TESTING PROJECT USING JUNIT

Jain, Mehul

Neurekar, Ranjeev

Table Of Contents

Open_token_stream

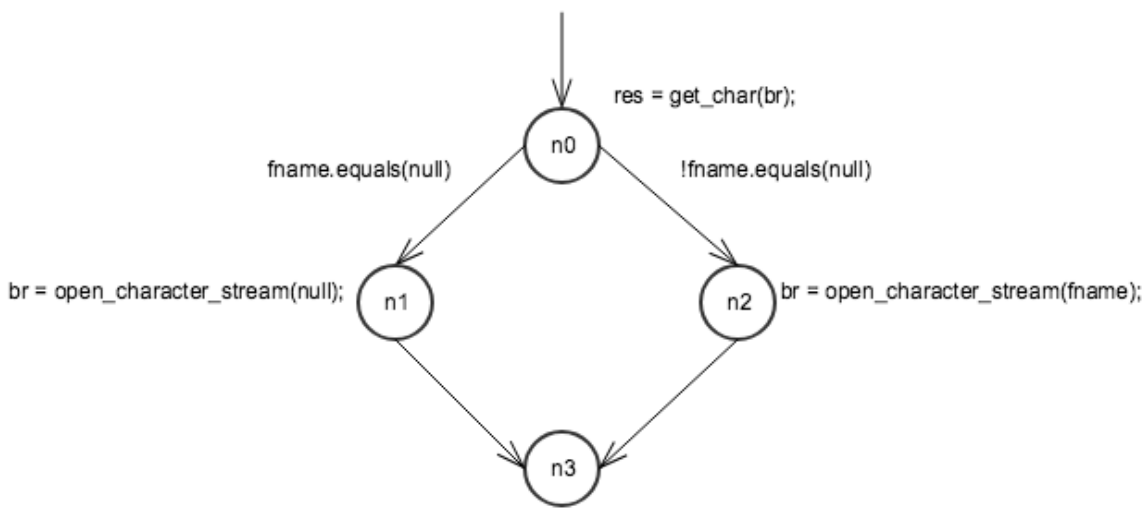
```
111     BufferedReader open_token_stream(String fname)
112     {
113         BufferedReader br;
            if (fname.equals(null))
115         {
116             br = open_character_stream(null);
117         }
118         else {
119             br = open_character_stream(fname);
120         }
121         return br;
122     }
```

Block	Lines	Entry	Exit
1	113,114	113	114
2	116	116	116
3	119	119	119
4	121	121	121

Test Cases:

Test 1: fname is null	Test 2: fname is not null

CFG:



Get_token

```
135         String get_token(BufferedReader br)
136         {
137             int i = 0, j;
138             int id = 0;
                int res = 0;
                char ch = '\0';
141             StringBuilder sb = new StringBuilder();
142             try {
143                 res = get_char(br);
144                 if (res == -1)
145                 {return null;}
146                 ch = (char) res;
147                 while (ch == ' ' || ch == '\n' || ch == '\r') /
148                 {res = get_char(br);
149                  ch = (char) res;}
150                 if (res == -1)
151                 {return null;}
152                 sb.append(ch);
153                 if (is_spec_symbol(ch) == true)
154                 {return sb.toString();}
155                 if (ch == '"')
156                 {id = 2;    /* prepare for string */}
157                 if (ch == 59)
158                 {id = 1;    /* prepare for comment */}
159                 res = get_char(br);
160                 if (res == -1)
161                 {unget_char(ch, br);
162                     return sb.toString();}
163                 ch = (char) res;
164                 while (is_token_end(id, res) == false)/* until
165                 { sb.append(ch);
166                     br.mark(4);
167                     res = get_char(br);
168                     if (res == -1)
169                     {break;}
170                     ch = (char) res;
171                 }
172                 if (res == -1) /* if end character is eof toke
173                 {unget_char(ch, br);
```

```
178                 {unget_char(ch, br);
179                     /* then put back this characte
180                     return sb.toString();
181             }
182                 if (id == 1) /* if end character is
183                 {
184                     sb.append(ch);
185                     return sb.toString();
186                 }
187                 if (id == 0 && ch == 59) /* when n
188                 {
189                     unget_char(ch, br);
190                     /* then put back this characte
191                     return sb.toString();
192                 }
193             }
194             catch (IOException e)
195             {
                    e.printStackTrace();
197             }
198
199             return sb.toString();
200             /* return nomal case token
201         }
202
```

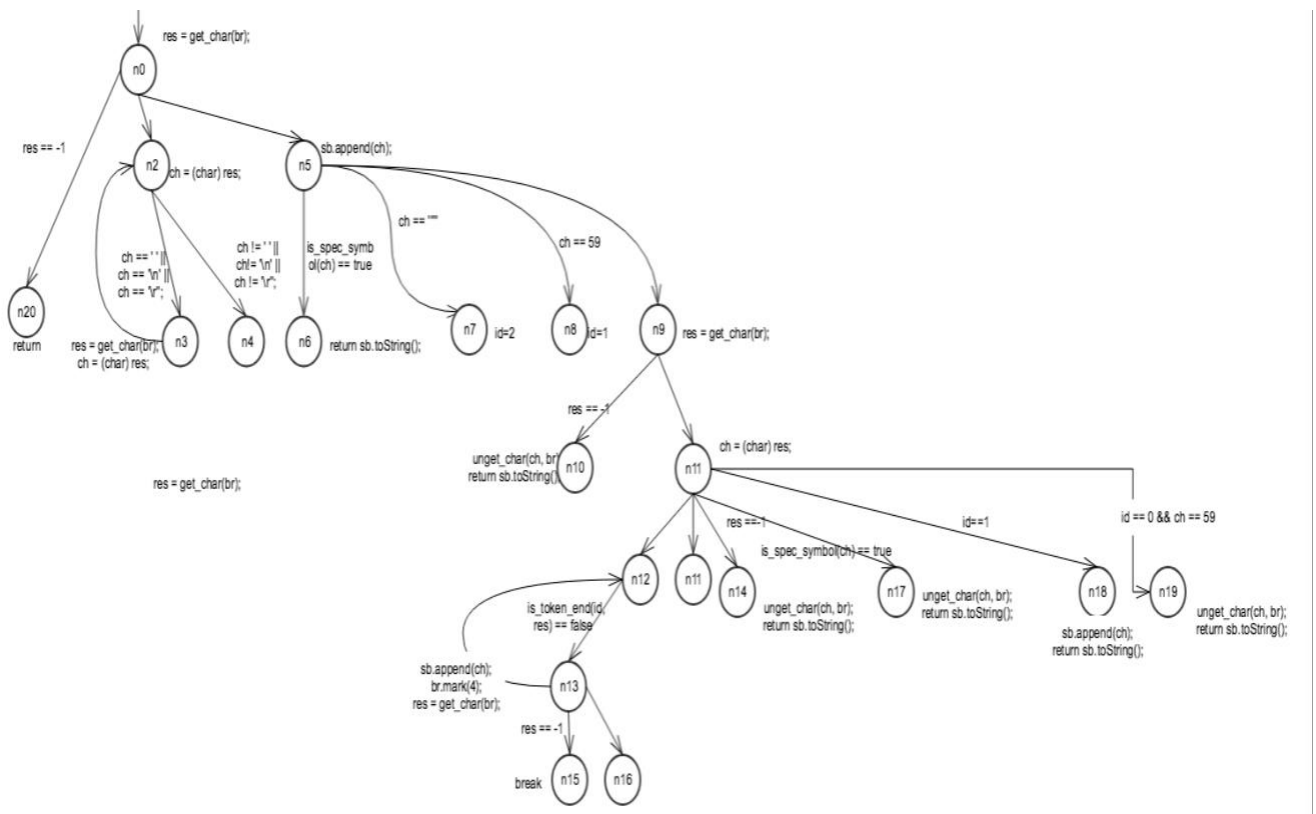
Block	Lines	Entry	Exit
1	137,138,139,140,141	137	141
2	143,144	143	144
3	145	145	145
4	146	146	146
5	147	147	147
6	148,149	148	149
7	150	150	150
8	151	151	151
9	152,153	152	153
10	154	154	154
11	155	155	155
12	156	156	156
13	157	157	157
14	158,159	158	159
15	160	160	160
16	161,162	161	162
17	163	163	163
18	164	164	164
19	165,166,167,168	165	168
20	169	169	169
21	170	170	170
22	172	172	172
23	173,175	173	175

24	182	182	182
25	184	184	185
26	187	187	187
27	189,191	189	191
28	194	194	194
29	196	196	196

Test Cases:

Test1: res is equal to -1
Test 2: br is null
Test 3: id is equal to zero and id is equal to 59
Test 4: is_Spec_symbol=true
Test 5: id is equal to one
Test 6: id is not 1,0,59
Test 7:res is not -1

CFG:



is_token_end

```
237     static boolean is_token_end(int str_com_id, int res)
238     {if (res == -1)
239         {
240             return (true); /* is eof token? */
241         }
242         char ch = (char) res;
243         if (str_com_id == 1) /* is string token */
244         {
            if (ch == '"' | ch == '\n' || ch == '\r') /* for str
246             {
247                 return true;
248             }
249             else
250             {
251                 return false;
252             }
253         }
254
255         if (str_com_id == 2) /* is comment token */
256         {
            if (ch == '\n' || ch == '\r' || ch == '\t') /* for c
258             {
259                 return true;
260             }
261             else
262             {
263                 return false;
264             }
265         }
266
267         if (is_spec_symbol(ch) == true)
268         {
269             return true; /* is special_symbol? */
270         }
            if (ch == ' ' || ch == '\n' || ch == '\r' || ch == 59)
272         {
273             return true;
274         }
275         return false;
```

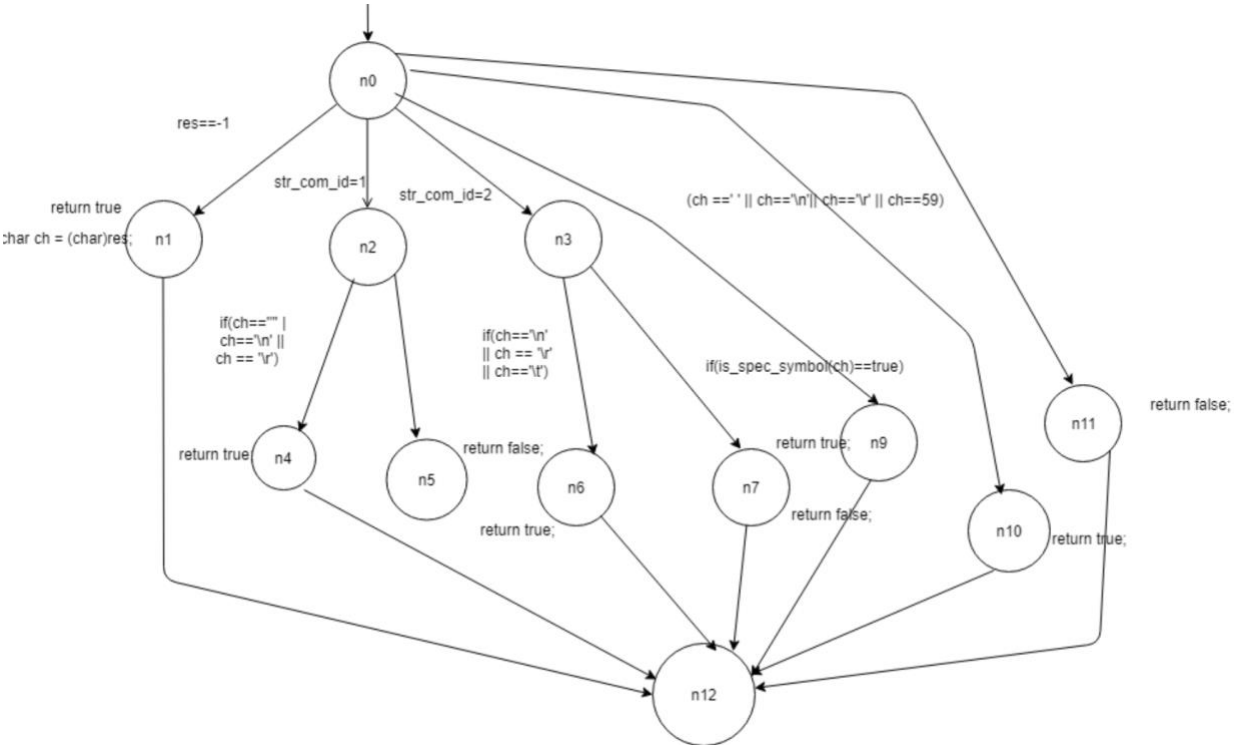
Block	Lines	Entry	Exit
1	238	238	238
2	240	240	240
3	242,243	242	243
4	245	245	245
5	247	247	247
6	251	251	251
7	255	255	255
8	257	257	257
9	259	259	259
10	263	263	263
11	267	267	267
12	269	269	269
13	271	271	271
14	273	273	273

15	275	275	275

Test Cases:

Test 1: res is equal to -1
Test 2: str_com_id is equal to 1 and ch is ‘’’,” \n”, ”\r”
Test 3: str_com_id is equal to 1 and ch is not equal to ‘’’,” \n”, ”\r”
Test 4: str_com_id is equal to 2 and ch is ‘\t’,”\n”, ”\r”
Test 5: str_com_id is equal to 2 and ch is not equal to ‘\t’,” \n”, ”\r”
Test 6: ch is a special symbol
Test 7: ch is ‘ ‘, ‘\n’,’\r’,59
Test 8: Not any of these above cases

CFG:



Token_type

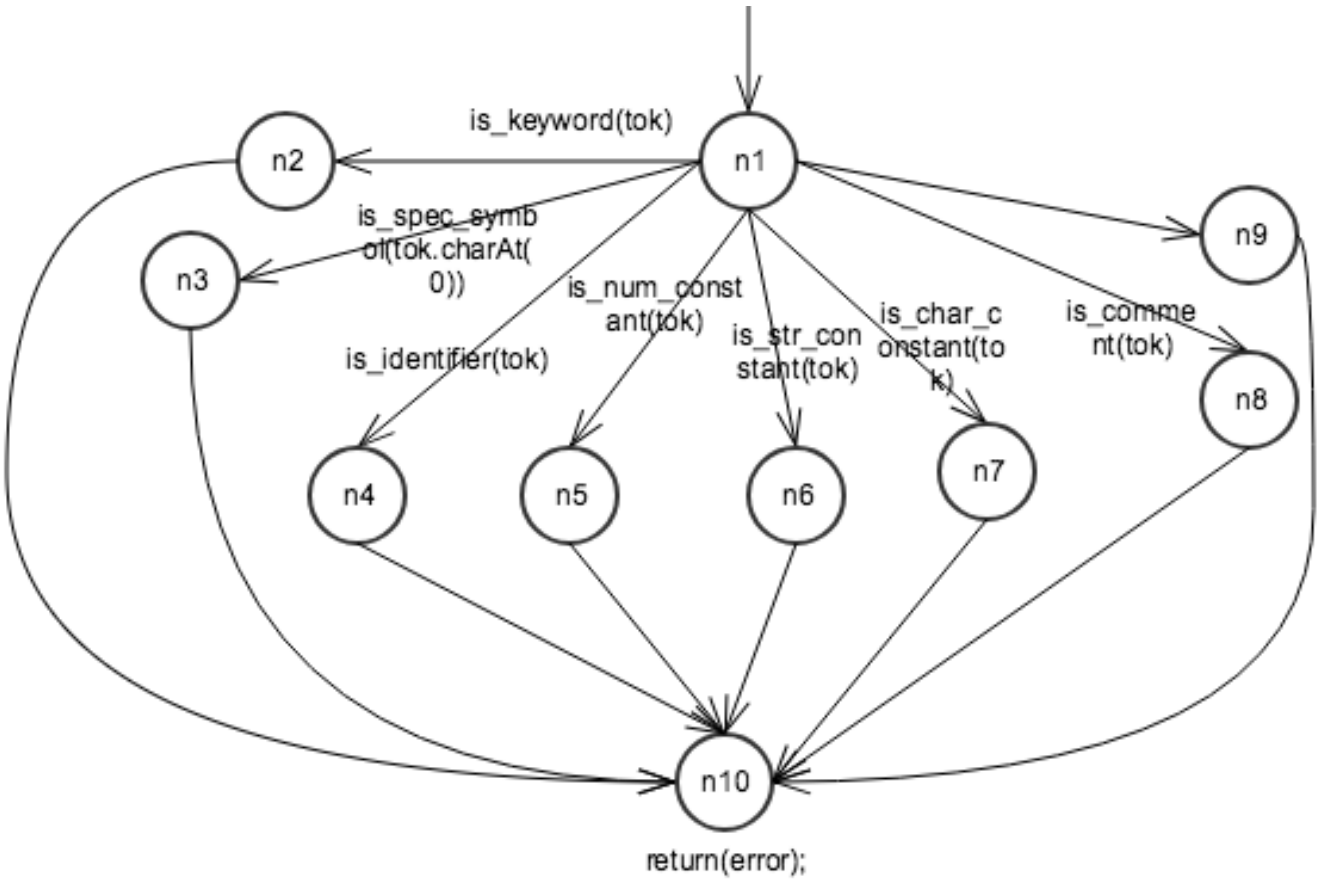
```
290     static int token_type(String tok)
291     {
292         if (is_keyword(tok))
293         {
294             return (keyword);
295         }
296         if (is_spec_symbol(tok.charAt(0)))
297         {
298             return (spec_symbol);
299         }
300         if (is_identifier(tok))
301         {
302             return (identifier);
303         }
304         if (is_num_constant(tok))
305         {
306             return (num_constant);
307         }
308         if (is_str_constant(tok))
309         {
310             return (str_constant);
311         }
312         if (is_char_constant(tok))
313         {
314             return (char_constant);
315         }
316         if (is_comment(tok))
317         {
318             return (comment);
319         }
320         return (error);
```

Block	Lines	Entry	Exit
1	292	292	292
2	294	294	294
3	296	296	296
4	298	298	298
5	300	300	300
6	302	302	302
7	304	304	304
8	306	306	306
9	308	308	308
10	310	310	310
11	312	312	312
12	314	314	314
13	316	316	316
14	318	318	318
15	320	320	320

Test Cases:

Test 1: token is a keyword
Test 2: token is a special symbol
Test 3: token is an identifier
Test 4: token is a number constant
Test 5: token is a string constant
Test 6: token is a character constant
Test 7: token is a comment
Test 8: token doesn't satisfy any of the above conditions

CFG:



Print_token

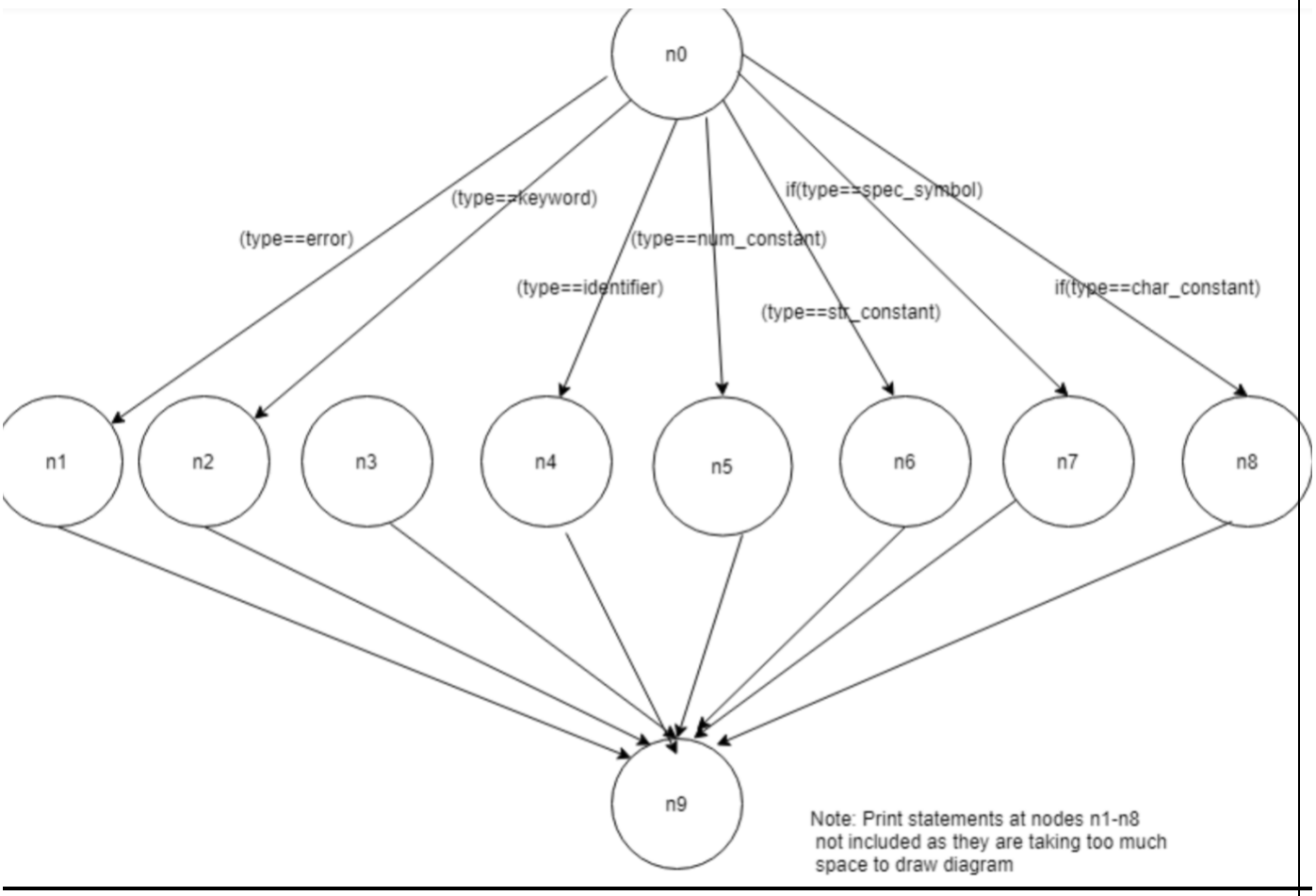
```
332     void print_token(String tok)
333     {
334         int type;
335         type = token_type(tok);
336         if (type == error)
337         {
338             System.out.print("error,\"" + tok + "\".\n");
339         }
340
341         if (type == keyword)
342         {
343             System.out.print("keyword,\"" + tok + "\".\n");
344         }
345
346         if (type == spec_symbol)
347         {
348             print_spec_symbol(tok);
349         }
350         if (type == identifier)
351         {
352             System.out.print("identifier,\"" + tok + "\".\n");
353         }
354         if (type == num_constant)
355         {
356             System.out.print("numeric," + tok + ".\n");
357         }
358         if (type == str_constant)
359         {
360             System.out.print("string," + tok + ".\n");
361         }
362         if (type == char_constant)
363         {
364             System.out.print("character,\"" + tok.charAt(1) + "\".\n");
365         }
366
367     }
```

Block	Lines	Entry	Exit
1	334,335,336	334	336
2	338	338	338
3	341	341	341
4	343	343	343
5	346	346	346
6	348	348	348
7	350	350	350
8	352	352	352
9	354	354	354
10	356	356	356
11	358	358	358
12	360	360	360
13	362	362	362
14	364	364	364

Test Cases:

Test 1: if type is 0
Test 2: if type is 1
Test 3: if type is 2
Test 4: if type is 3
Test 5: if type is 41
Test 6: if type is 42
Test 7: if type is 43
Test 8: if type is null

CFG:



Is_comment

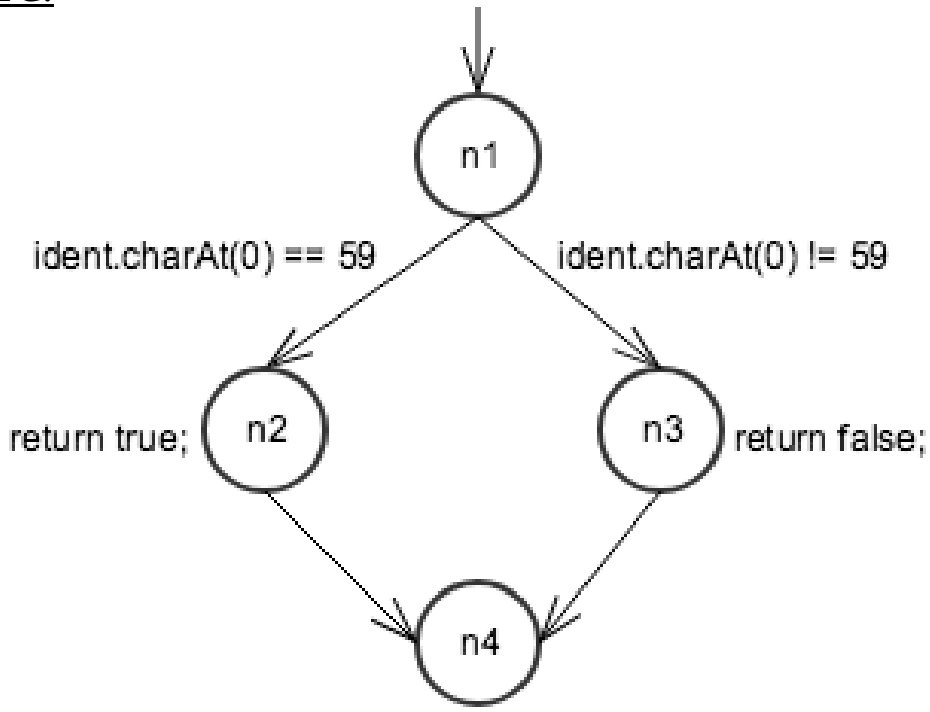
```
379     static boolean is_comment(String ident) {
            if (ident.charAt(0) == 59) /* the char i
381         {
382             return true;
383         }
384         else
385         {
386             return false;
387         }
388     }
```

Block	Lines	Entry	Exit
1	380	380	380
2	382	382	382
3	386	386	386

Test Cases:

Test 1: String is one character
Test 2: String with 10 characters
Test 3: Empty String
Test 4: Null String
Test 5: String with character at zero position=59

CFG:



Is_keyword

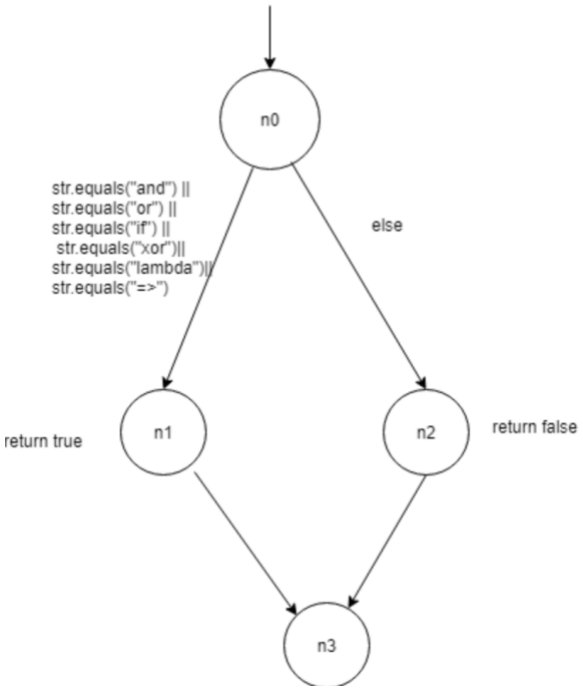
```
399     static boolean is_keyword(String str)
400     {
            if (str.equals("and") || str.equals("or") || str.equals("if")
402                 || str.equals("xor") || str.equals("lambda") || str.equals("=>"))
403         {
404             return true;
405         }
406         else
407         {
408             return false;
409         }
410     }
```

Block	Lines	Entry	Exit
1	401,402	401	402
2	404	404	404
3	408	408	408

Test Cases:

Test 1: str is null
Test 2: str is and, or, if, xor, lambda, =>
Test 3:str is not covered in test 1 or test 2

CFG:



Is_char_constant

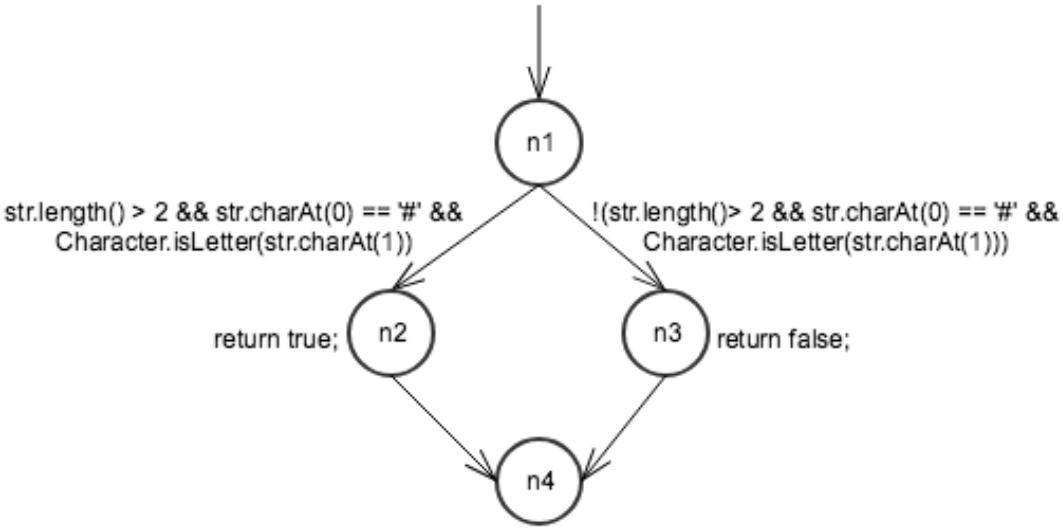
```
421     static boolean is_char_constant(String str)
422     {
            if (str.length() > 2 && str.charAt(0) == '#' && Character.isLetter(str.charAt(1)))
424         {
425             return true;
426         }
427         else
428         {
429             return false;
430         }
```

Block	Lines	Entry	Exit
1	423	423	423
2	425	425	425
3	429	429	429

Test Cases:

Test 1: str is null
Test 2: length of str is more than 2 and character at zero position is ‘#’ and character at one position is a letter
Test 3: length of str is less than two
Test 4: character at zero position is not #
Test 5: character at one position is not a letter

CFG



Is_num_constant

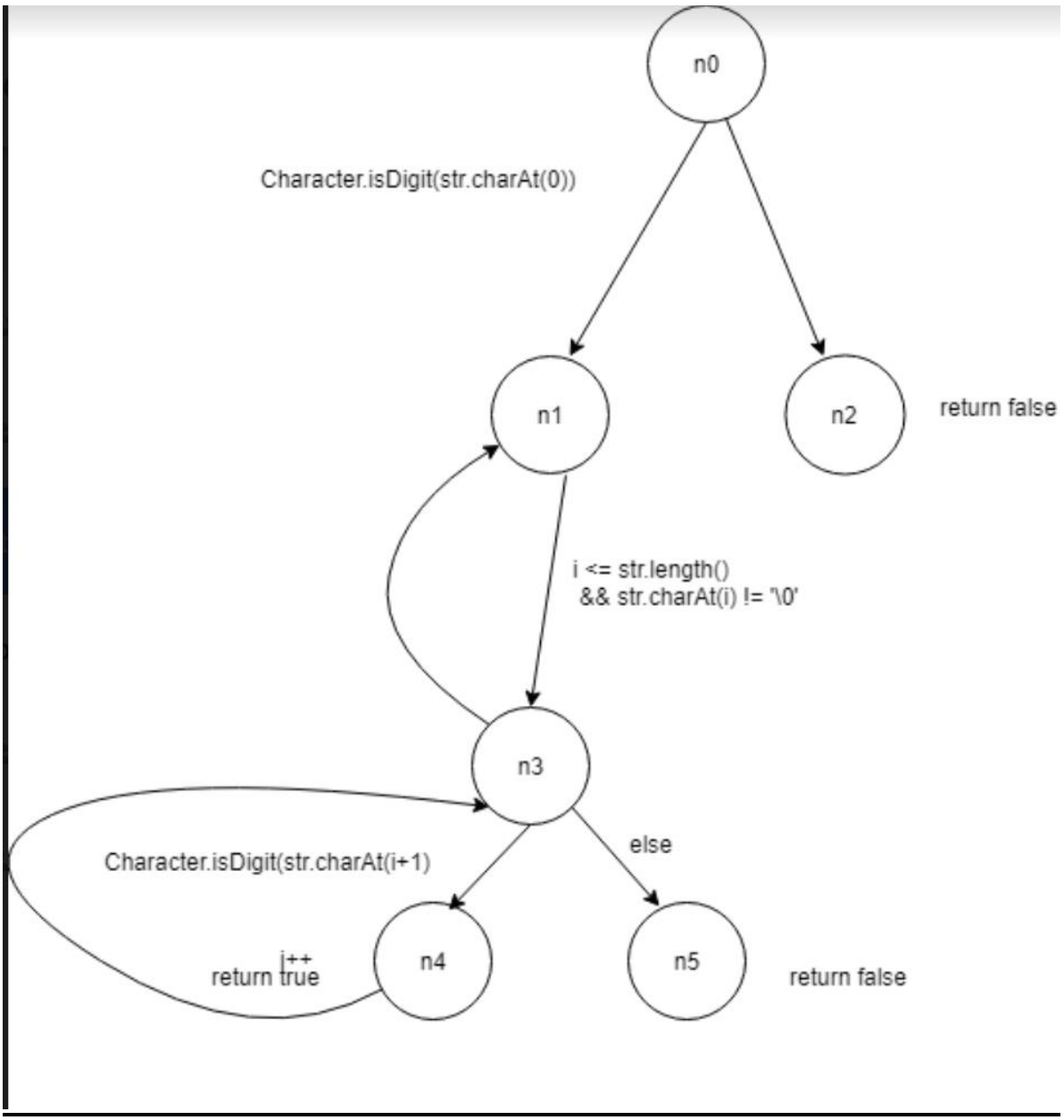
```
442     static boolean is_num_constant(String str)
443     {
444         int i = 1;
445
446         if (Character.isDigit(str.charAt(0)))
447         {
448             while (i <= str.length() && str.charAt(i) != '\0')
449             {
450                 if (Character.isDigit(str.charAt(i + 1)))
451                 {
452                     i++;
453                 }
454                 else
455                 {
456                     return false;
457                 }
458             }
459             /* end WHILE */
460             return true;
461         }
462         else
463         {
464             return false;               /* other return FALSE *
465         }
466     }
```

Block	Lines	Entry	Exit
1	444,446	444	446
2	448	448	448
3	450	450	450
4	452	452	452
5	456	456	456
6	460	460	460
7	464	464	464

Test Cases:

Test 1: str is null
Test 2: position zero of str is a digit and for every consequent position
Test 3: position zero of str is a digit and is not a digit for the next positions
Test 4: position zero of str is not a digit

CFG:


n0
Character.isDigit(str.charAt(0))
n1
n2
return false
i <= str.length()
&& str.charAt(i) != '\0'
n3
else
Character.isDigit(str.charAt(i+1)
i++
return true
n4
n5
return false

Is_str_constant

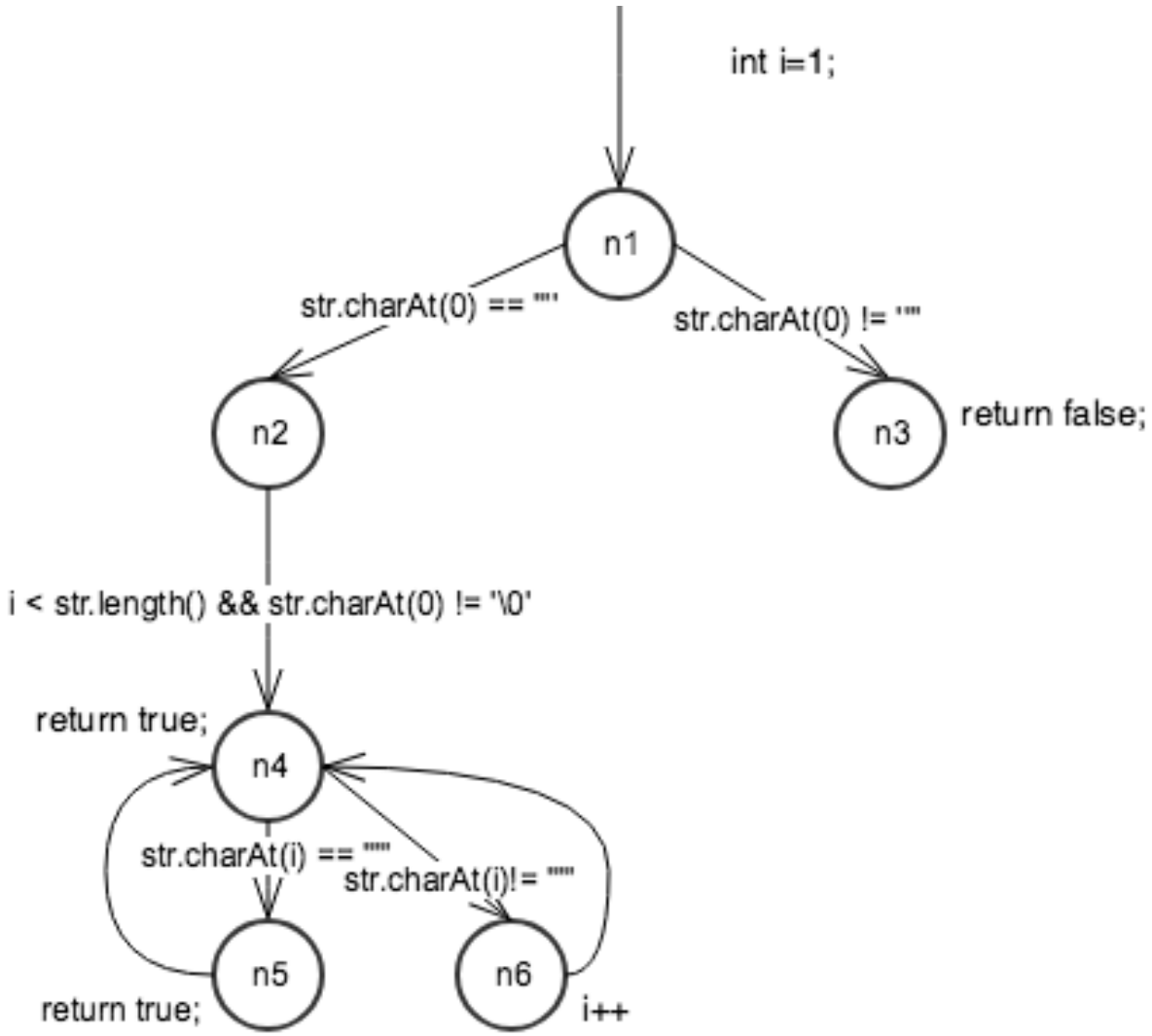
```
477     static boolean is_str_constant(String str)
478     {
479         int i = 1;
480
481         if (str.charAt(0) == '"')
482         {
483             while (i < str.length() && str.charAt(0) != '\0') /*
484             {
485                 if (str.charAt(i) == '"')
486                 {
487                     return true;        /* meet the second '"'
488                 }
489                 else
490                 {
491                     i++;
492                 }
493             }
494             /* end WHILE */
495             return true;
496         }
497         else
498         {
499             return false;       /* other return FALSE */
500         }
501     }
```

Block	Lines	Entry	Exit
1	479,480	479	480
2	483	483	483
3	485	485	485
4	487	487	487
5	491	491	491
6	499	499	495
7	499	499	499

Test Cases:

Test 1:str is null
Test 2: position zero of str is a “”” and for every consequent position
Test 3: position zero of str is a “”” and not or every consequent position
Test 4: position zero of str is not “””

CFG:


int i=1;
n1
str.charAt(0) == ""
str.charAt(0) != ""
n2
n3
return false;
i < str.length() && str.charAt(0) != '\0'
return true;
n4
str.charAt(i) == ""
str.charAt(i)!= ""
n5
n6
return true;
i++

Is_identifier

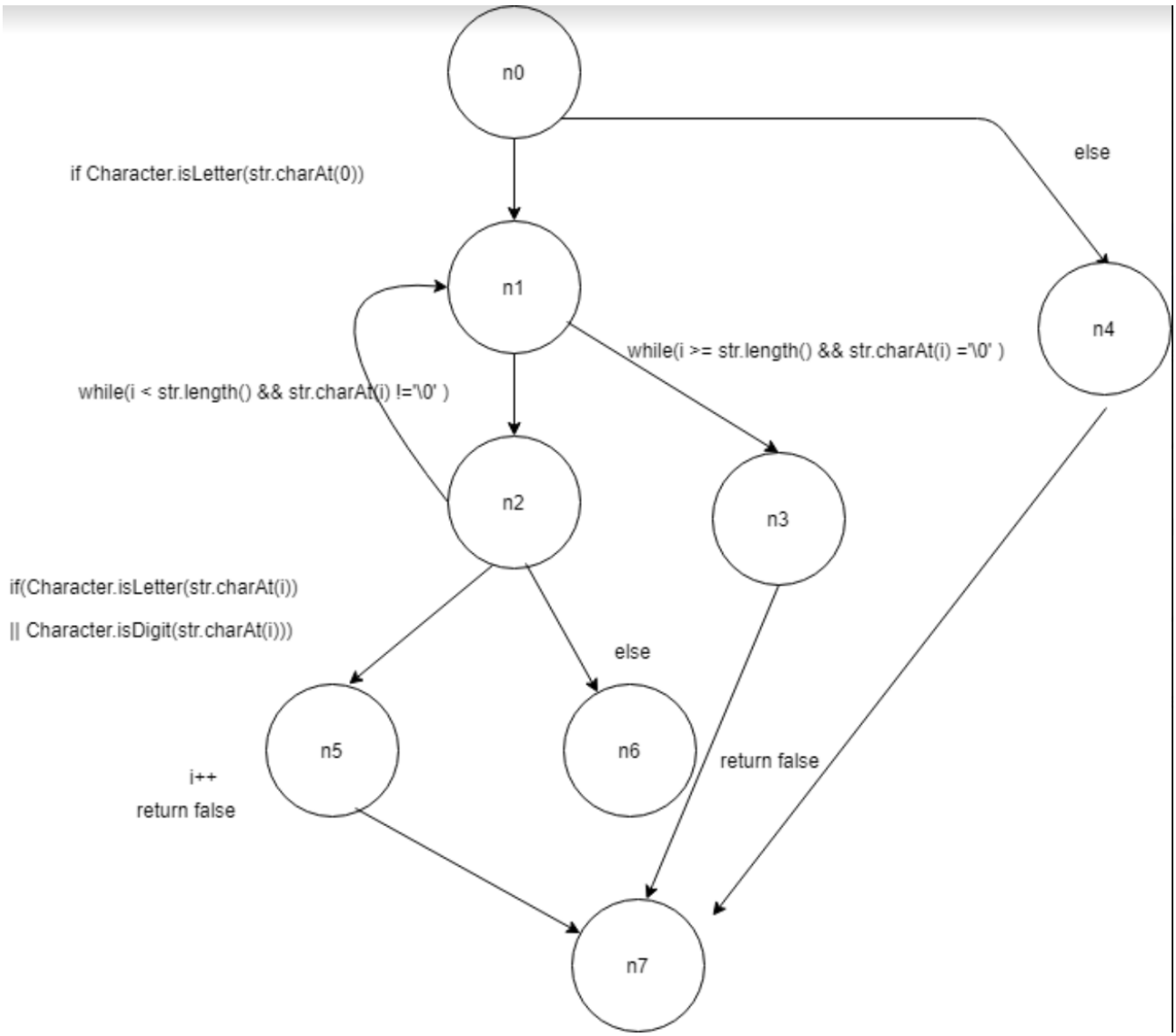
```
512     static boolean is_identifier(String str)
513     {
514         int i = 1;
515
516         if (Character.isLetter(str.charAt(0)))
517         {
518             while (i < str.length() && str.charAt(i) != '\0') /* unti meet the end token sig
519             {
520                 if (Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
521                 {
522                     i++;
523                 }
524                 else
525                 {
526                     return false;
527                 }
528             }
529             /* end WHILE */
530             return false;
531         }
532         else
533         {
534             return true;
535         }
536     }
```

Block	Lines	Entry	Exit
1	514,516	514	516
2	518	518	518
3	520	520	520
4	522	522	522
5	526	526	526
6	530	530	530
7	534	534	534

Test Cases:

Test 1:str is null
Test 2: string length is zero
Test 3: character at position i is a letter or a digit.
Test 4: character at position i is a special symbol
Test 5: character at position I is not a letter

CFG:


n0
if Character.isLetter(str.charAt(0))
else
n1
n4
while(i >= str.length() && str.charAt(i) ='\0')
while(i < str.length() && str.charAt(i) !='\0')
n2
n3
if(Character.isLetter(str.charAt(i))
|| Character.isDigit(str.charAt(i)))
else
n5
n6
return false
i++
return false
n7

print_spec_symbol

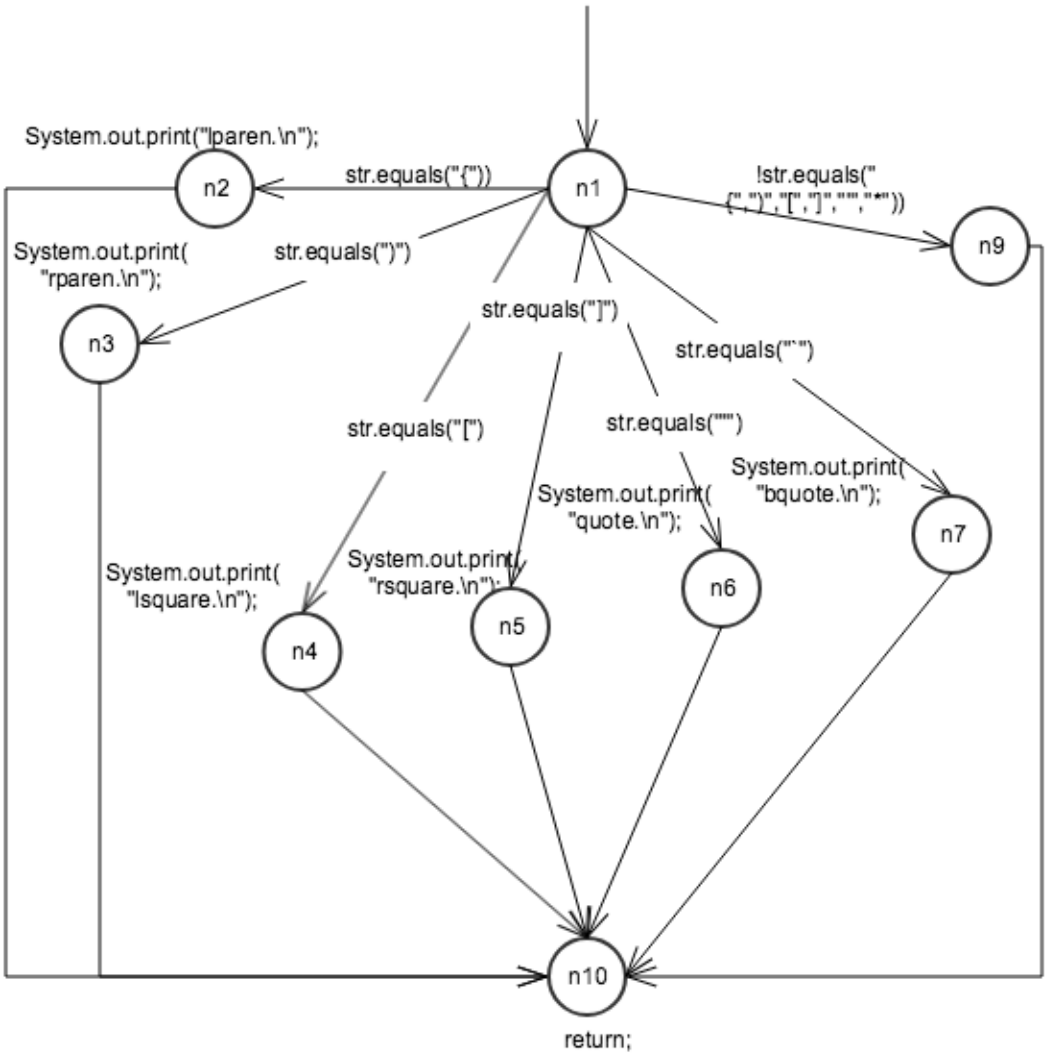
```
562     static void print_spec_symbol(String str)
563     {
564         if (str.equals("{"))
565         {
566
567             System.out.print("lparen.\n");
568             return;
569         }
570         if (str.equals(")"))
571         {
572
573             System.out.print("rparen.\n");
574             return;
575         }
576         if (str.equals("["))
577         {
578             System.out.print("lsquare.\n");
579             return;
580         }
581         if (str.equals("]"))
582         {
583
584             System.out.print("rsquare.\n");
585             return;
586         }
587         if (str.equals("'"))
588         {
589             System.out.print("quote.\n");
590             return;
591         }
592         if (str.equals("`"))
593         {
594
595             System.out.print("bquote.\n");
596             return;
597         }
598
599     }
```

Block	Lines	Entry	Exit
1	564	564	564
2	567,568	567	568
3	570	570	570
4	573,574	573	574
5	576	576	576
6	578,579	578	579
7	581	581	581
8	584,585	584	585
9	587	587	587
10	589,590	589	590
11	592	592	592
12	595,596	595	596

Test Cases:

Test 1: str is NULL
Test 2: str is zero
Test 3: str is ‘{’
Test 4: str is ‘)’
Test 5: str is ‘[’
Test 6: str is ‘]’
Test7: str is ‘’’
Test 8: str is ‘`’

CFG:



Is_spec_symbol

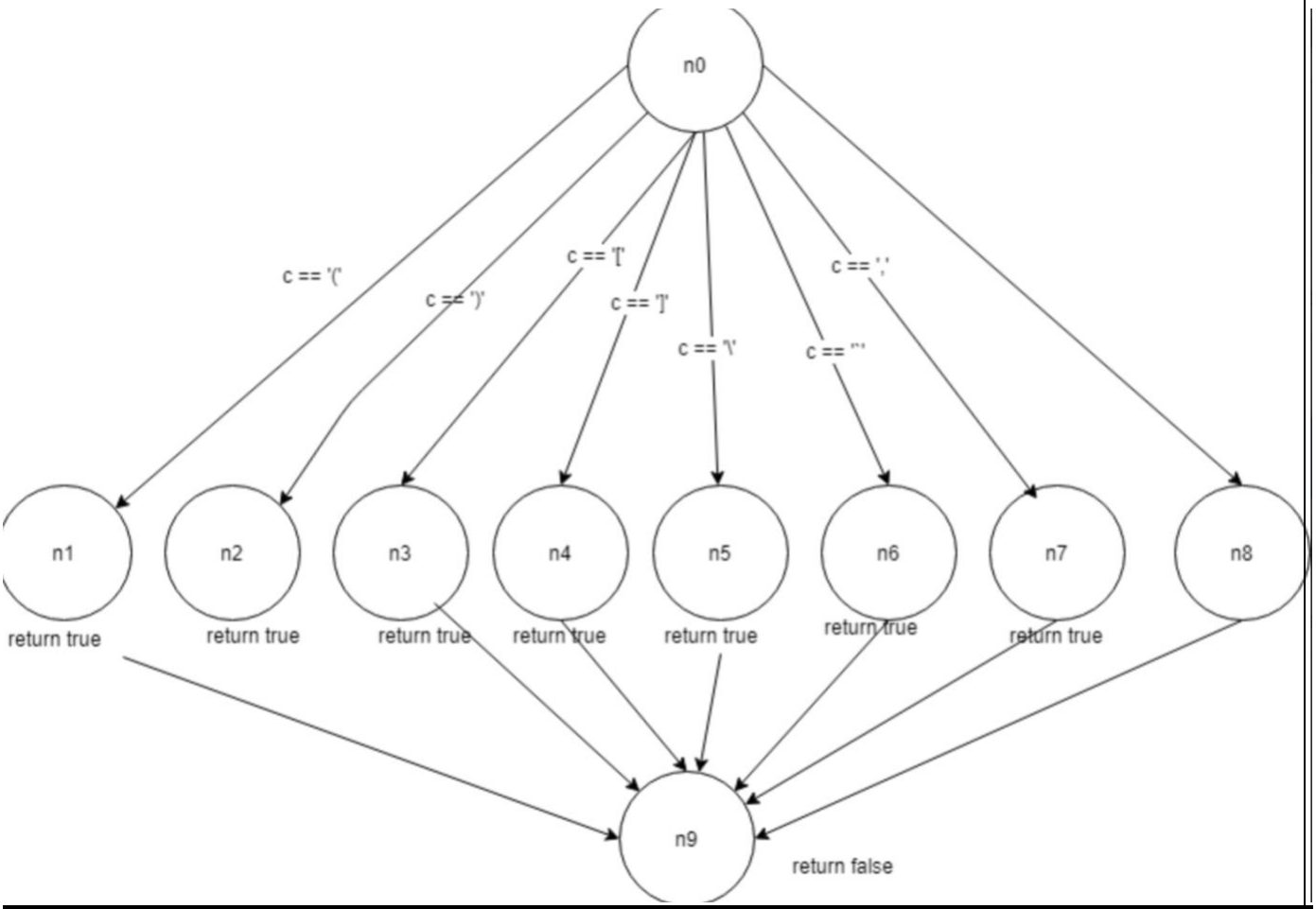
```
610     static boolean is_spec_symbol(char c)
611     {
612         if (c == '(')
613         {
614             return true;
615         }
616         if (c == ')')
617         {
618             return true;
619         }
620         if (c == '[')
621         {
622             return true;
623         }
624         if (c == ']')
625         {
626             return true;
627         }
628         if (c == '\'')
629         {
630             return true;
631         }
632         if (c == '`')
633         {
634             return true;
635         }
            if (c == ',')
637         {
638             return true;
639         }
640         return false;
641         /* others return FALSE */
642     }
```

Block	Lines	Entry	Exit
1	612	612	612
2	614	614	614
3	616	616	616
4	618	618	618
5	620	620	620
6	622	622	622
7	624	624	624
8	626	626	626
9	628	628	628
10	630	630	630
11	632	632	632
12	634	634	634
13	636	636	636
14	638	638	638
15	640	640	640

Test Cases:

Test 1: c is ‘(’
Test 2: c is ‘)’
Test 3: c is ‘[’
Test 4: c is ‘]’
Test 5: c is ‘\’
Test 6: c is ‘`’
Test 7: c is’,’

CFG



main

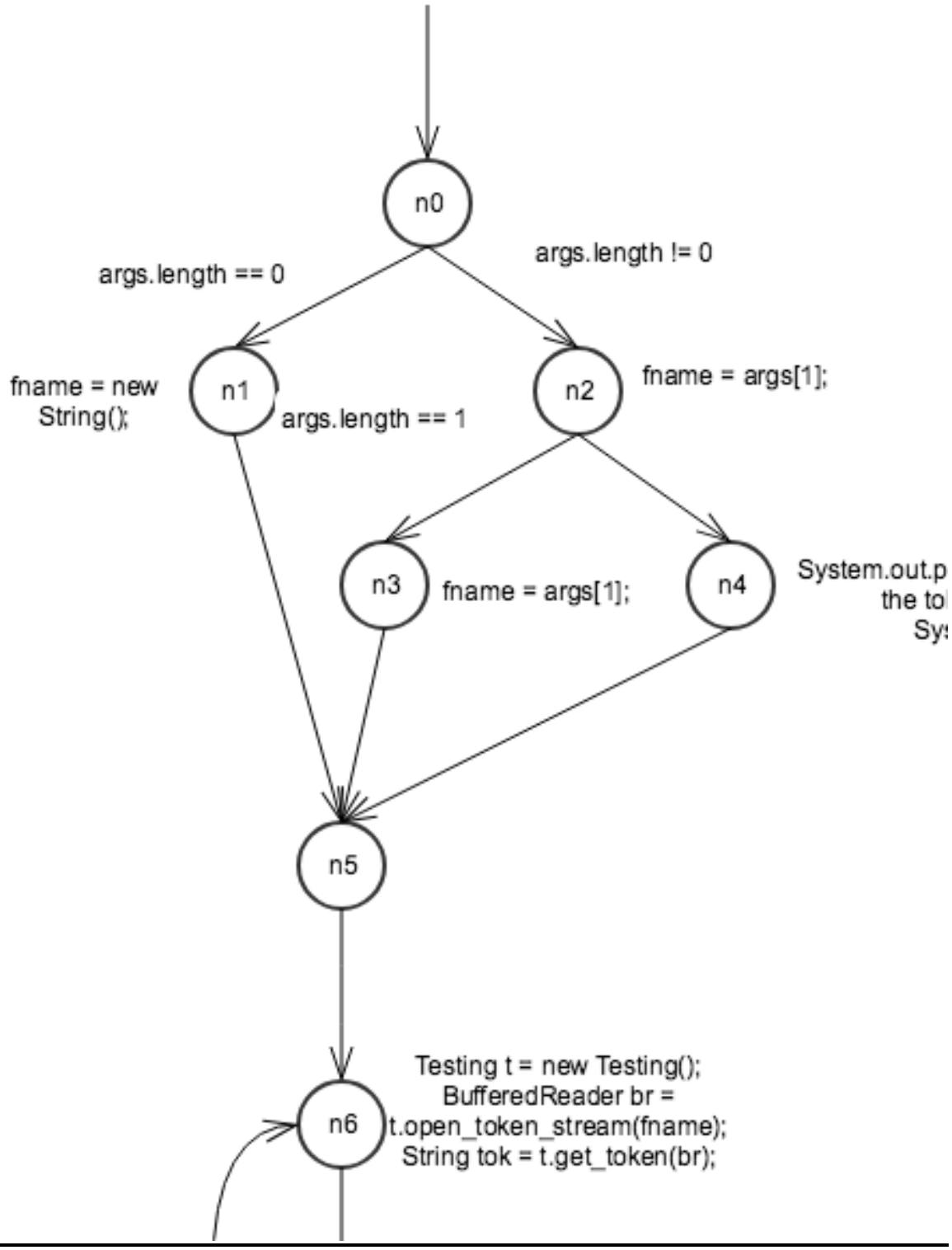
```
489
490     public static void main(String[] args) throws IOException {
491         String fname = null;
492         if (args.length == 0)
493         {   /* if not given filename,take as '"'' */
494             fname = new String();
495         } else if (args.length == 1)
496         {
497             fname = args[1];
498         } else
499         {
500             System.out.print("Error!,please give the token stream\n");
501             System.exit(0);
502         }
503         junit t = new junit();
504         BufferedReader br = t.open_token_stream(fname); /* open token stream */
505         String tok = t.get_token(br);
506         while (tok != null)
507         {   /* take one token each time until eof */
508             t.print_token(tok);
509             tok = t.get_token(br);
510         }
511
512         System.exit(0);
513     }
514
```

Block	Lines	Entry	Exit
1	491,492	491	492
2	494	494	494
3	495	495	495
4	497	497	497
5	500,501	500	501
6	503,504,505	503	505
7	506	506	506
8	508,509	508	509
9	512	512	512

Test Cases:

Test 1: length of args is 0
Test 2: length of args is 1
Test 3: br is null
Test 4: br contains a file
Test5: tok is null
Test 6: token is not null

CFG:


n0
args.length == 0
args.length != 0
fname = new String();
n1
n2
fname = args[1];
args.length == 1
n3
fname = args[1];
n4
System.out.p
the to
Sy
n5
n6
Testing t = new Testing();
BufferedReader br =
t.open_token_stream(fname);
String tok = t.get_token(br);