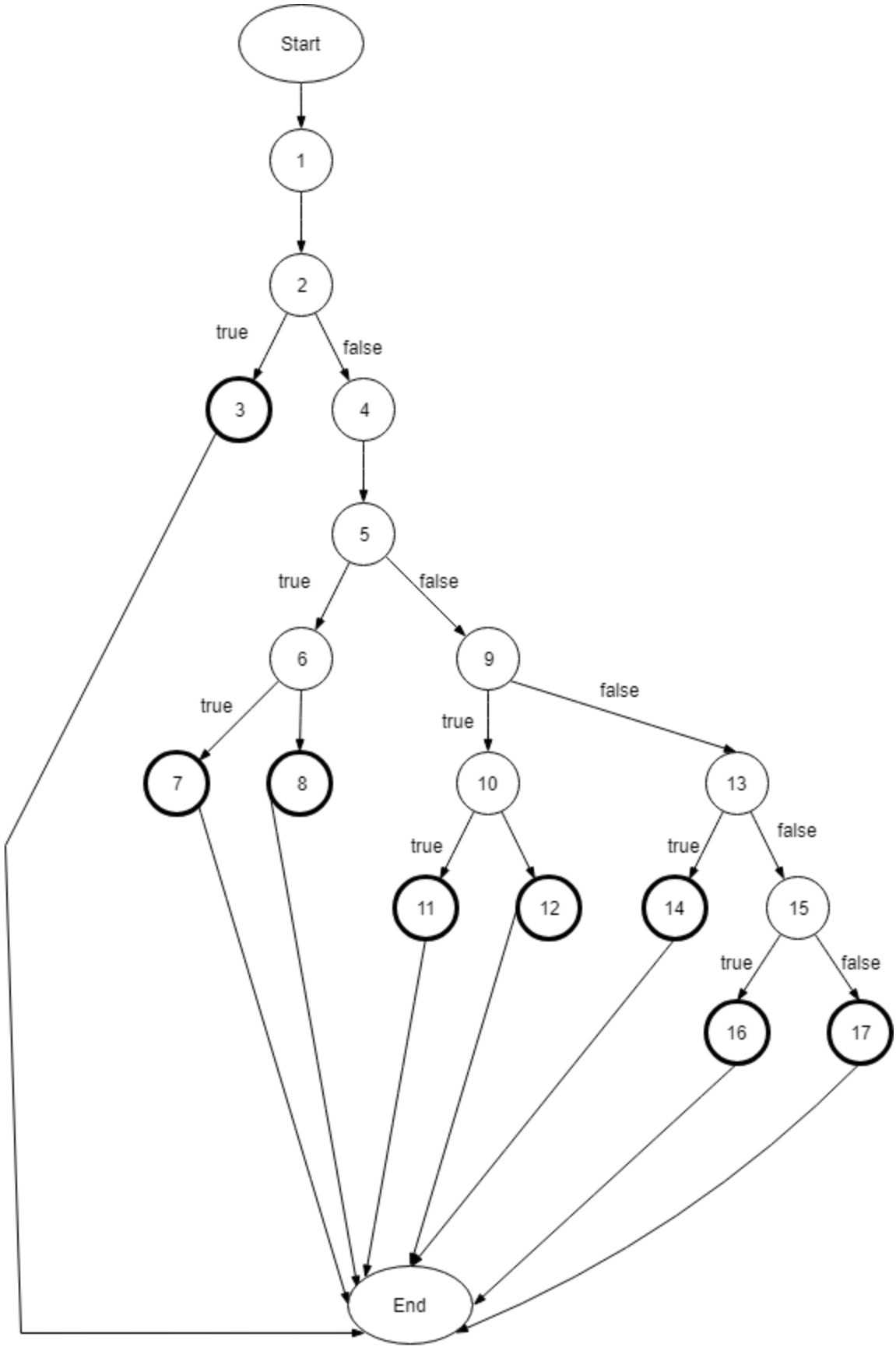
```
132
133         while (is_token_end(id,res) == false)/* until meet the end character */
134         {
135             sb.append(ch);
136             br.mark(4);
137             res = get_char(br);
138             if (res == -1) {
139                 break;
140             }
141             ch = (char)res;
142         }
143
144         if(res == -1)       /* if end character is eof token    */
145           { unget_char(ch,br);        /* then put back eof on token_stream */
146             return sb.toString();
147           }
148
149         if(is_spec_symbol(ch)==true)     /* if end character is special_symbol */
150           { unget_char(ch,br);        /* then put back this character       */
151             return sb.toString();
152           }
153         if(id==1)                   /* if end character is " and is string */
154           {
155             sb.append(ch);
156             return sb.toString();
157           }
158         if(id==0 && ch==59)
159                                          /* when not in string or comment,meet ";" */
160           { unget_char(ch,br);       /* then put back this character         */
161             return sb.toString();
162           }
163       } catch (IOException e) {
164           e.printStackTrace();
165       }
166
167         return sb.toString();                   /* return nomal case token
168       }
```


Start
1
2
true
false
3
4
5
true
6
false
7
true
false
8
9
10
true
false
11
12
true
false
13
14
true
false
15
16
true
false
17
18
19
true
false
20
true
false
break;
21
22
23
true
false
24
25
true
false
26
27
true
false
28
29
true
false
30
31
End

Method 6 : is_token_end

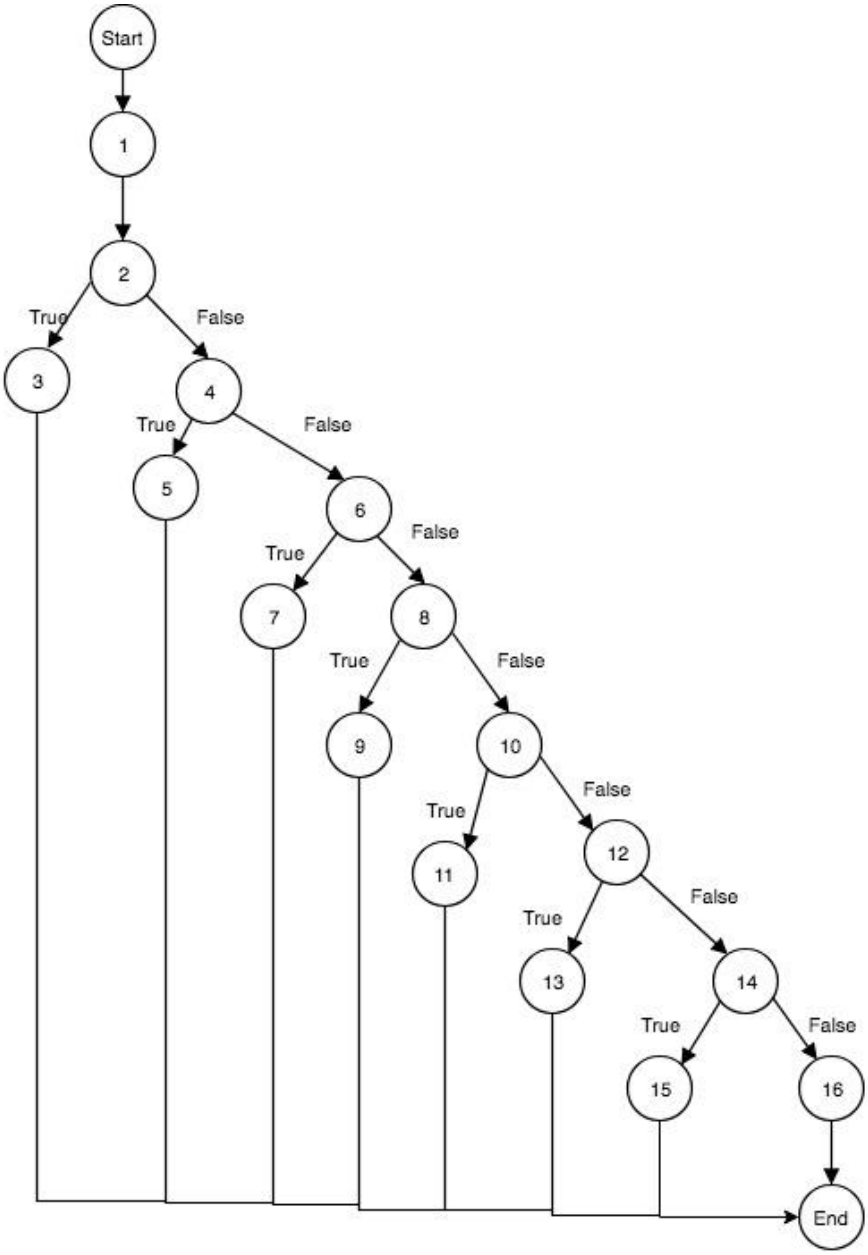
Block	Lines	Entry	Exit
1	175	175	175
2	177	177	177
3	177	177	177
4	178	178	178
5	179	179	179
6	180	180	180
7	181	181	181
8	183	183	183
9	186	186	186
10	187	187	187
11	188	188	188
12	190	190	190
13	193	193	193
14	193	193	193
15	194	194	194
16	194	194	194
17	196	196	196

```
170     /*********************************************************/
171     /* NAME:    is_token_end                                 */
172     /* INPUT:       a character,a token status               */
173     /* OUTPUT:  a BOOLEAN value                              */
174     /*********************************************************/
175     static boolean is_token_end(int str_com_id, int res)
176     {
177      if(res==1)return(true); /* is eof token? */
178      char ch = (char)res;
179      if(str_com_id==1)          /* is string token */
180         { if(ch=='"' | ch=='\n' || ch == '\r')   /* for string until meet another " */
181              return true;
182           else
183              return false;
184         }
185
186      if(str_com_id==2)    /* is comment token */
187        { if(ch=='\n' || ch == '\r' || ch==' ')     /* for comment until meet end of line */
188             return true;
189           else
190             return false;
191        }
192
193      if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
194      if(ch ==' ' || ch=='\r' || ch==59) return true;
195
196      return false;               /* other case,return FALSE */
197     }
```


Start
1
2
true
false
3
4
5
true
false
6
9
true
false
true
7
8
10
13
true
false
true
11
12
14
15
true
false
16
17
End

Method 7: token_type

```
206     static int token_type(String tok)
207     {
208      if(is_keyword(tok))return(keyword);
209      if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
210      if(is_identifier(tok))return(identifier);
211      if(is_num_constant(tok))return(num_constant);
212      if(is_str_constant(tok))return(str_constant);
213      if(is_char_constant(tok))return(char_constant);
214      if(is_comment(tok))return(comment);
215      return(error);                    /* else look as error token */
216     }
```



Block	Line	Entry	Exit
1	206	206	206
2	208	208	208
3	208	208	208
4	209	209	209
5	209	209	209

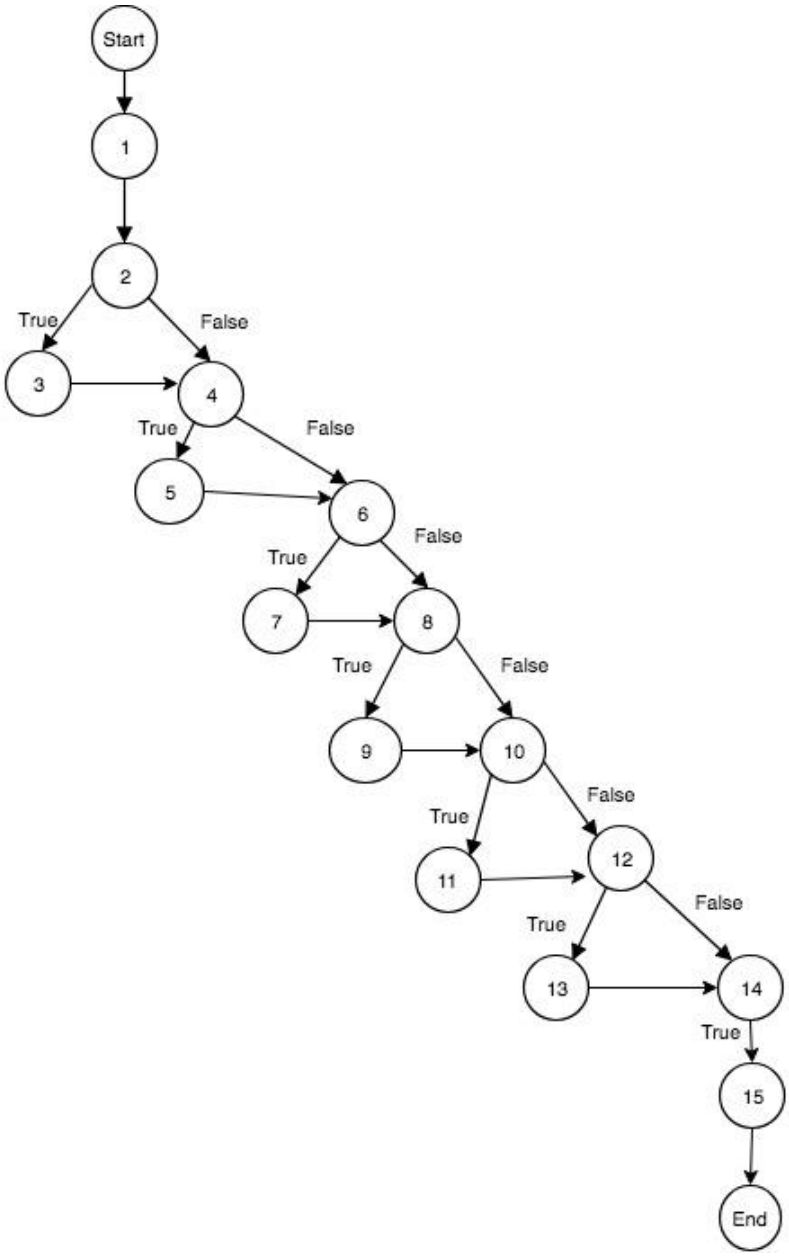
6	210	210	210
7	210	210	210
8	211	211	211
9	211	211	211
10	212	212	212
11	212	212	212
12	213	213	213
13	213	213	213
14	214	214	214
15	214	214	214
16	215	215	215

Method 8: print_token

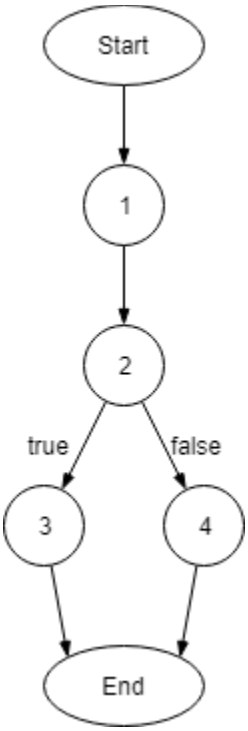
```
222       void print_token(String tok)
223       { int type;
224         type=token_type(tok);
225        if(type==error)
226          {
227           System.out.print("error,\"" + tok + "\".\n");
228          }
229
230        if(type==keyword)
231          {
232          System.out.print("keyword,\""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

Block	Line	Entry	Exit
1	222,223,224	223	224
2	225	225	225

3	227	227	227
4	230	230	230
5	232	232	232
6	235	235	235
7	235	235	235
8	236	236	236
9	238	238	238
10	240	240	240
11	242	242	242
12	244	244	244
13	246	246	246
14	248	248	248
15	250	250	250


Start
1
2
True
False
3
4
True
False
5
6
True
False
7
8
True
False
9
10
True
False
11
12
True
False
13
14
True
15
End

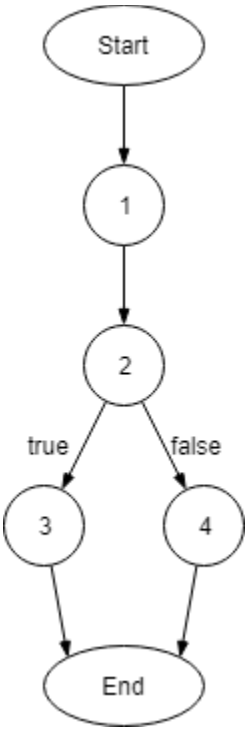
Method 9: is_comment



Block	Lines	Entry	Exit
1	263	263	263
2	265	265	265
3	266	266	266
4	268	268	268

```
258     /*************************************/
259     /* NAME:     is_comment              */
260     /* INPUT:    a token */
261     /* OUTPUT:       a BOOLEAN value       */
262     /*************************************/
263     static boolean is_comment(String ident)
264     {
265       if( ident.charAt(0) =='/' )    /* the char is 59    */
266          return true;
267       else
268          return false;
269     }
270
```

Method 10: is_keyword

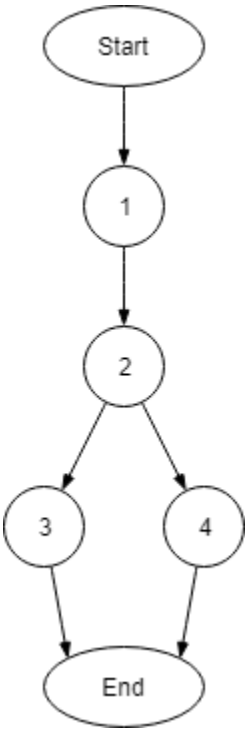


Block	Lines	Entry	Exit
1	276	276	276
2	278, 279	278, 279	278, 279
3	280	280	280
4	282	282	282

```
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
```

Method 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     }
```

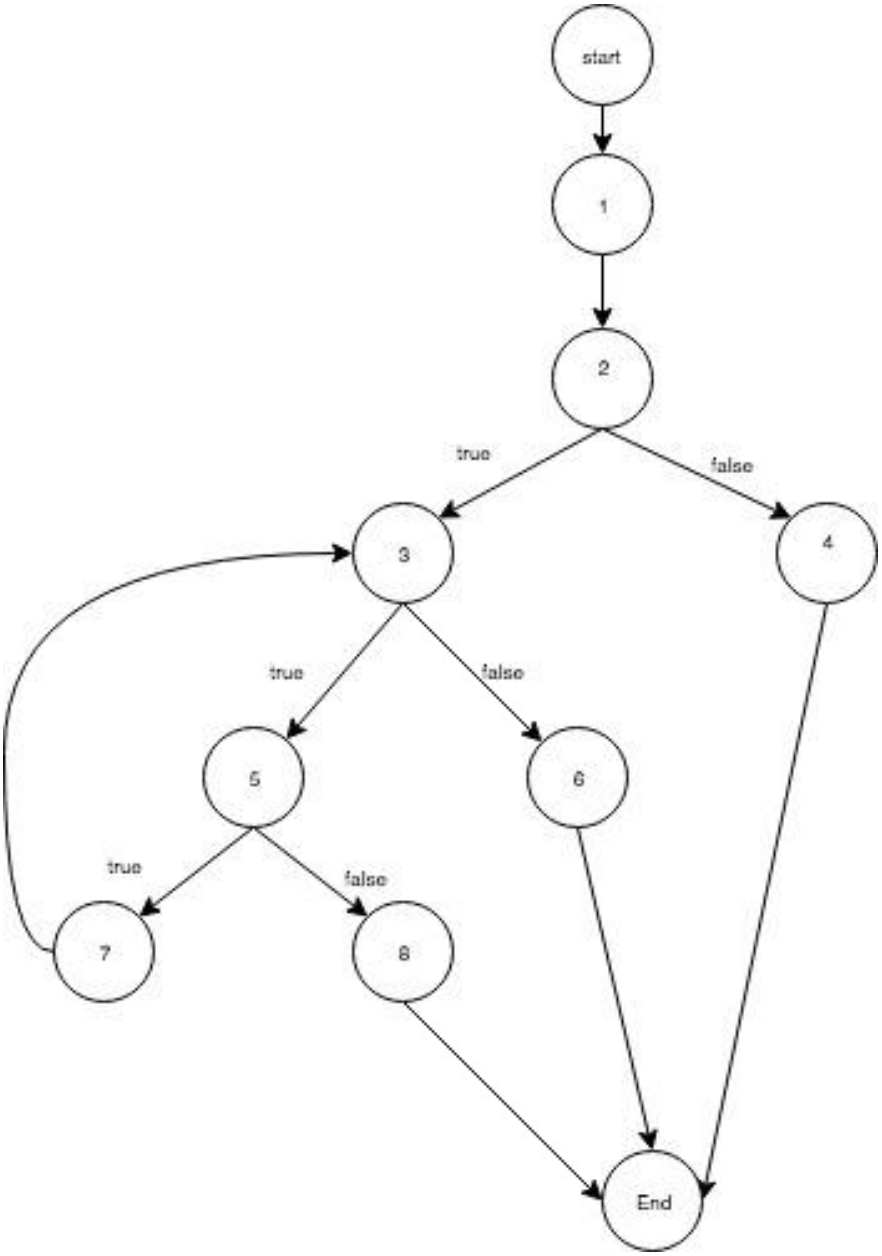


Block	Line	Entry	Exit
1	290	290	290
2	292	292	292
3	293	293	293
4	295	295	295

Method 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' )   /* until meet token end sign */
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     }
321
```

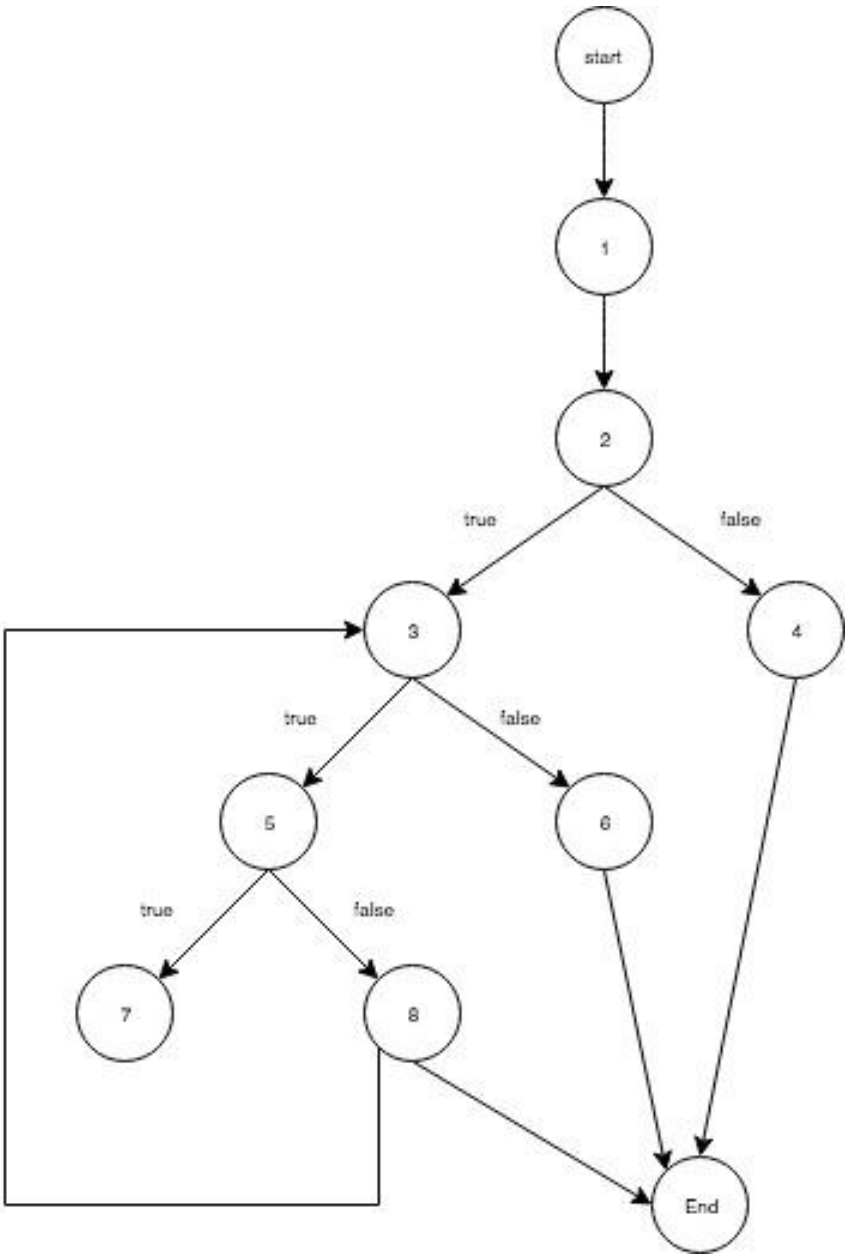
Block	Line	Entry	Exit
1	303,304,305	303	305
2	307	307	307
3	309	309	309
4	319	319	319
5	311	311	311
6	316	316	316
7	312	312	312
8	314	314	314


start
1
2
true
false
3
4
true
false
5
6
true
false
7
8
End

Method 13: is_str_constant

```
322     /************************************/
323     /* NAME:  is_str_constant      */
324     /* INPUT:   a token */
325     /* OUTPUT:      a BOOLEAN value      */
326     /************************************/
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(0)!='\0')  /* until meet the token end sign */
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
```

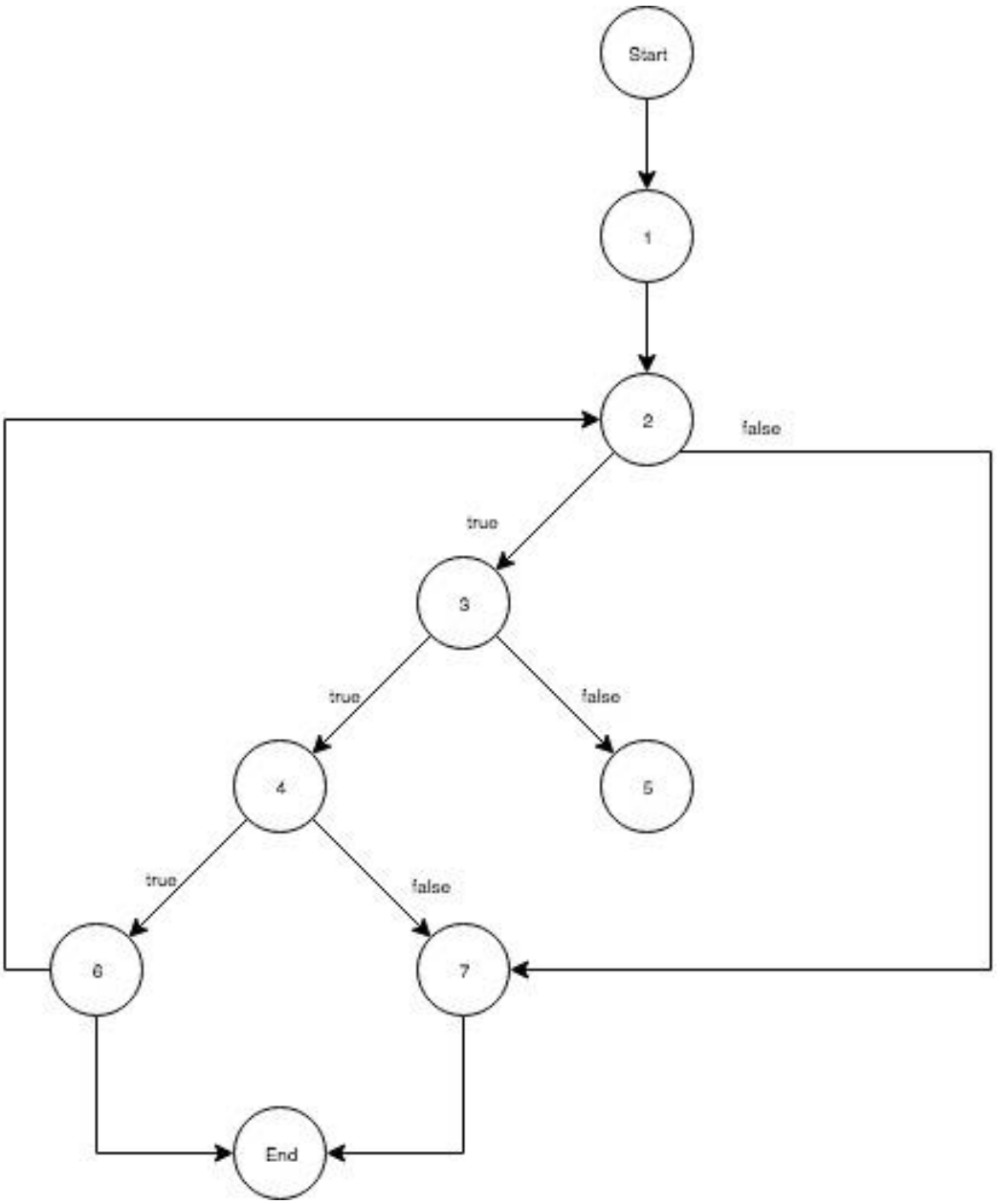
Block	Line	Entry	Exit
1	327,328,329	327	329
2	331	331	331
3	332	332	332
4	341	341	341
5	333	333	333
6	338	338	338
7	334	334	334
8	336	336	336


start
1
2
true
false
3
4
true
false
5
6
true
false
7
8
End

Method 14: is_identifier

```
344     /************************************/
345     /* NAME:  is_identifier         */
346     /* INPUT:   a token */
347     /* OUTPUT:      a BOOLEAN value      */
348     /************************************/
349     static boolean is_identifier(String str)
350     {
351       int i=0;
352
353       if ( Character.isLetter(str.charAt(0)) )
354          {
355             while(i < str.length() && str.charAt(i) !='\0' )   /* unti meet the end token sign */
                  {
                   if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
                      i++;
                   else
                      return false;
                  }      /* end WHILE */
          return true;
          }
       else
          return false;
     }
```

Block	Line	Entry	Exit
1	349,350,351	349	351
2	353	353	353
3	355	355	355
4	357	357	357
5	358	358	358
6	360,365	360,365	360,365
7	362	362	362


Start
1
2
false
true
3
true
false
4
5
true
false
6
7
End

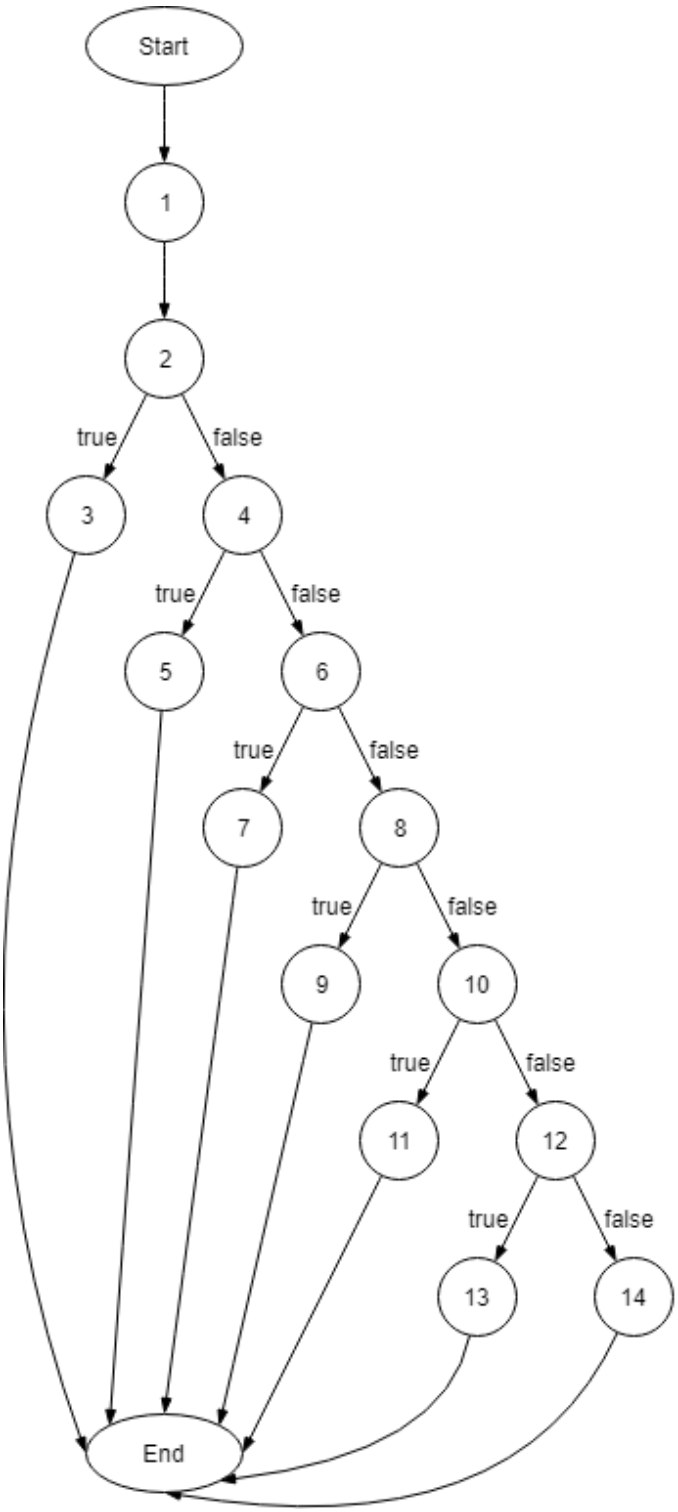
Method 15 : unget_error → Skipped

Method 16: print_spec_symbol

```
384     static void print_spec_symbol(String str)
385     {
386         if      (str.equals("("))
387         {
388 
389                  System.out.print("lparen.\n");
390                  return;
391         }
392         if (str.equals(")"))
393         {
394 
395                  System.out.print("rparen.\n");
396                  return;
397         }
398         if (str.equals("["))
399         {
400                  System.out.print("lsquare.\n");
401                  return;
402         }
403         if (str.equals("]"))
404         {
405 
406                  System.out.print("rsquare.\n");
407                  return;
408         }
409         if (str.equals("'"))
410         {
411                  System.out.print("quote.\n");
412                  return;
413         }
414         if (str.equals("`"))
415         {
416 
417                  System.out.print("bquote.\n");
418                  return;
419         }
420 
421         System.out.print("comma.\n");
422     }
```

Block	Line	Entry	Exit
1	384	384	384
2	386	386	386

3	389,390	389	390
4	392	392	392
5	395,396	395	396
6	398	398	398
7	400,401	400	401
8	403	403	403
9	406,407	406	407
10	409	409	409
11	411,412	411	412
12	414	414	414
13	417,418	417	418
14	421	421	421

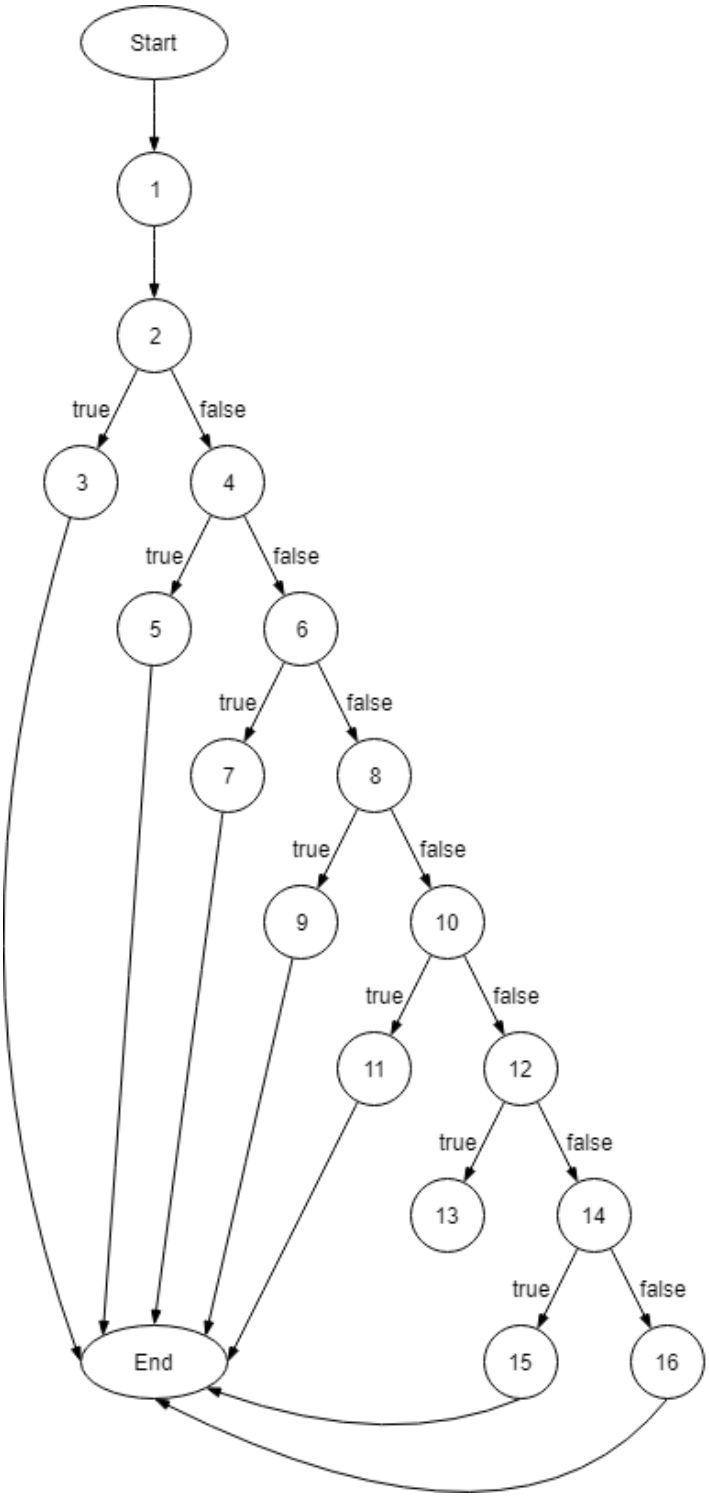

Start
1
2
true
false
3
4
true
false
5
6
true
false
7
8
true
false
9
10
true
false
11
12
true
false
13
14
End

Method 17: is_spec_symbol

```
429     static boolean is_spec_symbol(char c)
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         if (c == ', ')
456         {
457             return true;
458         }
459         return false;     /* others return FALSE */
460     }
```

Block	Line	Entry	Exit
1	429	429	429

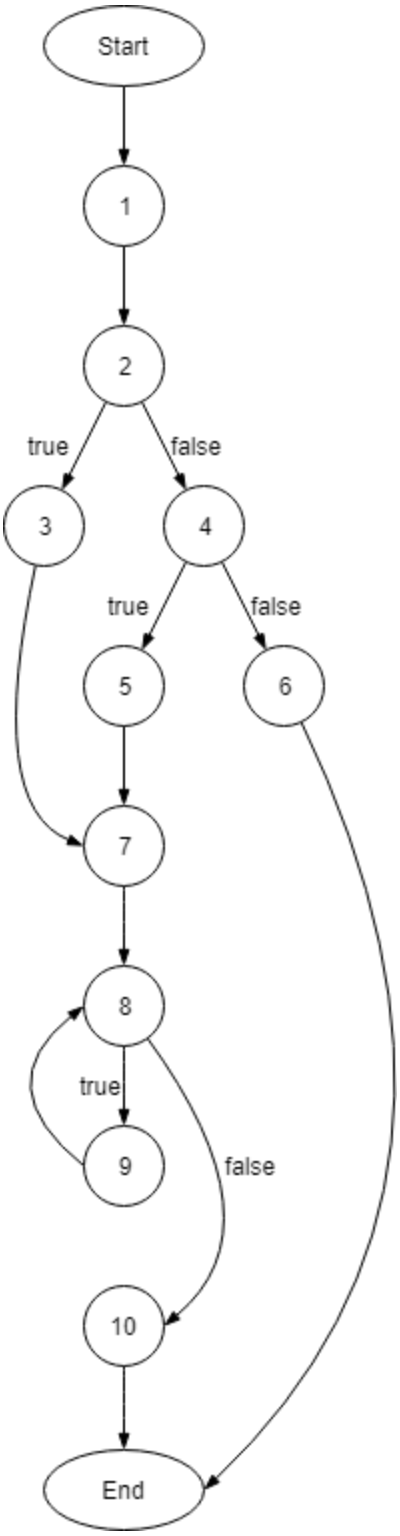
2	431	431	431
3	435	435	435
4	439	439	439
5	443	443	443
6	447	447	447
7	451	451	451
8	455	455	455
9	459	459	459
10	433,437,441,445,449,453,457	433,437,441,445,449,453,457	433,437,441,445,449,453,457


Start
1
2
true
false
3
4
true
false
5
6
true
false
7
8
true
false
9
10
true
false
11
12
true
false
13
14
true
false
15
16
End

Method 18: Main Method

Block	Lines	Entry	Exit
1	463	463	463
2	464	464	464
3	465	465	465
4	466	466	466
5	467	467	467
6	469, 470	469	470
7	472 to 474	472	474
8	475	475	475
9	476, 477	476	477
10	480	480	480

```
462     public static void main(String[] args) throws IOException {
463         String fname = null;
464         if (args.length == 0) { /* if not given filename,take as '"' */
465             fname = new String();
466         } else if (args.length == 1) {
467             fname = args[1];
468         } else {
469             System.out.print("Error!,please give the token stream\n");
470             System.exit(0);
471         }
472         Printtokens2 t = new Printtokens2();
473         BufferedReader br = t.open_token_stream(fname); /* open token stream */
474         String tok = t.get_token(br);
475         while (tok != "") { /* take one token each time until eof */
476             t.print_token(tok);
477             tok = t.get_token(br);
478         }
479
480         System.exit(0);
481     }
482 }
```


Start
1
2
true
false
3
4
true
false
5
6
7
8
true
9
false
10
End