CSE-4321 Software Testing Final Project - Control Flow Graphs

Maximum Effort: Elliot Mai & Benjamin Niccum

December 2, 2024

Contents

1 Method: openCharacterStream

```
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    }
```

Figure 1: Code Snippet for openCharacterStream

Block Number	Lines	Entry	Exit	Function Calls
1	23, 24	23	24	
2	25	25	25	
3	26 - 34	26	34	
4	36	36	36	

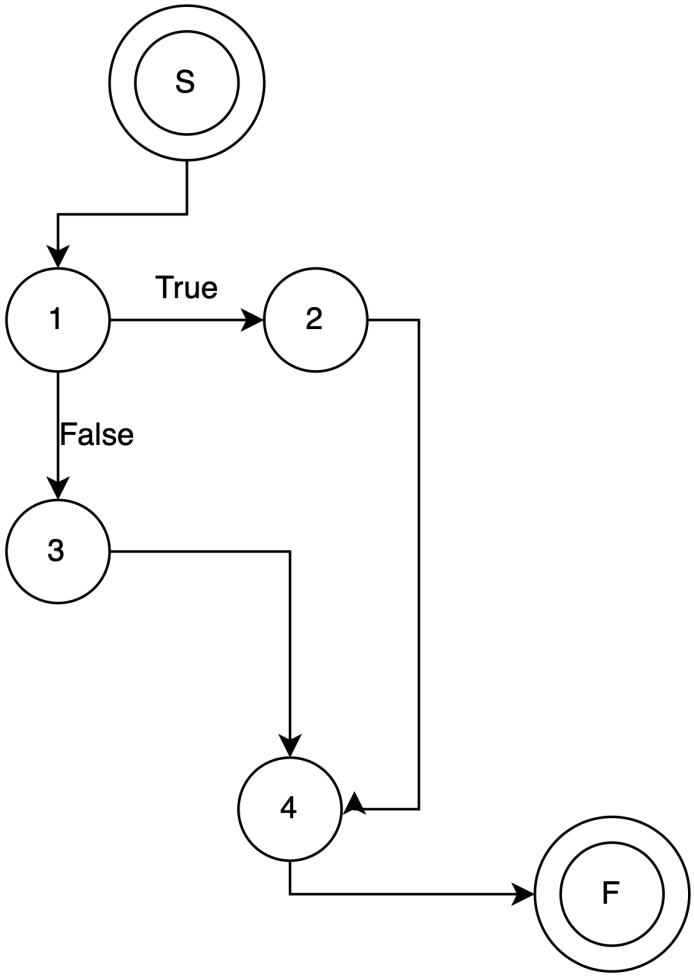

Figure 2: Control Flow Graph for openCharacterStream

2 Method: getChar

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

Figure 3: Code Snippet for getChar

Block Number	Lines	Entry	Exit	Function Calls
1	45 - 51	45	51	
2	1	52	52	

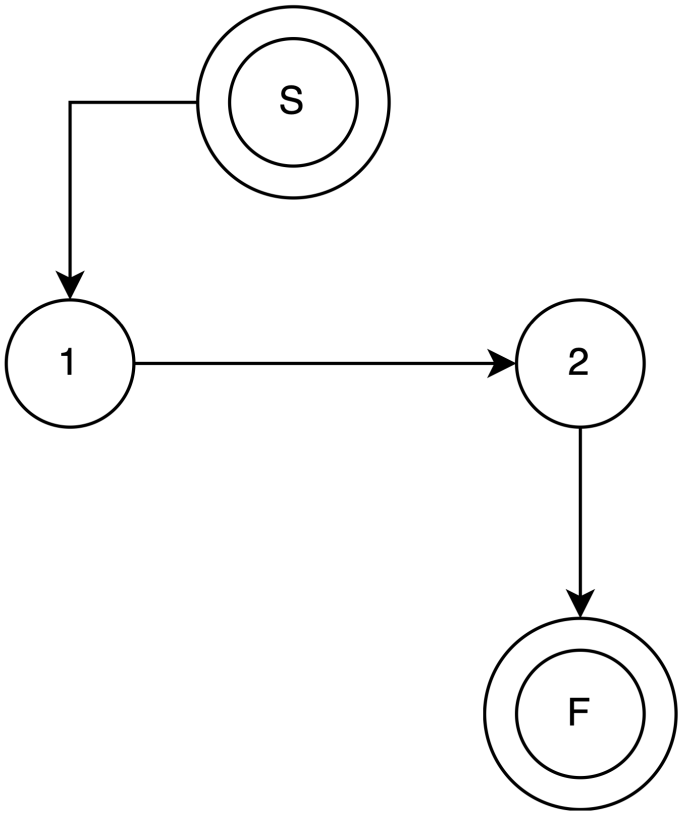

Figure 4: Control Flow Graph for getChar

3 Method: ungetChar

```
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    }
```

Figure 5: Code Snippet for ungetChar

Block Number	Lines	Entry	Exit	Function Calls
1	62 - 66	62	66	
2	67	67	67	

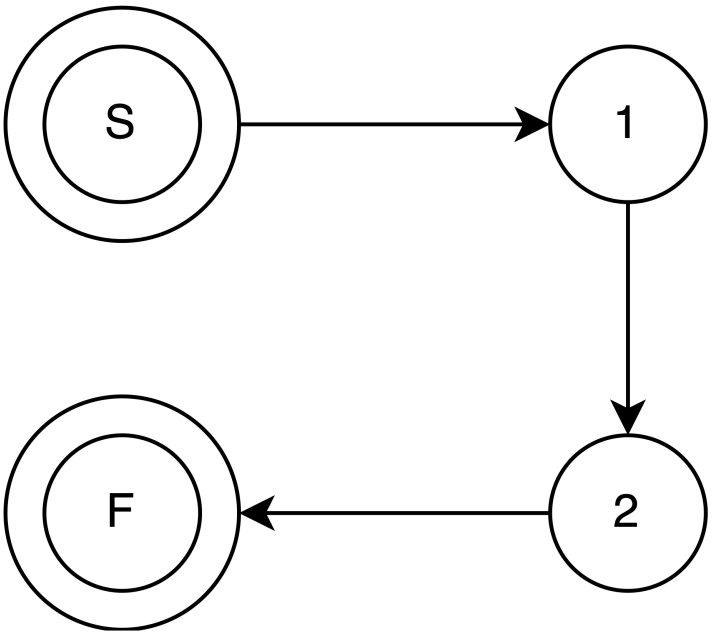

Figure 6: Control Flow Graph for ungetChar

4 Method: openTokenStream

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

Figure 7: Code Snippet for openTokenStream

Block Number	Lines	Entry	Exit	Function Calls
1	78 - 80	78	80	
2	81	81	81	openCharacterStream
3	82, 83	82	83	openCharacterStream

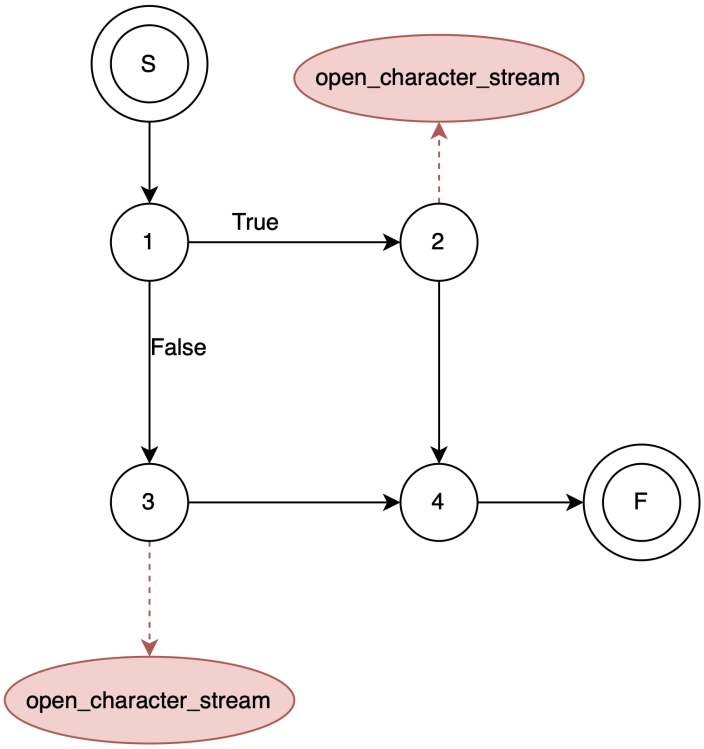

Figure 8: Control Flow Graph for openTokenStream

5 Method: getToken

```
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 the end character */
129      {
130        sb.append(ch);
131        br.mark(readAheadLimit:4);
132        res = get_char(br);
133        if (res == -1) {
134          break;
135        }
136        ch = (char)res;
137      }
138
139      if(res == -1)       /* if end character is eof token    */
140        { unget_char(ch,br);        /* then put back eof on token_stream */
141          return sb.toString();
142        }
143
144      if(is_spec_symbol(ch)==true)     /* if end character is special_symbol */
145        { unget_char(ch,br);        /* then put back this character       */
146          return sb.toString();
147        }
148      if(id==1)                  /* if end character is " and is string */
149       {
150         if (ch == '"') {
151           sb.append(ch);
152         }
153         return sb.toString();
154       }
155      if(id==0 && ch==59)
156                                  /* when not in string or comment,meet ";" */
157        { unget_char(ch,br);       /* then put back this character         */
158          return sb.toString();
159        }
160   } catch (IOException e) {
161     e.printStackTrace();
162   }
163
164     return sb.toString();                  /* return nomal case token            */
165   }
```

Block Number	Lines	Entry	Exit	Function Calls
1	95 - 108	95	108	
2	109	109	109	
3	110 - 113	110	113	
4	115	115	115	
5	116, 117	116	117	isSpecialSymbol
6	118	118	118	
7	119	119	119	
8	121, 122	121	122	getChar
9	123 - 125	123	125	ungetChar
10	126	126	126	
11	128	128	128	isTokenEnd
12	129 - 137	129	137	getChar
13	139	139	139	
14	140 - 142	140	142	ungetChar
15	143	143	143	
16	145 - 147	145	147	ungetChar
17	158	158	158	
18	149 - 154	149	154	
19	155	155	155	
20	156 - 162	156	162	ungetChar
21	164	164	164	

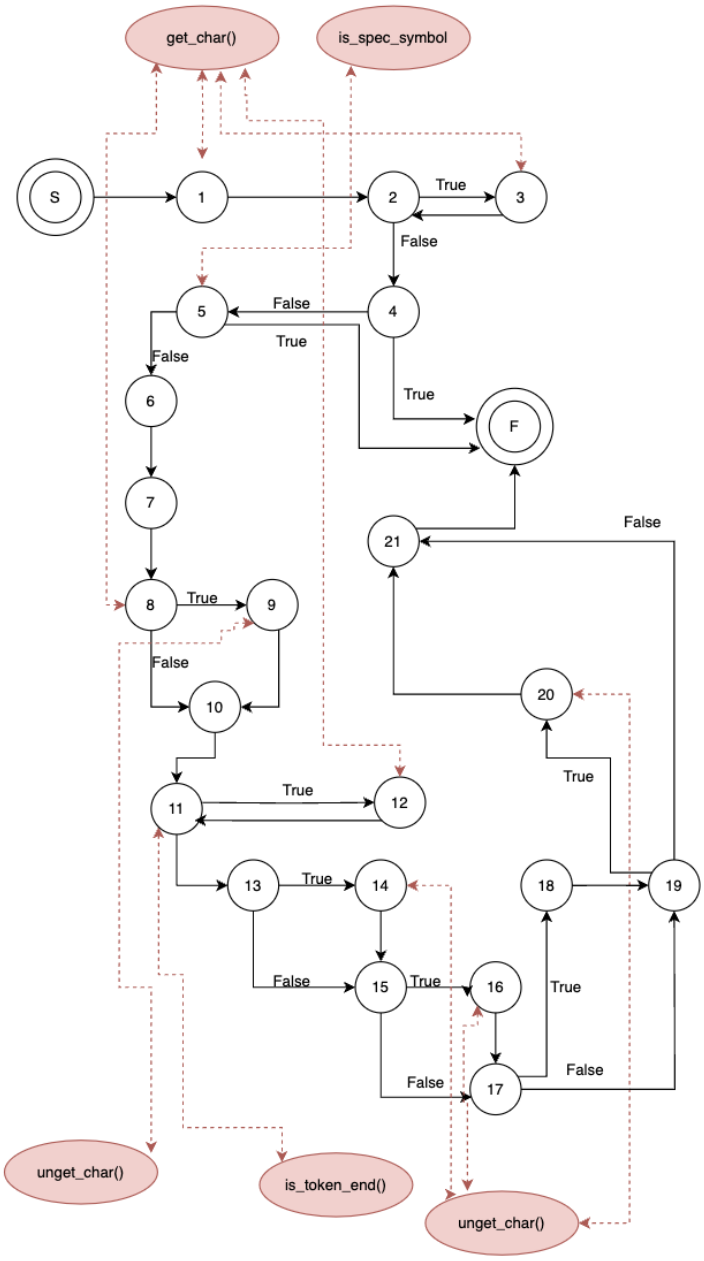

Figure 10: Control Flow Graph for getToken

6 Method: isTokenEnd

```
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')   /* for string until meet another " */
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 comment until meet end of line */
185  |     return true;
186  |   else
187  |     return false;
188    }
189
190  if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
191  if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59) return true;
192     |   |   |   |   |   |   | /* others until meet blank or tab or 59 */
193  return false;              /* other case,return FALSE */
194  }
```

Figure 11: Code Snippet for isTokenEnd

Block Number	Lines	Entry	Exit	Function Calls
1	174	174	174	
2	175, 176	175	176	
3	177	177	177	
4	178	178	178	
5	179 - 181	179	181	
6	183	183	183	
7	184	184	184	
8	185	185	185	
9	186 - 188	186	188	
10	190	190	190	isSpecialSymbol
11	191a	191a	191a	
12	191b	191b	191b	
13	193	193	193	

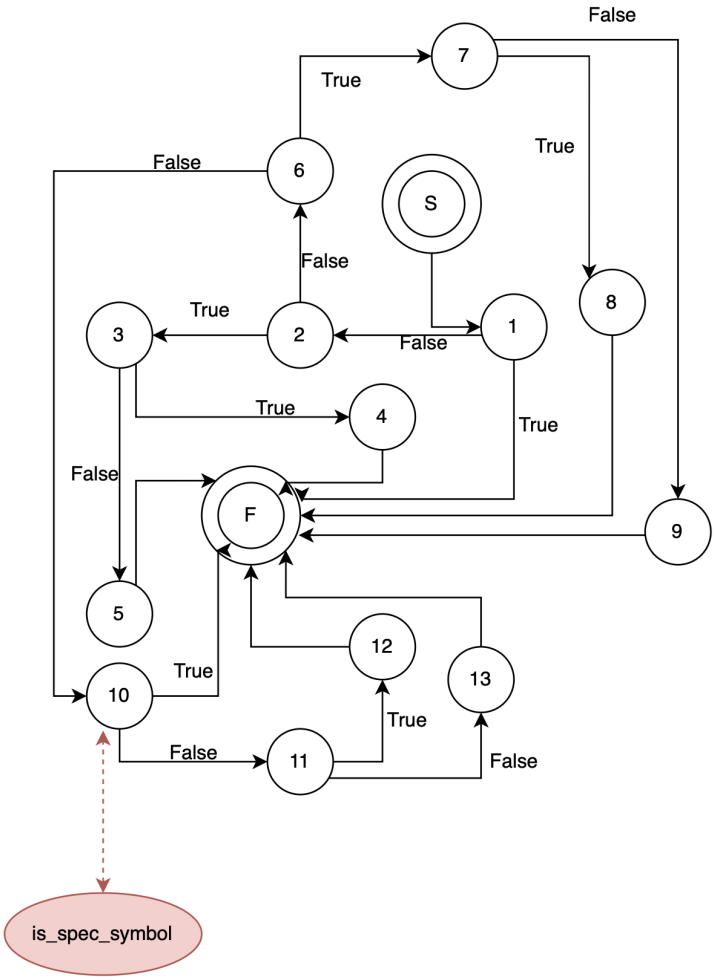

Figure 12: Control Flow Graph for isTokenEnd

7 Method: tokenType

```
203     static int token_type(String tok)
204     {
205     if(is_keyword(tok))return(keyword);
206     if(is_spec_symbol(tok.charAt(index: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 error token */
213     }
```

Figure 13: Code Snippet for tokenType

Block Number	Lines	Entry	Exit	Function Calls
1	205a	205a	205a	isKeyword
2	205b	205b	205b	
3	206a	206a	206a	isSpecialSymbol
4	206b	206b	206b	
5	207a	207a	207a	isIdentifier
6	207b	207b	207b	
7	208a	208a	208a	isNumConstant
8	208b	208b	208b	
9	209a	209a	209a	isStrConstant
10	209b	209b	209b	
11	210a	210a	210a	isCharConstant
12	210b	210b	210b	
13	211a	211a	211a	isComment
14	211b	211b	211b	
15	212	212	212	

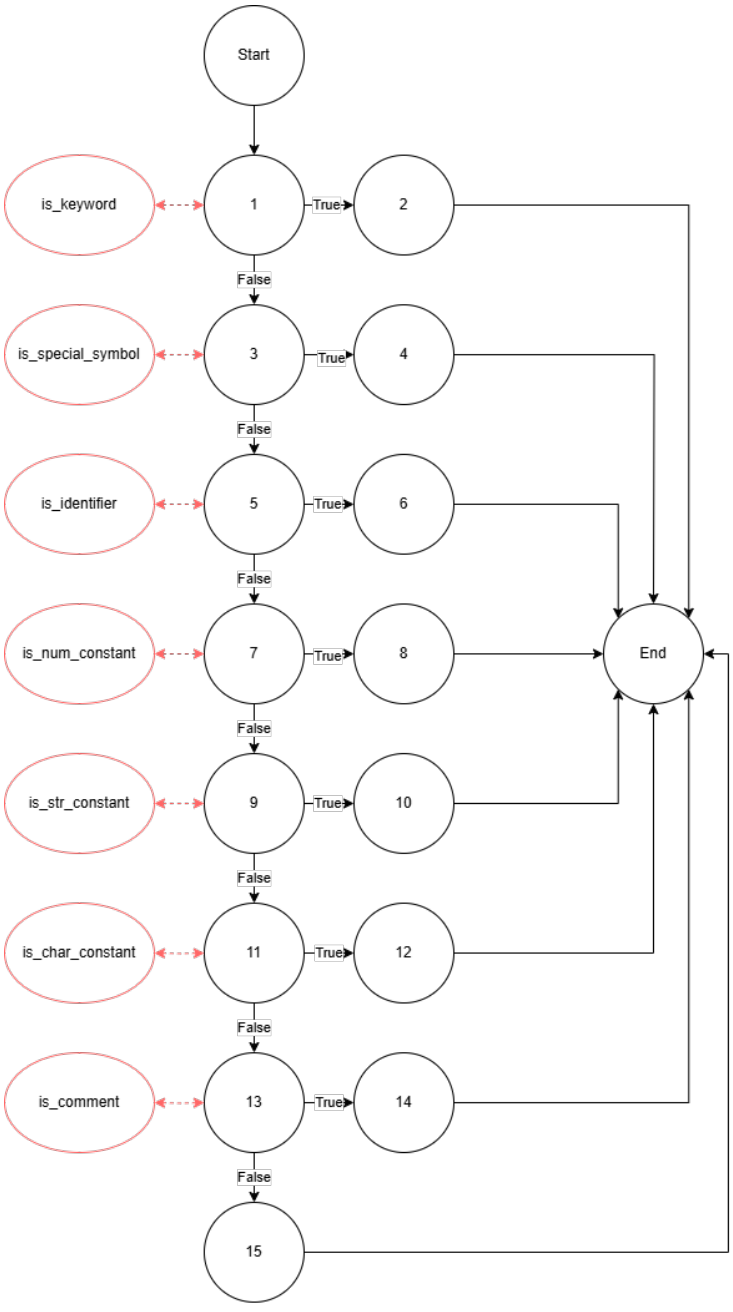

Figure 14: Control Flow Graph for tokenType

8 Method: printToken

```
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(index:1) + "\".\n");
248         }
249       if(type==comment)
250         {
251          System.out.print("comment,\"" + tok + "\".\n");
252         }
253     }
```

Figure 15: Code Snippet for printToken

Block Number	Lines	Entry	Exit	Function Calls
1	220, 222	220	222	tokenType
2	223, 225	223	225	
3	227	227	227	
4	228, 230	228	230	
5	232-a	232-a	232-a	
6	232-b	232-b	232-b	printSpecSymbol
7	233	233	233	
8	234-236	234	236	
9	237	237	237	
10	238, 240	238	240	
11	241	241	241	
12	242, 244	242	244	
13	241	241	241	
14	246, 248	246	248	
15	249	249	249	
16	250, 252	250	252	

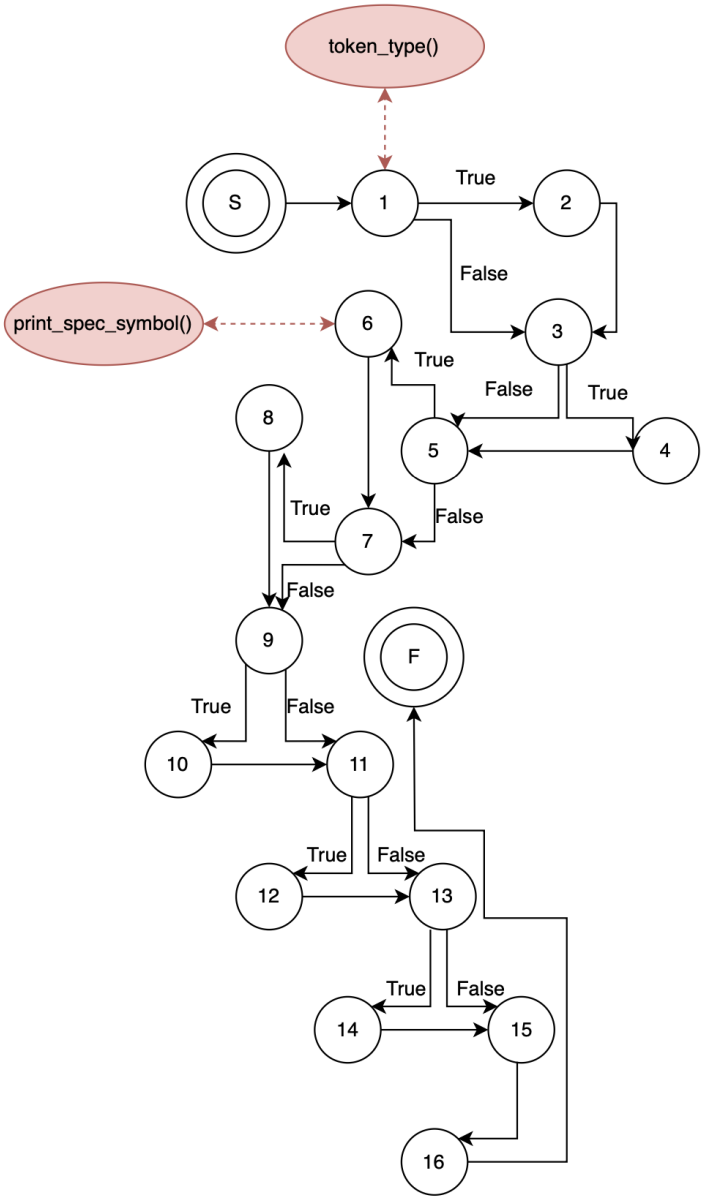

Figure 16: Control Flow Graph for printToken

9 Method: isComment

```
263     static boolean is_comment(String ident)
264     {
265       if( ident.charAt(index:0) ==59 )   /* the char is 59   */
266         return true;
267       else
268         return false;
269     }
```

Figure 17: Code Snippet for isComment

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

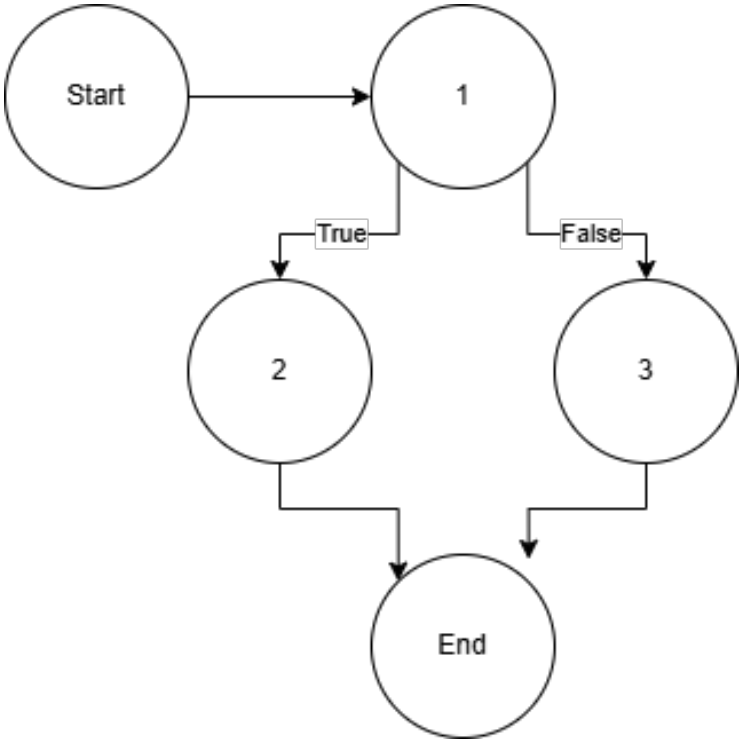

Figure 18: Control Flow Graph for isComment

10 Method: isKeyword

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

Figure 19: Code Snippet for isKeyword

Block Number	Lines	Entry	Exit	Function Calls
1	278a, 278b, 278c, 278d, 278e, 278f	278a	278f	
2	280	280	280	
3	282	282	282	

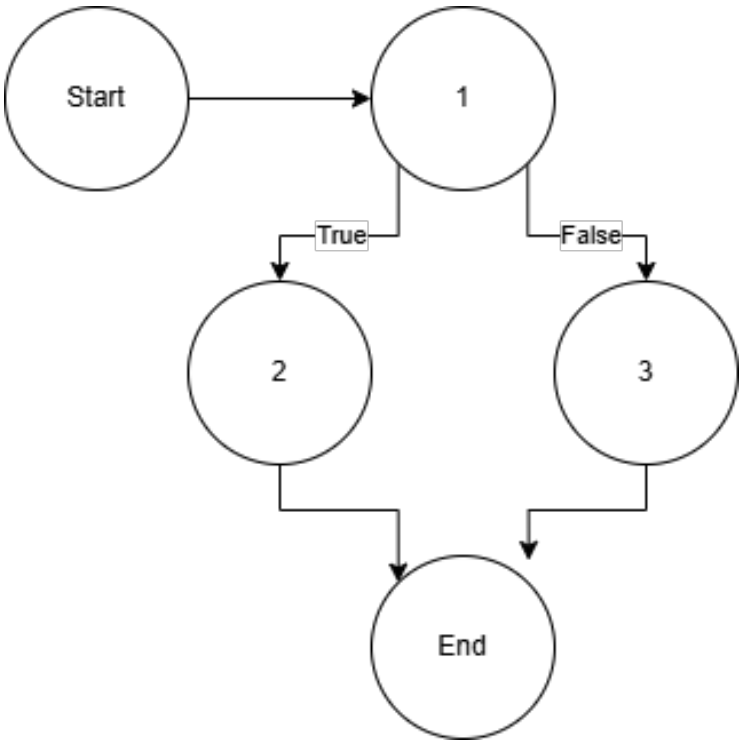

Figure 20: Control Flow Graph for isKeyword

11 Method: isCharConstant

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

Figure 21: Code Snippet for isCharConstant

Block Number	Lines	Entry	Exit	Function Calls
1	292a, 292b, 292c	292a	292c	
2	293	293	293	
3	295	295	295	

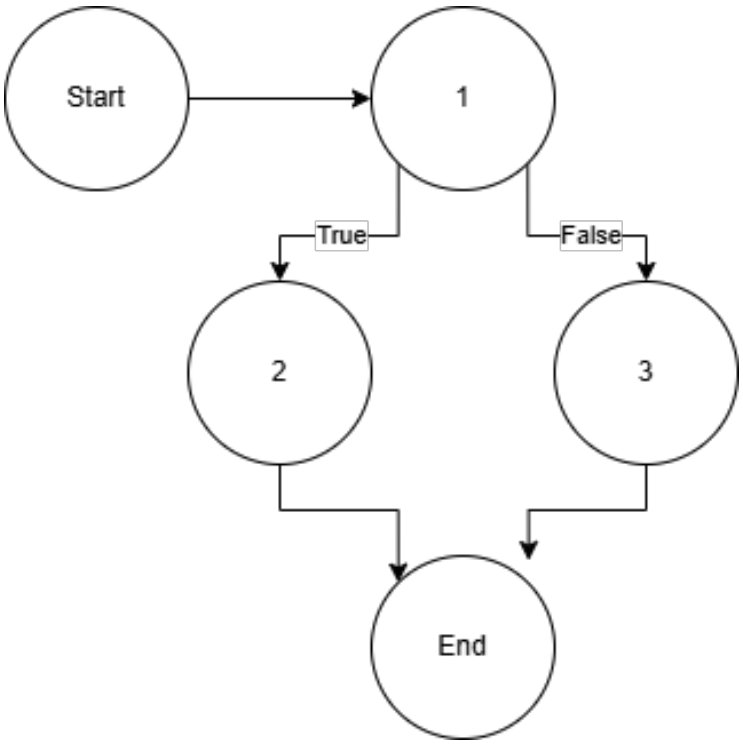

Figure 22: Control Flow Graph for isCharConstant

12 Method: isNumConstant

```
303   static boolean is_num_constant(String str)
304   {
305     int i=1;
306
307     if ( Character.isDigit(str.charAt(index: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   }
```

Figure 23: Code Snippet for isNumConstant

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	

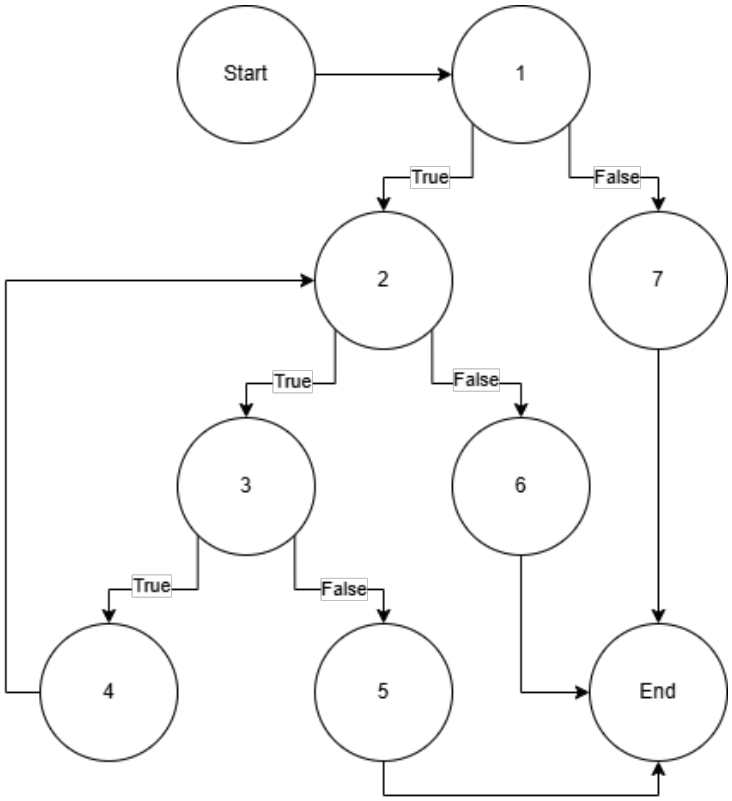

Figure 24: Control Flow Graph for isNumConstant

13 Method: isStrConstant

```
327   static boolean is_str_constant(String str)
328   {
329     int i=1;
330
331     if ( str.charAt(index: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   }
```

Figure 25: Code Snippet for isStrConstant

Block Number	Lines	Entry	Exit	Function Calls
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	

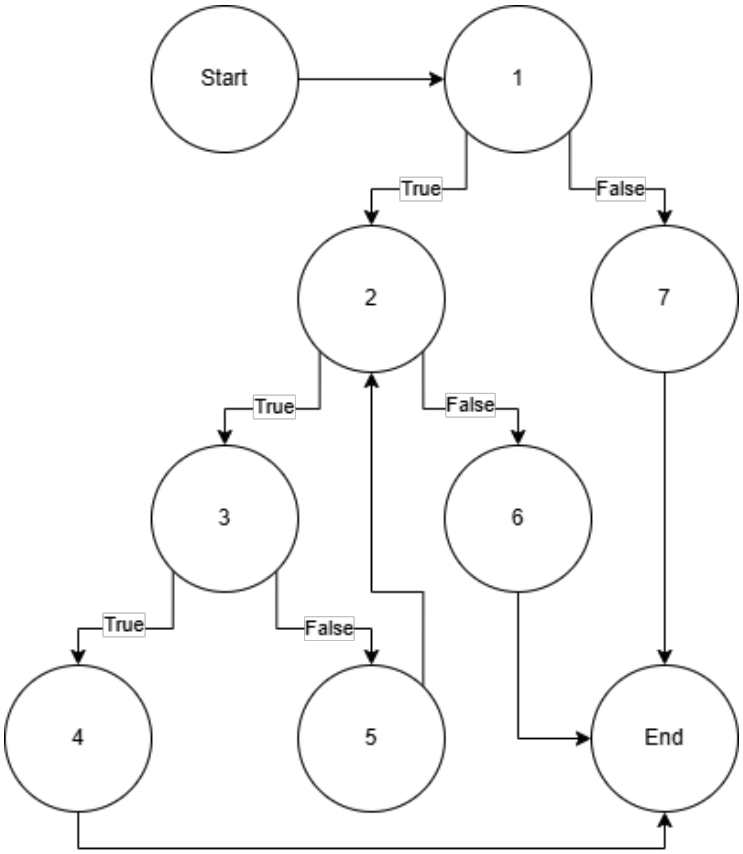

Figure 26: Control Flow Graph for isStrConstant

14 Method: isIdentifier

```
349   static boolean is_identifier(String str)
350   {
351     int i=1;
352
353     if ( Character.isLetter(str.charAt(index:0)) )
354       {
355         while(i < str.length() && str.charAt(i) !='\0' )   /* unti meet the end token sign */
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   }
```

Figure 27: Code Snippet for isIdentifier

Block Number	Lines	Entry	Exit	Function Calls
1	350, 352	350	352	
2	354	354	354	
3	355a, 355b	355a	355b	
4	357a, 357b	357a	357b	
5	359	359	359	
6	361	361	361	
7	364	364	364	

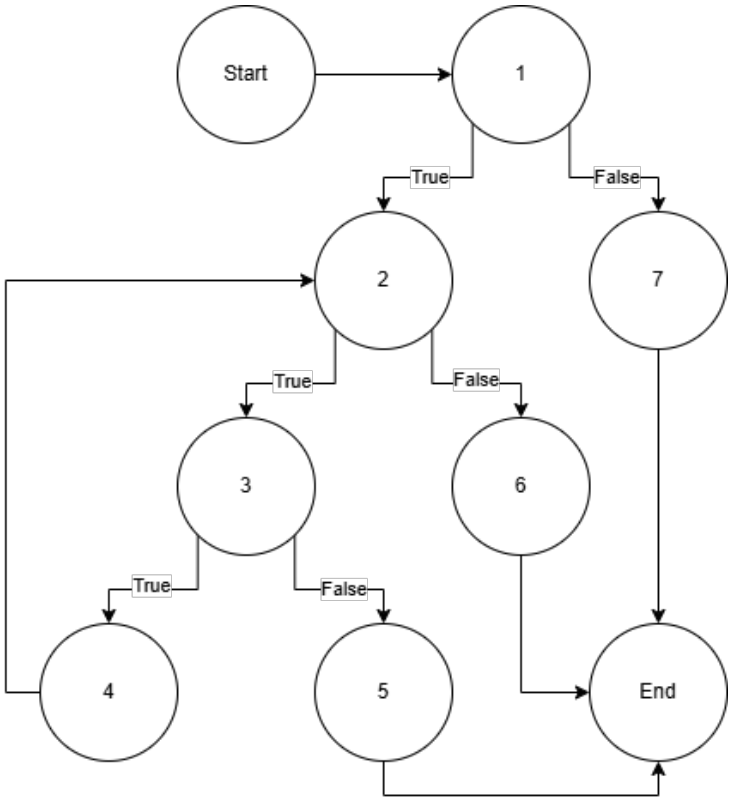

Figure 28: Control Flow Graph for isIdentifier

15 Method: printSpecSymbol

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

Figure 29: Code Snippet for printSpecSymbol

Block Number	Lines	Entry	Exit	Function Calls
1	377	377	377	
2	380, 381	380	381	
3	383	383	383	
4	386, 387	386	387	
5	389	389	389	
6	391, 392	391	392	
7	394	394	394	
8	396, 397	396	397	
9	400	400	400	
10	402, 403	402	403	
11	405	405	405	
12	408, 409	408	409	
13	412	412	412	
14	414, 415	414	415	

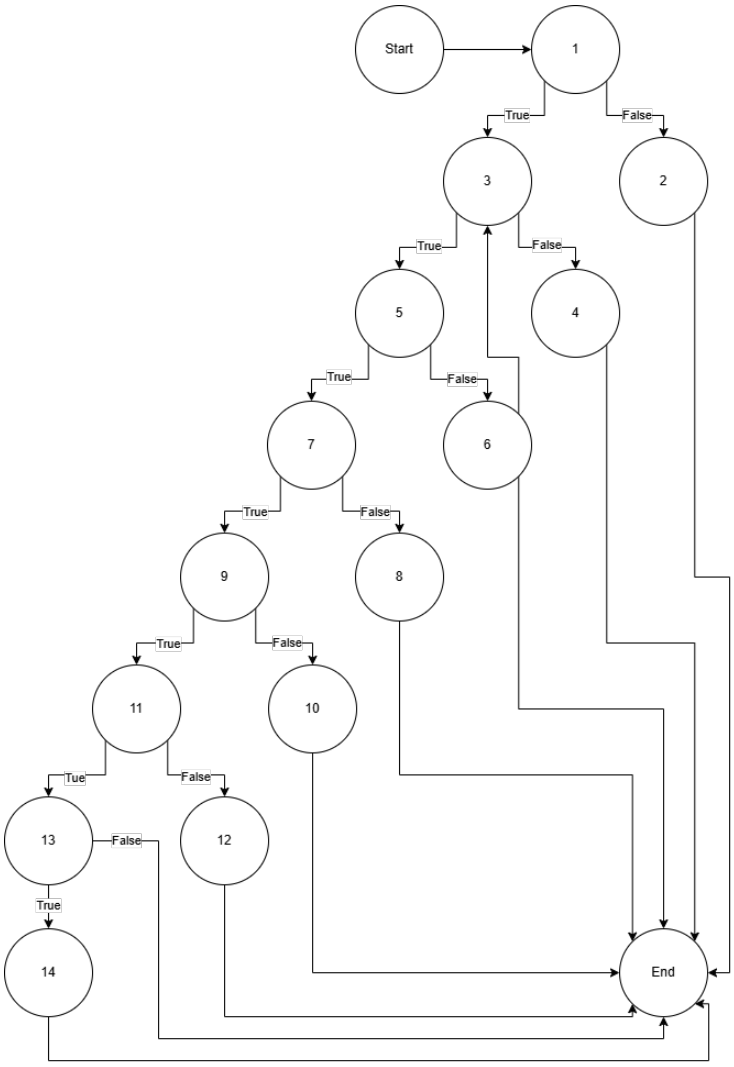

Figure 30: Control Flow Graph for printSpecSymbol

16 Method: isSpecSymbol

```
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     }
```

Figure 31: Code Snippet for isSpecSymbol

Block Number	Lines	Entry	Exit	Function Calls
1	426	426	426	
2	428	428	428	
3	430	430	430	
4	432	432	432	
5	434	434	434	
6	436	436	436	
7	438	438	438	
8	440	440	440	
9	442	442	442	
10	444	444	444	
11	446	446	446	
12	448	448	448	
13	450	450	450	
14	452	452	452	
15	454	454	454	

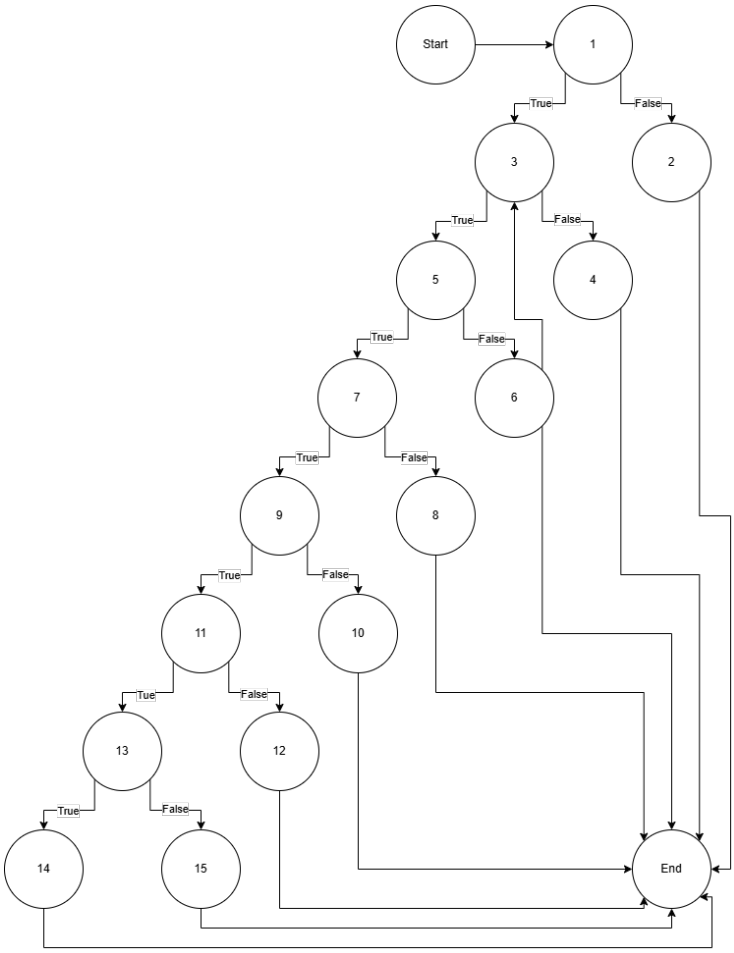

Figure 32: Control Flow Graph for isSpecSymbol

17 Method: 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(s:"Error! Please give the token stream\n");
466             System.exit(status: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(status:0);
478     }
```

Figure 33: Code Snippet for main

Block Number	Lines	Entry	Exit	Function Calls
1	458, 459	458	459	
2	460	460	460	
3	461	461	461	
4	462	462	462	
5	464, 465	464	465	
6	467	467	467	
7	468	468	468	open-token-stream
8	469	469	469	get-token
9	470	470	470	
10	471	471	471	print-token
11	472	472	472	get-token
12	476	476	476	

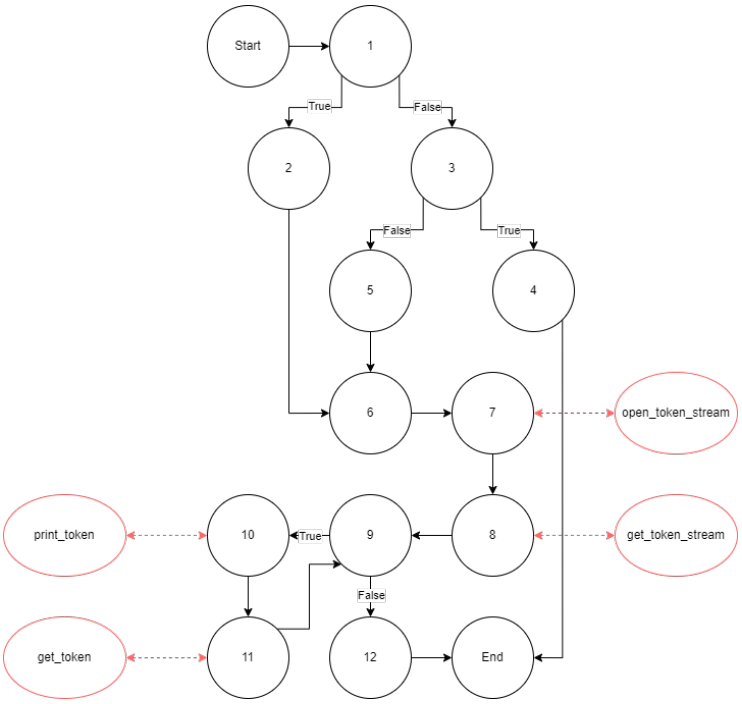

Figure 34: Control Flow Graph for main