Software Testing Project

CFGs that also contains the source code of each method, and the basic block table that identifies each basic block are given below:

1.open_character_stream method:

```
20  /*********************************************/
21  /* NMAE:  open_character_stream          */
22  /* INPUT:       a filename                   */
23  /* OUTPUT:      a BufferedReader */
24  /* DESCRIPTION: when not given a filename,     */
25  /*              open stdin,otherwise open      */
26  /*              the existed file               */
27  /*********************************************/
28  BufferedReader open_character_stream(String fname) {
29    BufferedReader br = null;
30    if (fname == null) {
31      br = new BufferedReader(new InputStreamReader(System.in));
32    } else {
33      try {
34        FileReader fr = new FileReader(fname);
35        br = new BufferedReader(fr);
36      } catch (FileNotFoundException e) {
37        System.out.print("The file " + fname +" doesn't exists\n");
38        e.printStackTrace();
39      }
40    }
41
42    return null;
43  }
```

Block	Lines	Entry	Exit
1	29,30	29	30
2	31	31	31
3	34,35	34	35
4	42	42	42

Fig: Basic Block Table

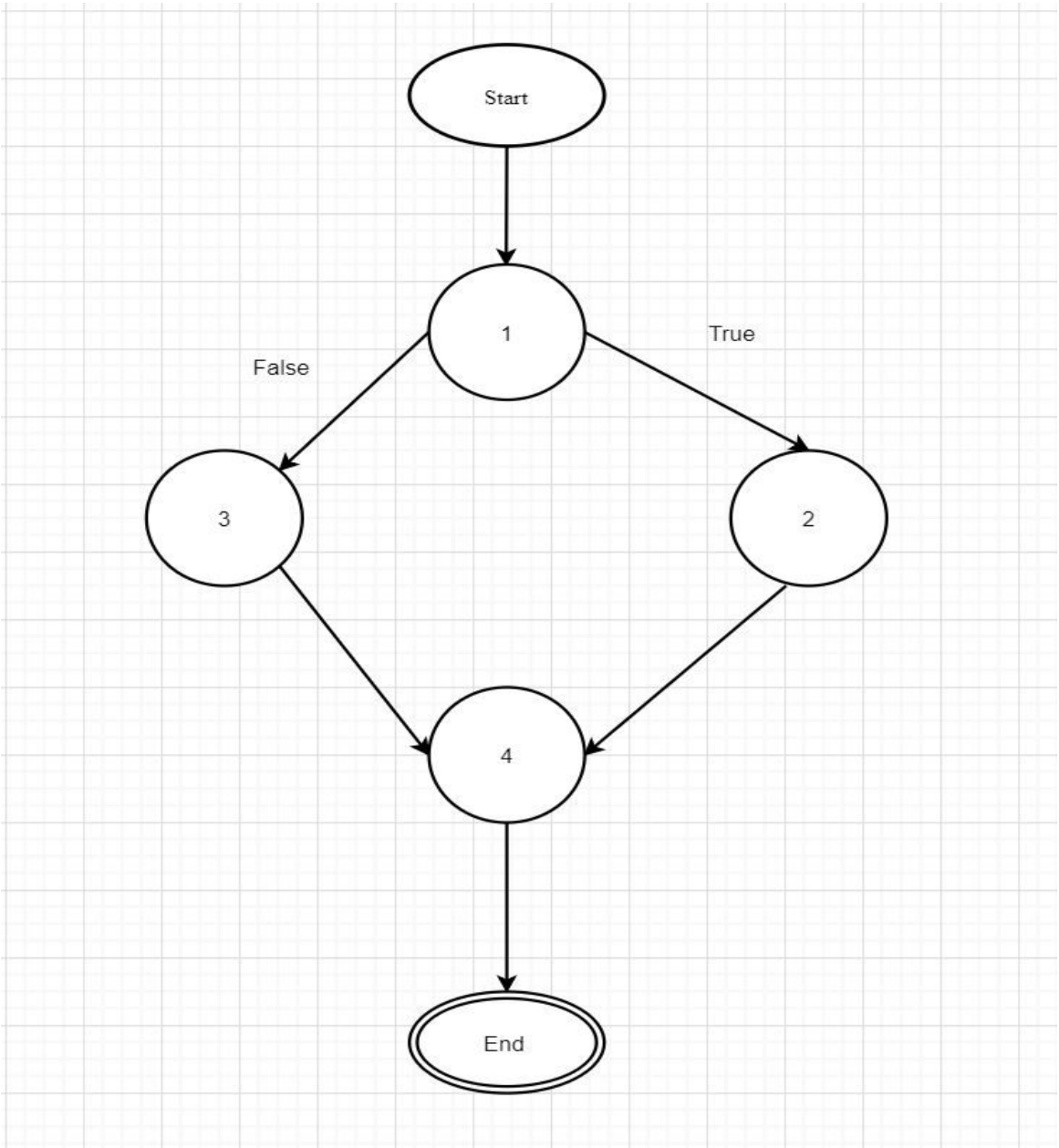

Fig: Control flow graph

2. Open_token_stream method:

```
76  /*********************************************************/
77  /* NAME:  open_token_stream                              */
78  /* INPUT:       a filename                               */
79  /* OUTPUT:      a BufferedReader              */
80  /* DESCRIPTION: when filename is EMPTY,choice standard   */
81  /*              input device as input source             */
82  /*********************************************************/
83  BufferedReader open_token_stream(String fname)
84  {
85    BufferedReader br;
86   if(fname.equals(null))
87      br=open_character_stream(null);
88   else
89      br=open_character_stream(fname);
90   return br;
91  }
92
```

Block	Lines	Entry	Exit
1	85,86	85	86
2	87	87	87
3	89	89	89
4	90	90	90

Fig: Basic Block Table

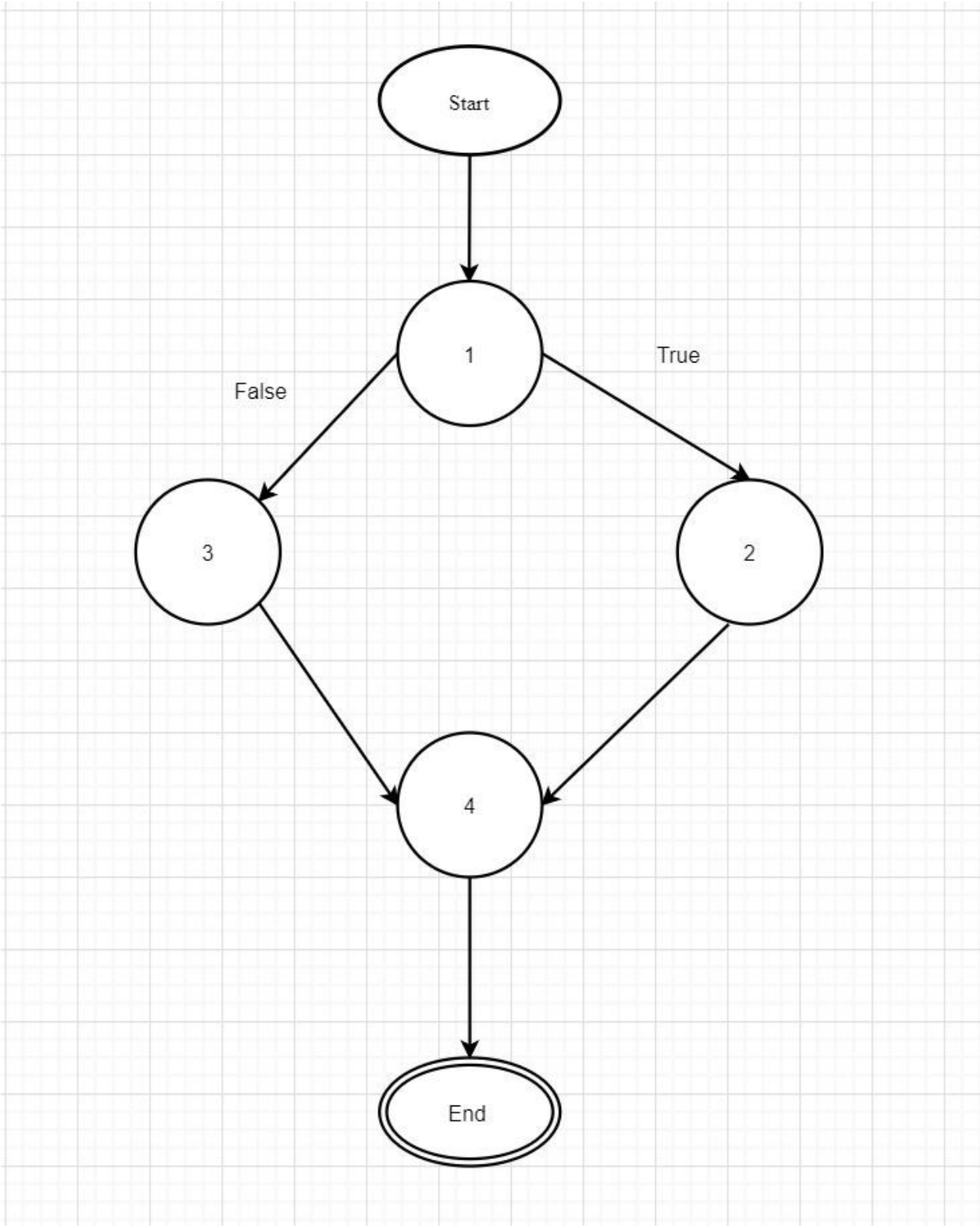

Fig: Control flow graph

3. Get_token method:

```
String get_token(BufferedReader br)
{
  int i=0,j;
  int id=0;
  int res = 0;
  char ch = '\0';

  StringBuilder sb = new StringBuilder();

   try {
     res = get_char(br);
     if (res == -1) {
       return null;
     }
     ch = (char)res;
  while(ch=='\t'||ch=='\n' || ch == '\r')      /* strip all blanks until meet characters */
      {
    res = get_char(br);
    ch = (char)res;
      }

   if(res == -1)return null;
   sb.append(ch);
   if(is_spec_symbol(ch)==true)return sb.toString();
   if(ch =='"')id=2;    /* prepare for string */
   if(ch ==59)id=1;    /* prepare for comment */

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

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

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

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

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

Block	Lines	Entry	Exit
1	102,103,104,105,107,110,111	102	111
2	112	112	112
3	114,115	114	115
4	117,118	117	118
5	121	121	121
6	121	121	121
7	122,123	122	123
8	123	123	123
9	124	124	124
10	124	124	124
11	125	125	125
12	125	125	125
13	127,128	127	128
14	129,130	129	130
15	132,134	132	134
16	136,137,138,139	136	139
17	142	142	142
18	140,145	140	145
19	146,147	146	147
20	150	150	150
21	151,152	151	152

22	**154**	**154**	**154**
23	**156,157**	**156**	**157**
24	**159**	**159**	**159**
25	**161,162**	**161**	**162**
26	**168**	**168**	**168**

Fig: Basic Block Table

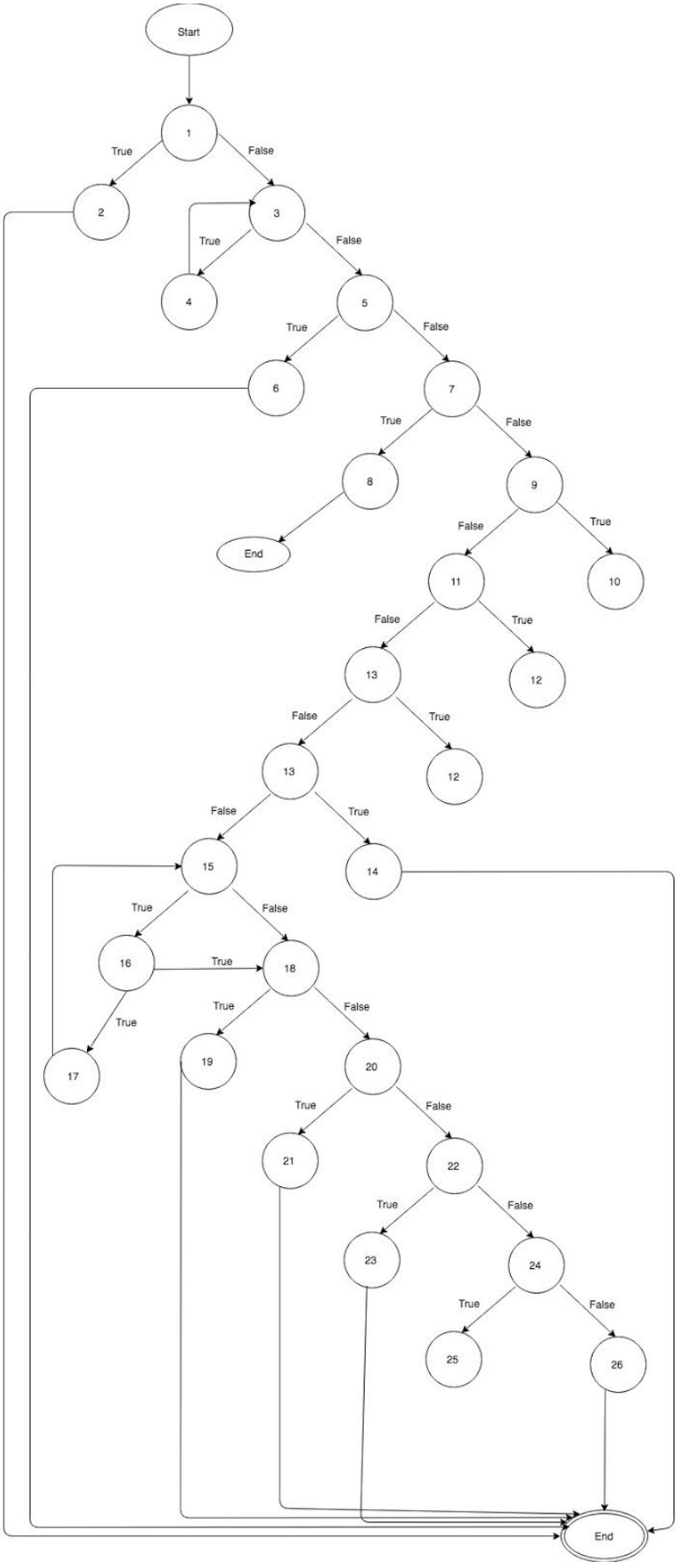

Fig: Control flow graph

4. Is_token_end method:

```
171   /*****************************************************/
172   /* NAME:  is_token_end                               */
173   /* INPUT:       a character,a token status            */
174   /* OUTPUT:  a BOOLEAN value                          */
175   /*****************************************************/
176   static boolean is_token_end(int str_com_id, int res)
177   {
178    if(res==-1)return(true); /* is eof token? */
179    char ch = (char)res;
180    if(str_com_id==1)          /* is string token */
181       { if(ch=='"' | ch=='\n' || ch == '\r')   /* for string until meet another " */
182           return true;
183         else
184           return false;
185       }
186
187    if(str_com_id==2)    /* is comment token */
188      { if(ch=='\n' || ch == '\r' || ch=='\t')     /* for comment until meet end of line */
189           return true;
190         else
191           return false;
192      }
193
194    if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
195    if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59) return true;
196
197    return false;              /* other case,return FALSE */
198   }
199
```

Block	Line	Entry	Exit
1	178	178	178
2	178	178	178
3	179, 180	179	180
4	181	181	181
5	182	182	182
6	184	184	184
7	187	187	187

8	**188**	**188**	**188**
9	**189**	**189**	**189**
10	**191**	**191**	**191**
11	**194**	**194**	**194**
12	**194**	**194**	**194**
13	**195**	**195**	**195**
14	**195**	**195**	**195**
15	**197**	**197**	**197**

Fig: Basic Block Table

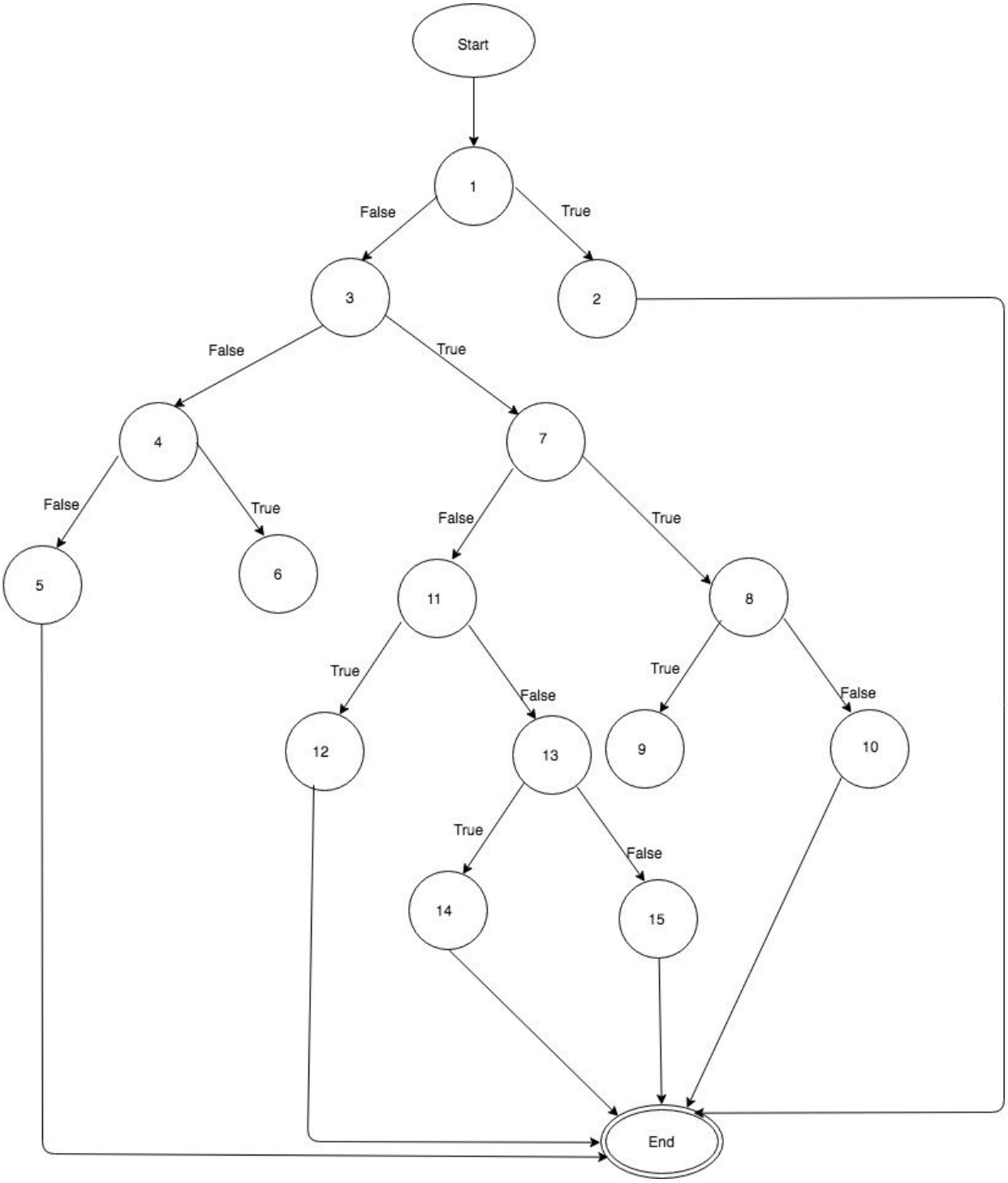

Fig: Control flow graph

5.token_type method:

```
202   /************************************************/
203   /* NAME : token_type                            */
204   /* INPUT:       a token              */
205   /* OUTPUT:      an integer value                */
206   /* DESCRIPTION: the integer value is corresponding  */
207   /*              to the different token type     */
208   /************************************************/
209   static int token_type(String tok)
210   {
211    if(is_keyword(tok))return(keyword);
212    if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
213    if(is_identifier(tok))return(identifier);
214    if(is_num_constant(tok))return(num_constant);
215    if(is_str_constant(tok))return(str_constant);
216    if(is_char_constant(tok))return(char_constant);
217    if(is_comment(tok))return(comment);
218    return(error);                  /* else look as error token */
219   }
```

Block	Line	Entry	Exit
1	211	211	211
2	211	211	211
3	212	212	212
4	212	212	212
5	213	213	213
6	213	213	213
7	214	214	214
8	214	214	214

9	**215**	**215**	**215**
10	**215**	**215**	**215**
11	**216**	**216**	**216**
12	**216**	**216**	**216**
13	**217**	**217**	**217**
14	**217**	**217**	**217**
15	**218**	**218**	**218**

Fig: Basic Block Table

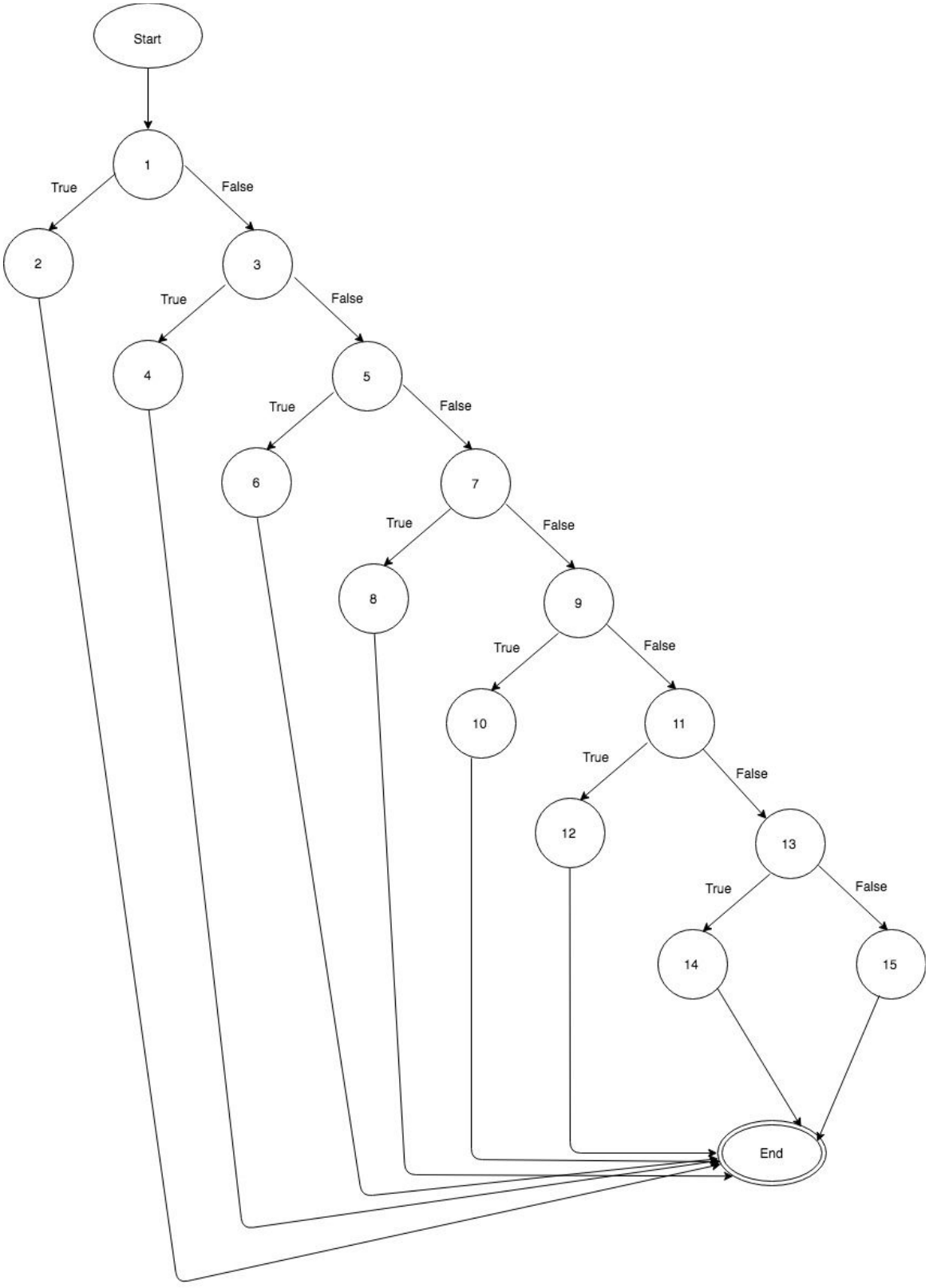

Fig: Control flow graph

6.print_token method:

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

Block	Line	Entry	Exit
1	226,227,228	226	228
2	230	230	230
3	233	233	233
4	235	235	235
5	238	238	238
6	238	238	238
7	239	239	239
8	241	241	241
9	243	243	243
10	245	245	245
11	247	247	247
12	249	249	249

Fig: Basic Block Table

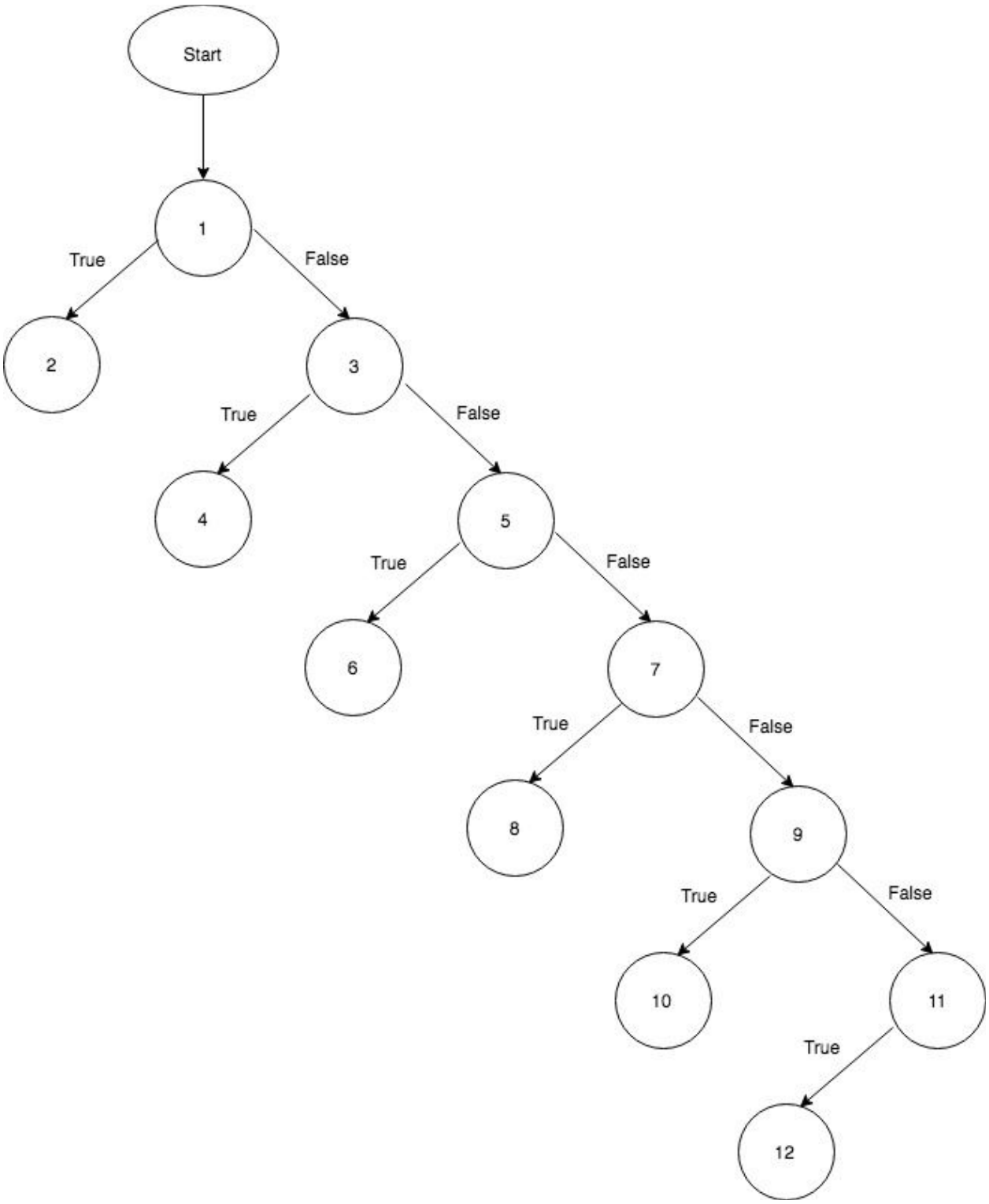

Fig: Control flow graph

7.is_comment method:

```
257  /************************************/
258  /* NAME:  is_comment           */
259  /* INPUT:   a token */
260  /* OUTPUT:      a BOOLEAN value      */
261  /************************************/
262  static boolean is_comment(String ident)
263  {
264    if( ident.charAt(0) ==59 )   /* the char is 59   */
265        return true;
266    else
267        return false;
268  }
```

Block	Line	Entry	Exit
1	264	264	264
2	265	265	265
3	267	267	267

Fig: Basic Block Table

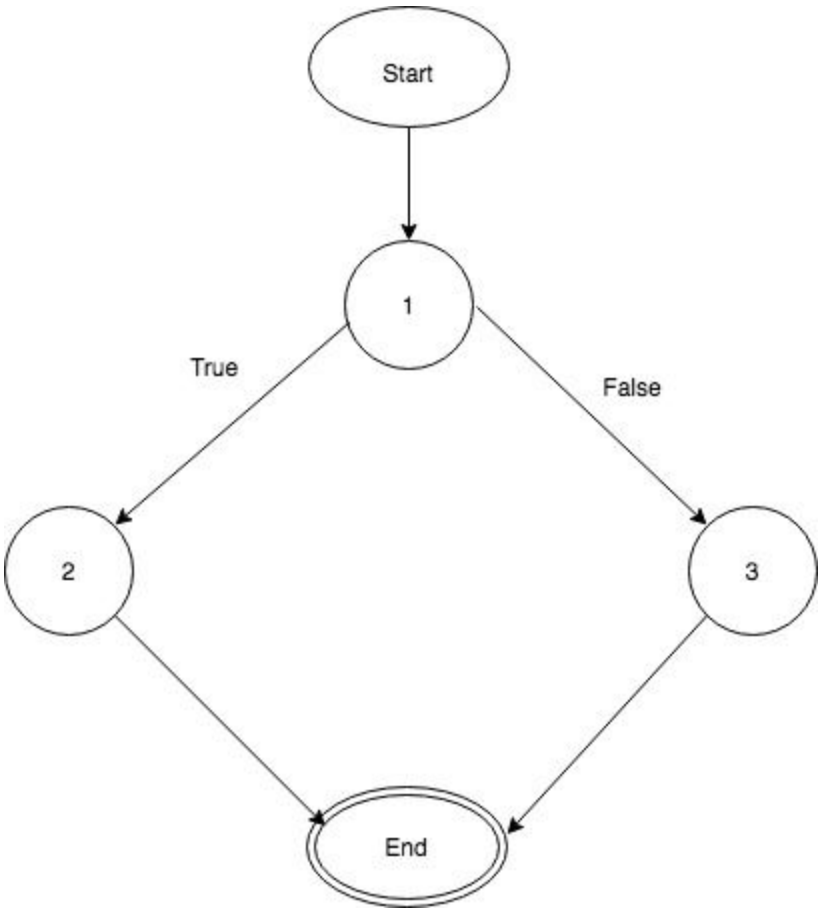

Fig: Control flow graph

8. Is_keyword method:

```
270   /************************************/
271   /* NAME:  is_keyword           */
272   /* INPUT:   a token */
273   /* OUTPUT:      a BOOLEAN value      */
274   /************************************/
275   static boolean is_keyword(String str)
276   {
277    if (str.equals("and") || str.equals("or") || str.equals("if") ||
278        str.equals("xor")||str.equals("lambda")||str.equals("=>"))
279         return true;
280     else
281         return false;
282   }
```

Block	Line	Entry	Exit
1	277	277	277
2	279	279	27
3	281	281	281

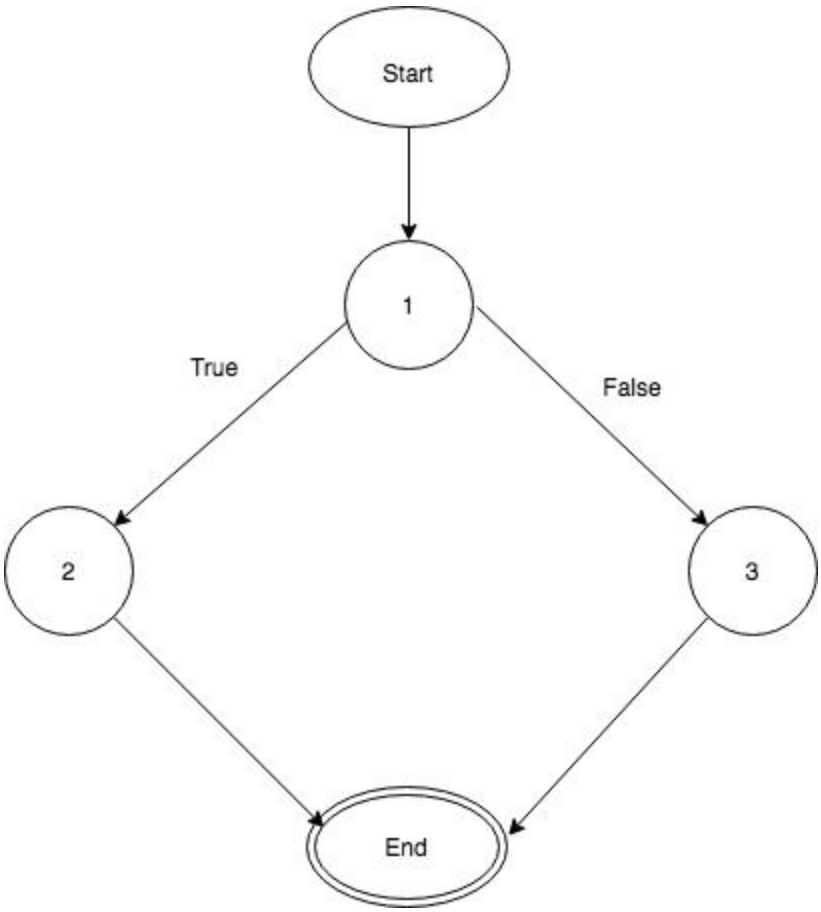

Fig: Control flow graph

9. Is_char_constant method:

```
284   /***********************************/
285   /* NAME:  is_char_constant     */
286   /* INPUT:   a token */
287   /* OUTPUT:      a BOOLEAN value      */
288   /***********************************/
289   static boolean is_char_constant(String str)
290   {
291     if (str.length() > 2 && str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
292         return true;
293     else
294         return false;
295   }
```

Block	Line	Entry	Exit
1	291	291	291
2	292	292	292
3	294	294	294

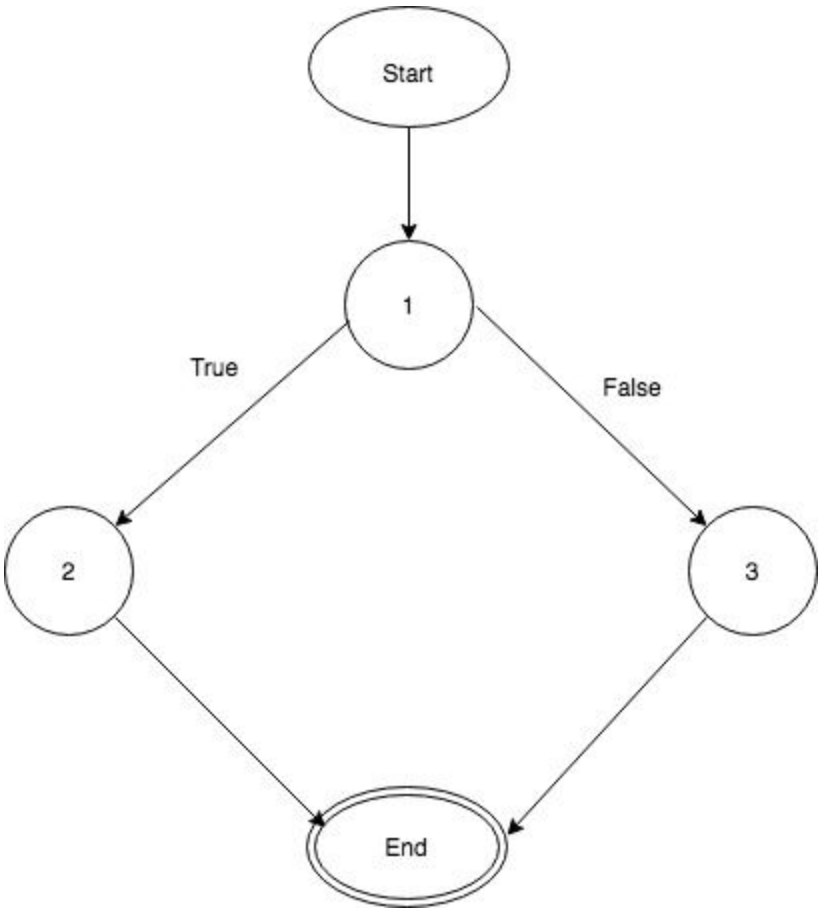

Fig: Control flow graph

10.is_num_constant

```
297   /***********************************/
298   /* NAME:  is_num_constant      */
299   /* INPUT:   a token */
300   /* OUTPUT:      a BOOLEAN value      */
301   /***********************************/
302   static boolean is_num_constant(String str)
303   {
304     int i=1;
305
306     if ( Character.isDigit(str.charAt(0)))
307       {
308       while ( i <= str.length() && str.charAt(i) != '\0' )   /* until meet token end sign */
309         {
310         if(Character.isDigit(str.charAt(i+1)))
311           i++;
312         else
313           return false;
314         }                          /* end WHILE */
315       return true;
316       }
317     else
318       return false;               /* other return FALSE */
319   }
```

Block	Line	Entry	Exit
1	304,306	304	306
2	308	308	308
3	310	310	310
4	311	311	311
5	313	313	313
6	315	315	315
7	318	318	318

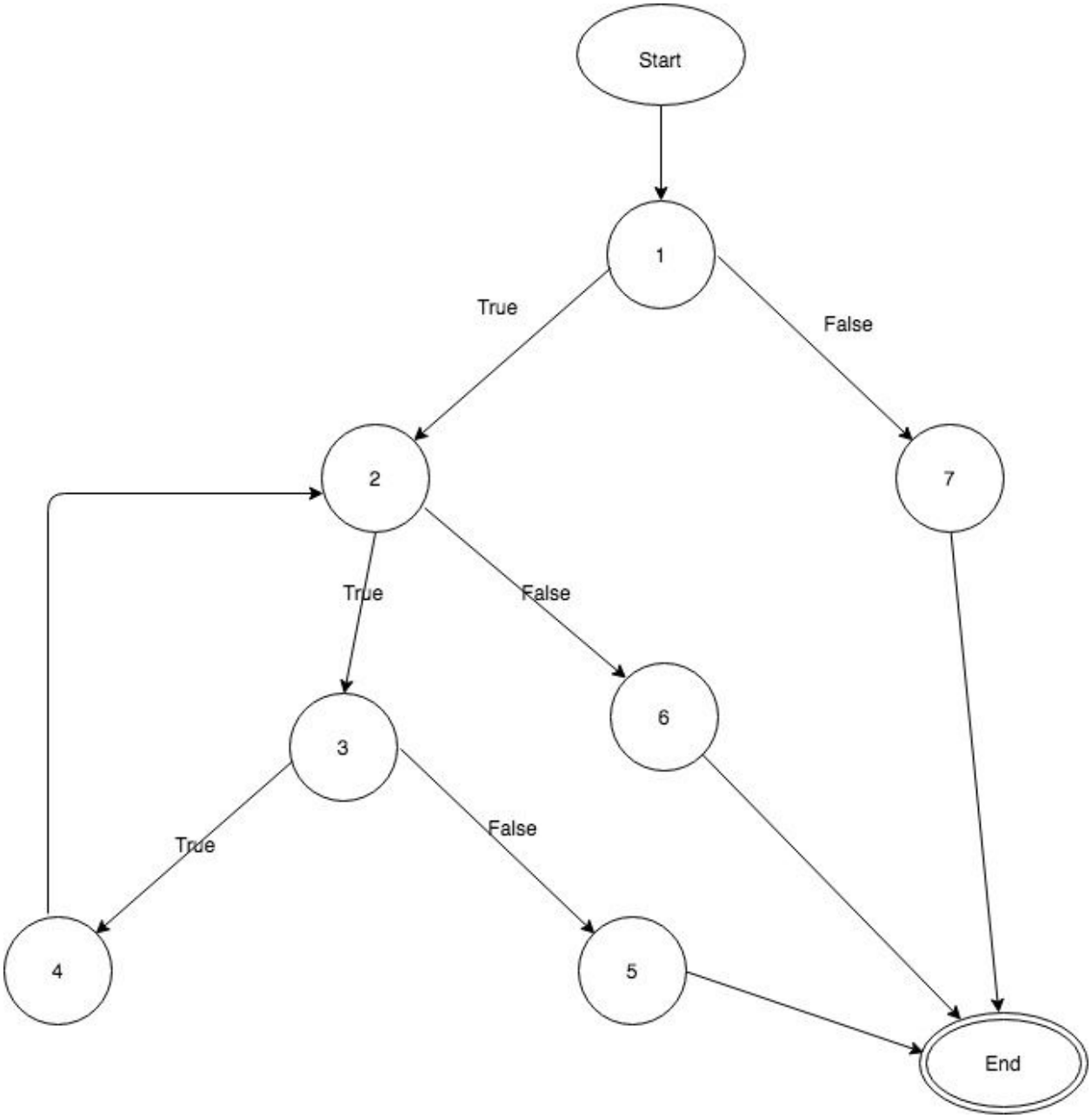

Fig: Control flow graph

11.is_str_constant

```
321  /**********************************/
322  /* NAME:  is_str_constant      */
323  /* INPUT:   a token */
324  /* OUTPUT:      a BOOLEAN value      */
325  /**********************************/
326  static boolean is_str_constant(String str)
327  {
328    int i=1;
329
330    if ( str.charAt(0) =='"')
331       { while (i < str.length() && str.charAt(0)!='\0')  /* until meet the token end sign */
332           { if(str.charAt(i)=='"')
333               return true;        /* meet the second '"'           */
334             else
335             i++;
336           }               /* end WHILE */
337       return true;
338       }
339    else
340      return false;       /* other return FALSE */
341  }
```

Block	Line	Entry	Exit
1	328,330	328	330
2	331	331	331
3	332	332	332
4	333	333	333
5	335	335	335
6	337	337	337
7	340	340	340

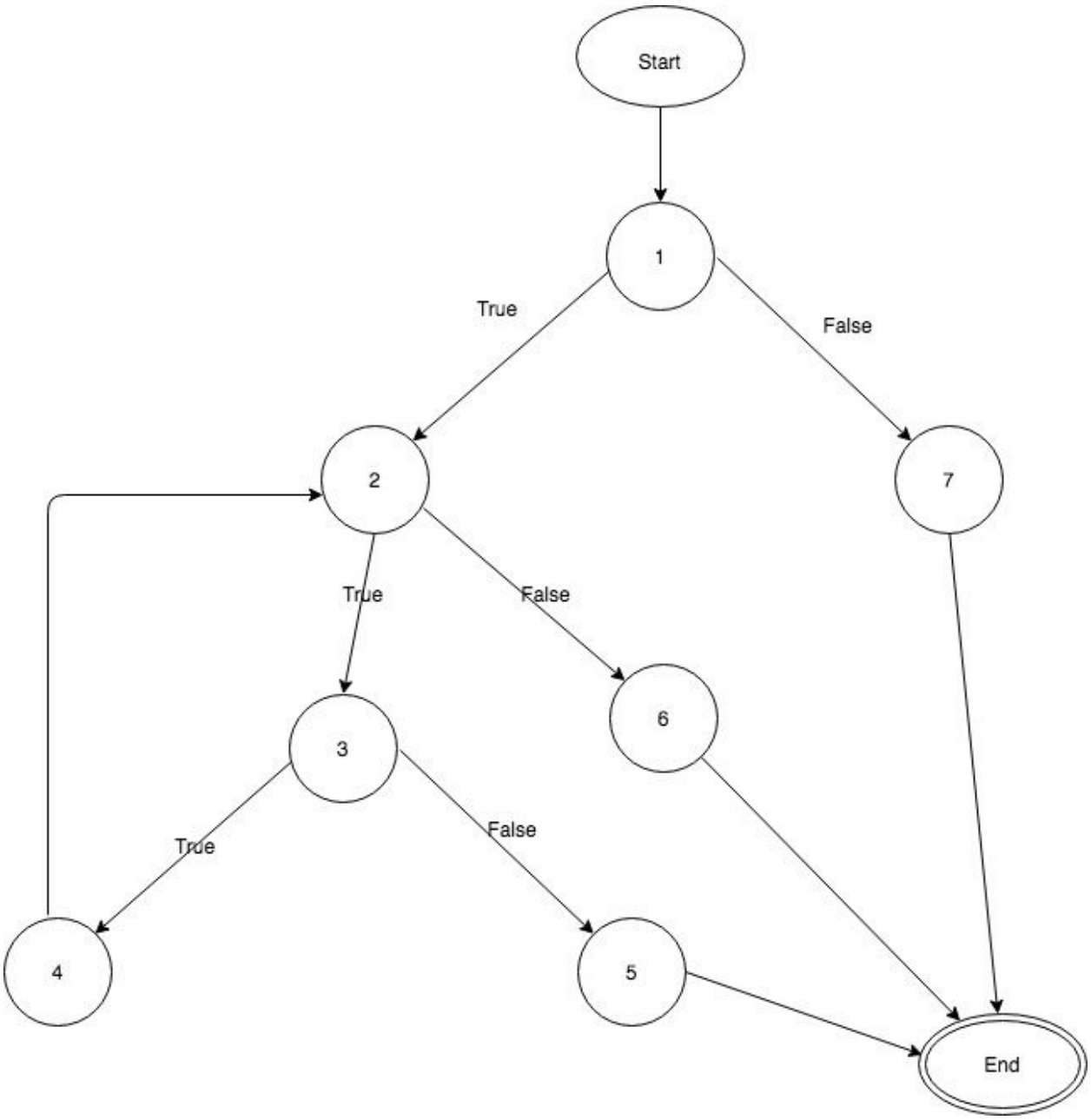

Fig: Control flow graph

12. Is_identifier method:

```
343  /***********************************/
344  /* NAME:  is_identifier         */
345  /* INPUT:   a token */
346  /* OUTPUT:      a BOOLEAN value      */
347  /***********************************/
348  static boolean is_identifier(String str)
349  {
350    int i=0;
351
352    if ( Character.isLetter(str.charAt(0)) )
353      {
354        while(i < str.length() && str.charAt(i) !='\0' )   /* unti meet the end token sign */
355          {
356           if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
357             i++;
358           else
359             return false;
360          }      /* end WHILE */
361      return false;
362      }
363    else
364      return true;
365  }
```

Block	Line	Entry	Exit
1	350,352	350	352
2	354	354	354
3	356	356	356
4	357	357	357
5	359	359	359
6	361	361	361
7	364	364	364

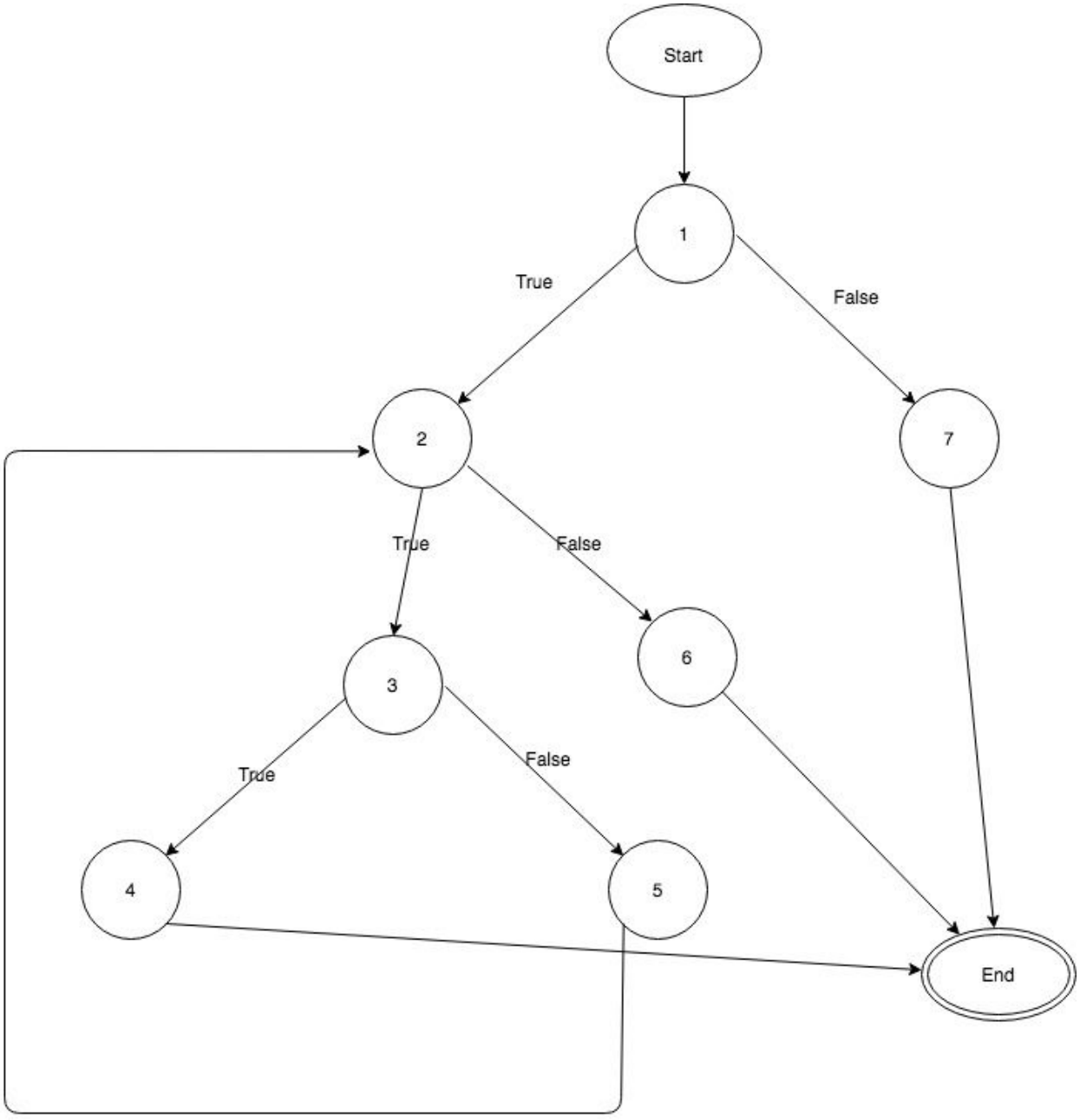

Fig: Control flow graph

13. Print_spec_symbol method:

```
/*************************************************/
/* NAME:        print_spec_symbol                */
/* INPUT:       a spec_symbol token */
/* OUTPUT :     print out the spec_symbol token  */
/*              according to the form required   */
/*************************************************/
static void print_spec_symbol(String str){
    if      (str.equals("{")){
          System.out.print("lparen.\n");
              return;
    }
    if (str.equals(")")){
       System.out.print("rparen.\n");
              return;
    }
    if (str.equals("[")){
       System.out.print("lsquare.\n");
              return;
    }
    if (str.equals("]")){
       System.out.print("rsquare.\n");
              return;
    }
    if (str.equals("'")){
              System.out.print("quote.\n");
              return;
    }
    if (str.equals("`")){
             System.out.print("bquote.\n");
              return;
    }
  }
```

Block	Line	Entry	Exit
1	384	384	384
2	385,386	385	386
3	388	388	388
4	389,390	389	390
5	392	392	392
6	393,394	393	394
7	396	396	396
8	397,398	397	398
9	400	400	400
10	401,402	401	402
11	404	404	404
12	405,406	405	406

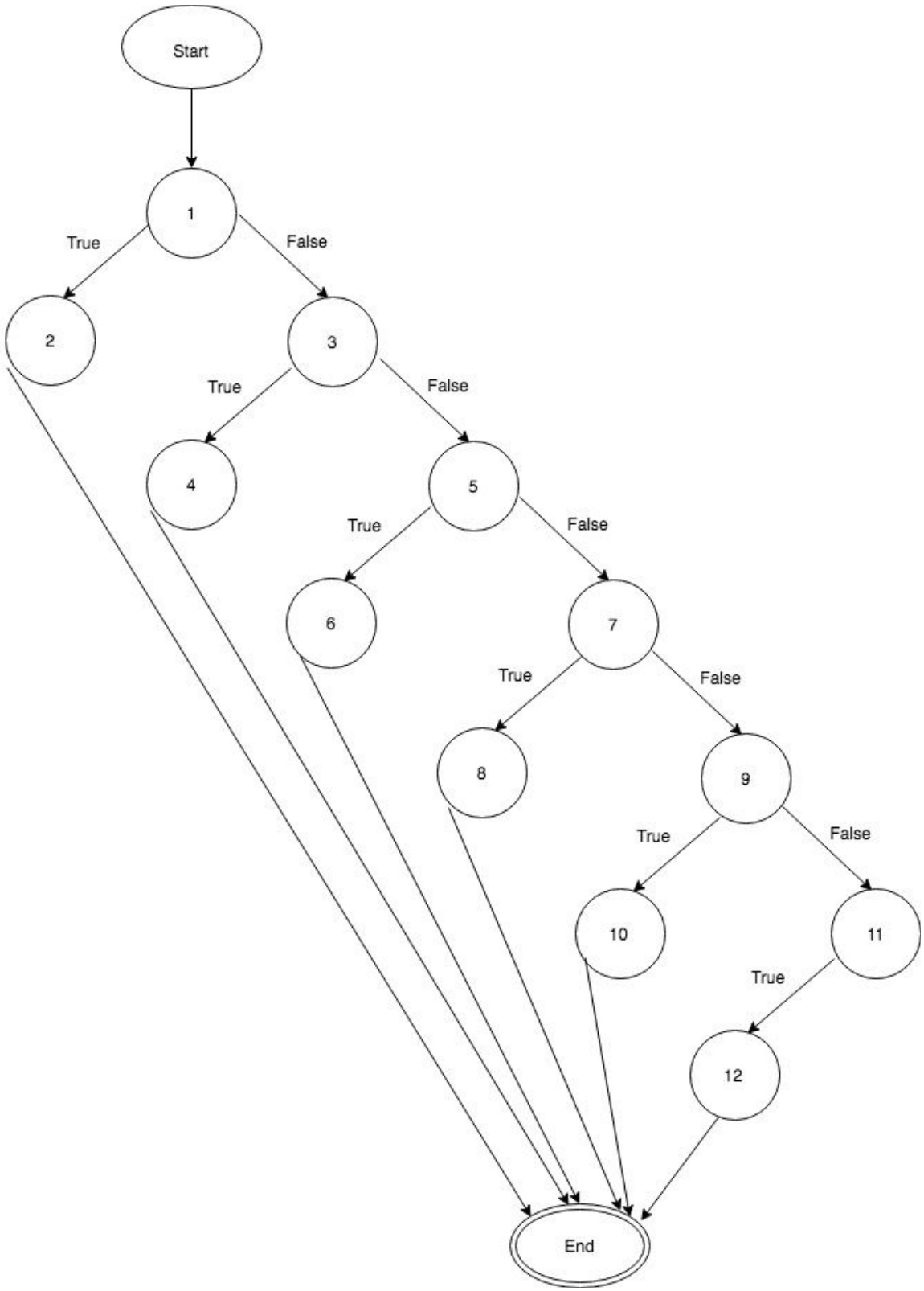

Fig: Control flow graph

14. Is_spec_symbol method:

```
411     /* NAME:         is_spec_symbol       */
412     /* INPUT:        a token */
413     /* OUTPUT:       a BOOLEAN value      */
414     /*************************************/
415     static boolean is_spec_symbol(char c)
416     {
417         if (c == '(')
418         {
419             return true;
420         }
421         if (c == ')')
422         {
423             return true;
424         }
425         if (c == '[')
426         {
427             return true;
428         }
429         if (c == ']')
430         {
431             return true;
432         }
433         if (c == '/')
434         {
435             return true;
436         }
437         if (c == '`')
438         {
439             return true;
440         }
441         if (c == ',')
442         {
443             return true;
444         }
445         return false;     /* others return FALSE */
446     }
```

Block	Line	Entry	Exit
1	417	417	417
2	419	419	419
3	421	421	421
4	423	423	423
5	425	425	425
6	427	427	427
7	429	429	429
8	431	431	431
9	433	433	433
10	435	435	435
11	437	437	437
12	439	439	439
13	441	441	441
14	443	443	443
15	445	445	445

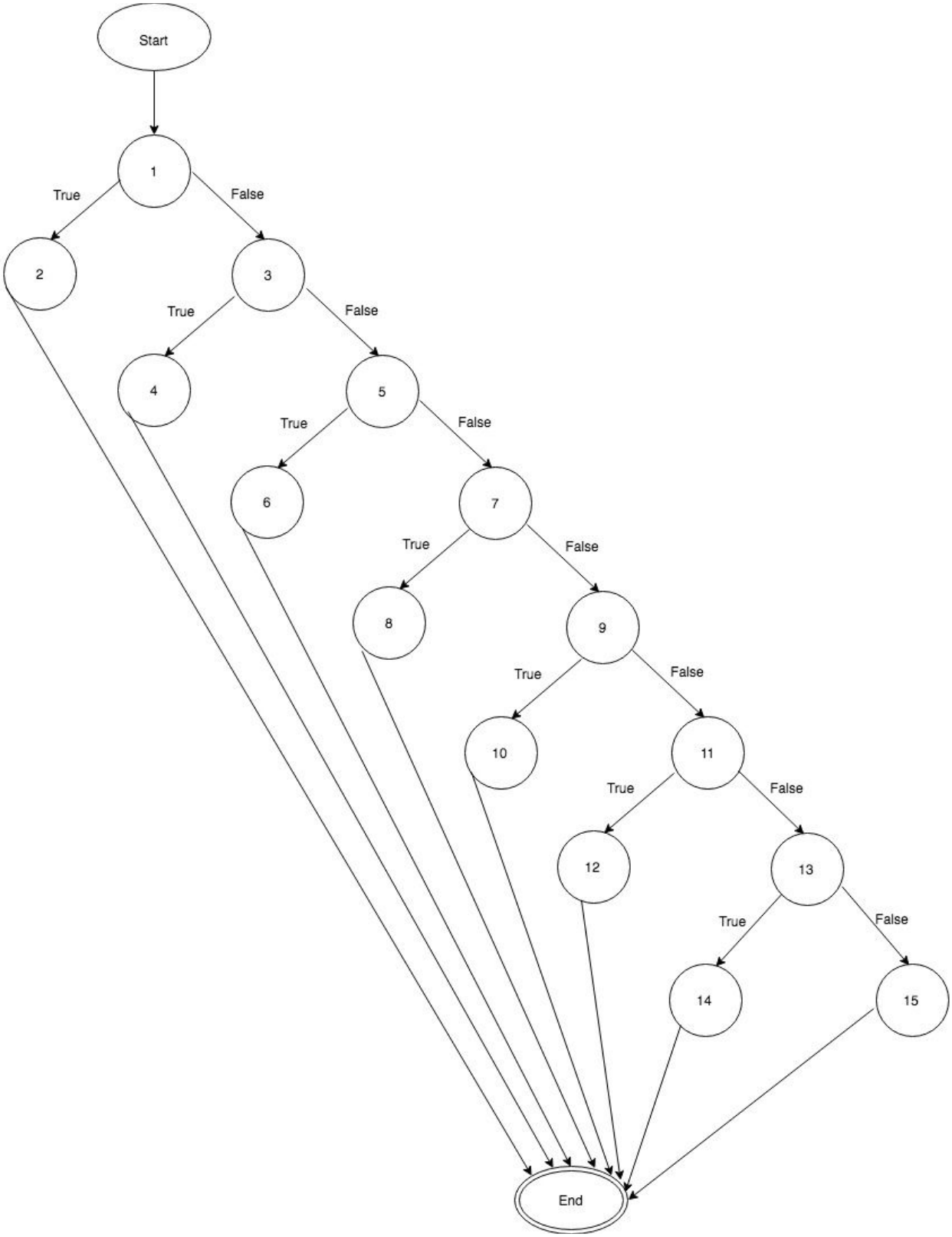

Fig: Control flow graph

15. main method:

```
448     public static void main(String[] args) throws IOException {
449       String fname = null;
450       if (args.length == 0) { /* if not given filename,take as '"'' */
451         fname = new String();
452       } else if (args.length == 1) {
453         fname = args[1];
454       } else {
455         System.out.print("Error!,please give the token stream\n");
456         System.exit(0);
457       }
458       Printtokens2 t = new Printtokens2();
459       BufferedReader br = t.open_token_stream(fname); /* open token stream */
460       String tok = t.get_token(br);
461       while (tok != null) { /* take one token each time until eof */
462         t.print_token(tok);
463         tok = t.get_token(br);
464       }
465
466       System.exit(0);
467     }
468   }
```

Block	Line	Entry	Exit
1	449,450	449	450
2	451	451	451
3	452	452	452
4	453	453	453
5	455,456	455	456
6	458,459,460,461	458	461
7	462,463	462	463
8	466	466	466

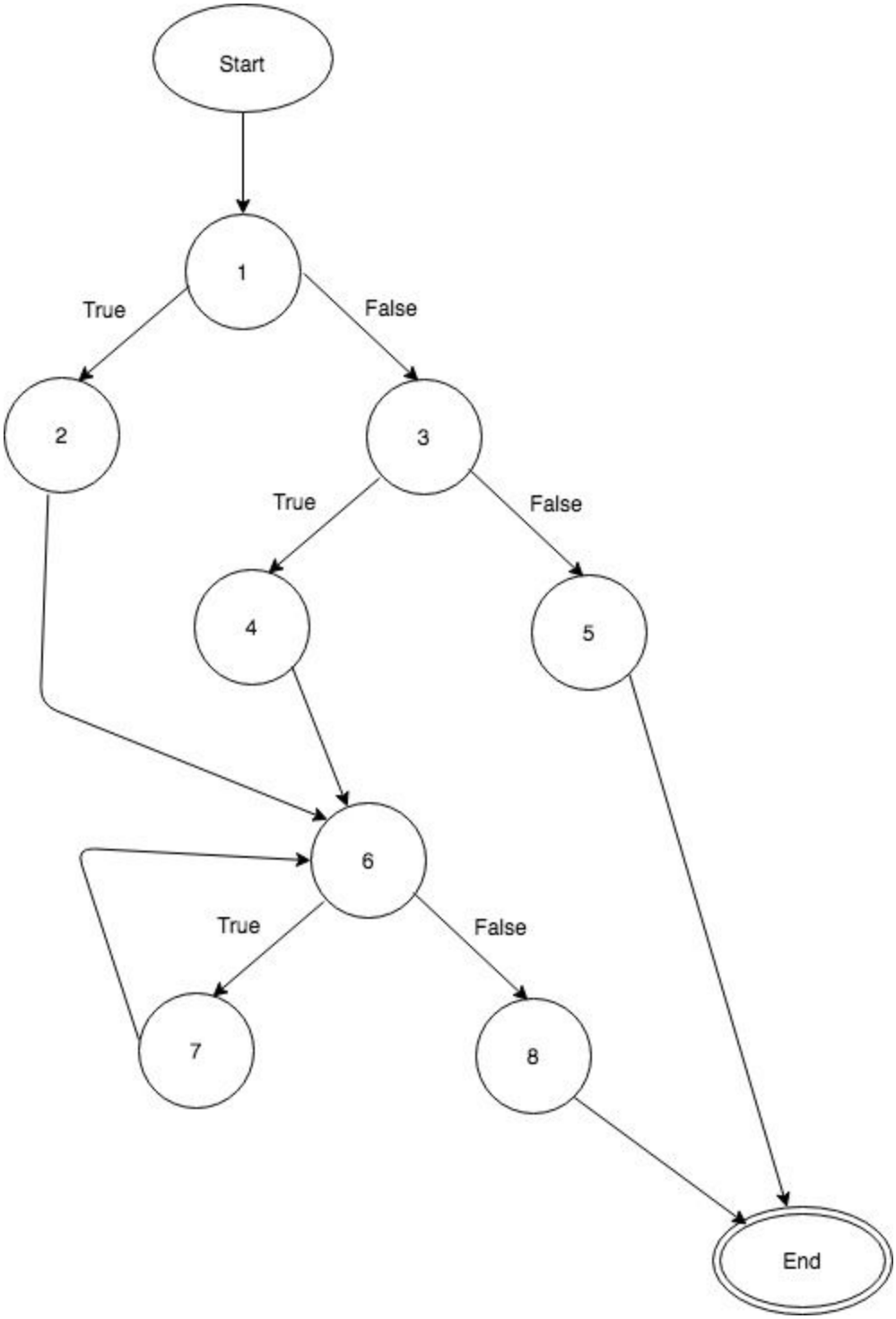

Fig: Control flow graph