Control Flow Graph (CFG) Analysis

Document Purpose: Method-Level CFG Generation Excluding Catch Blocks

By: Cesar Frayre

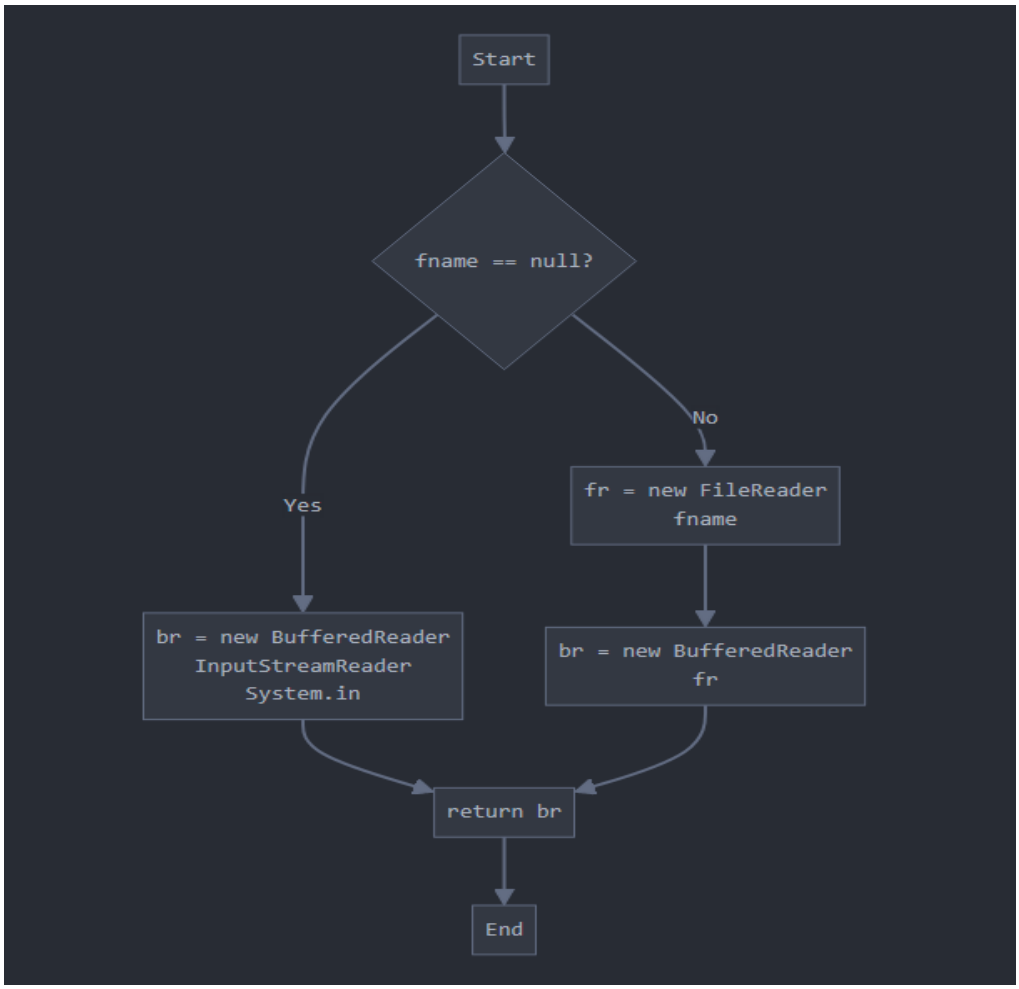
Date: November 25, 2024

---

1. BufferedReader open_character_stream(String fname)

```
23      BufferedReader open_character_stream(String fname) {  2 usages
24          BufferedReader br = null;
25          if (fname == null) {
26              br = new BufferedReader(new InputStreamReader(System.in));
27          } else {
28              try {
29                  FileReader fr = new FileReader(fname);
30                  br = new BufferedReader(fr);
31              } catch (FileNotFoundException e) {
32                  System.out.print("The file " + fname +" doesn't exists\n");
33                  e.printStackTrace();
34              }
35          }
36
37          return br;
38      }
```

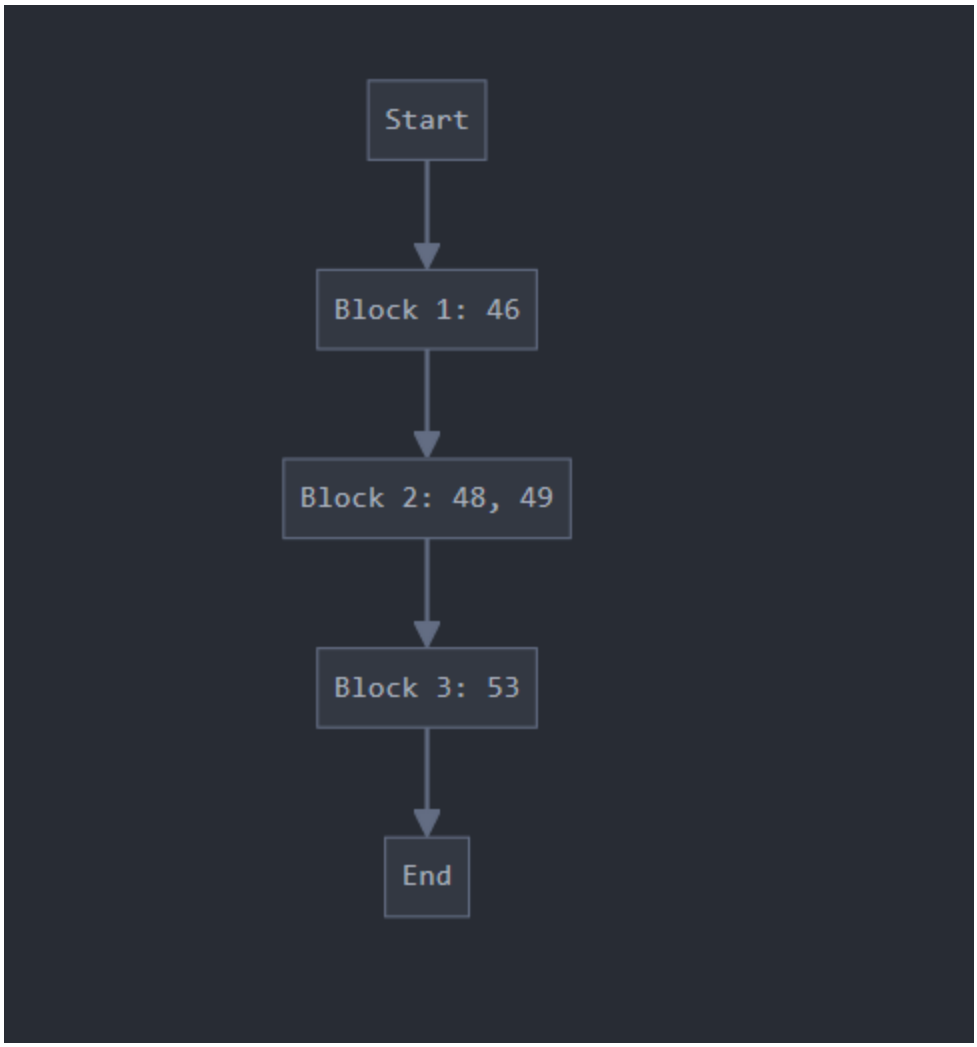
Block Number	Lines Number
1	24, 25
2	26
3	29, 30
4	37



2. Int get_char(BufferedReader br)

```
45  @   int get_char(BufferedReader br){  4 usages
46          int ch = 0;
47          try {
48              br.mark( readAheadLimit: 4);
49              ch= br.read();
50          } catch (IOException e) {
51              e.printStackTrace();
52          }
53          return ch;
54      }
55
56      /*****************************************************/
```

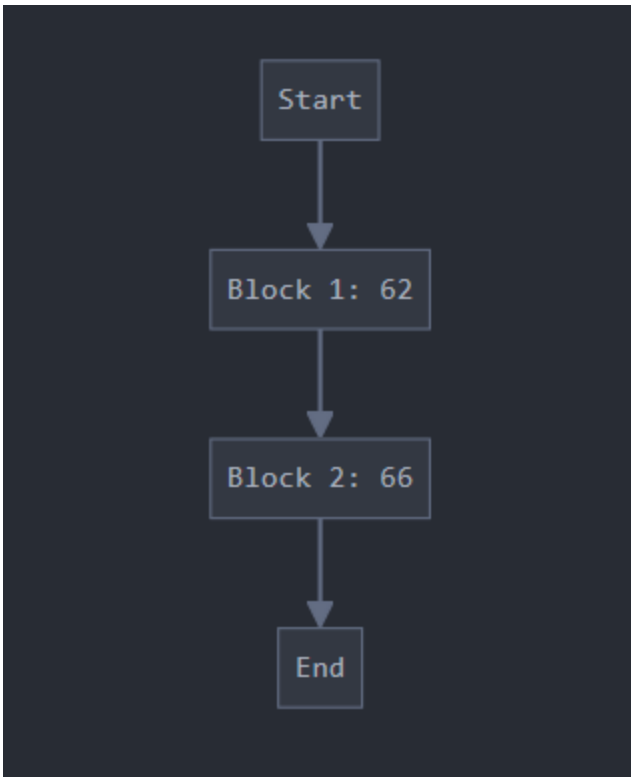
Block Number	Lines Number
1	46
2	48,49
3	53



3. Char unget_char(int ch, BufferedReader br)

```
char unget_char (int ch,BufferedReader br) {  4 usages
    try {
        br.reset();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return 0;
}
```

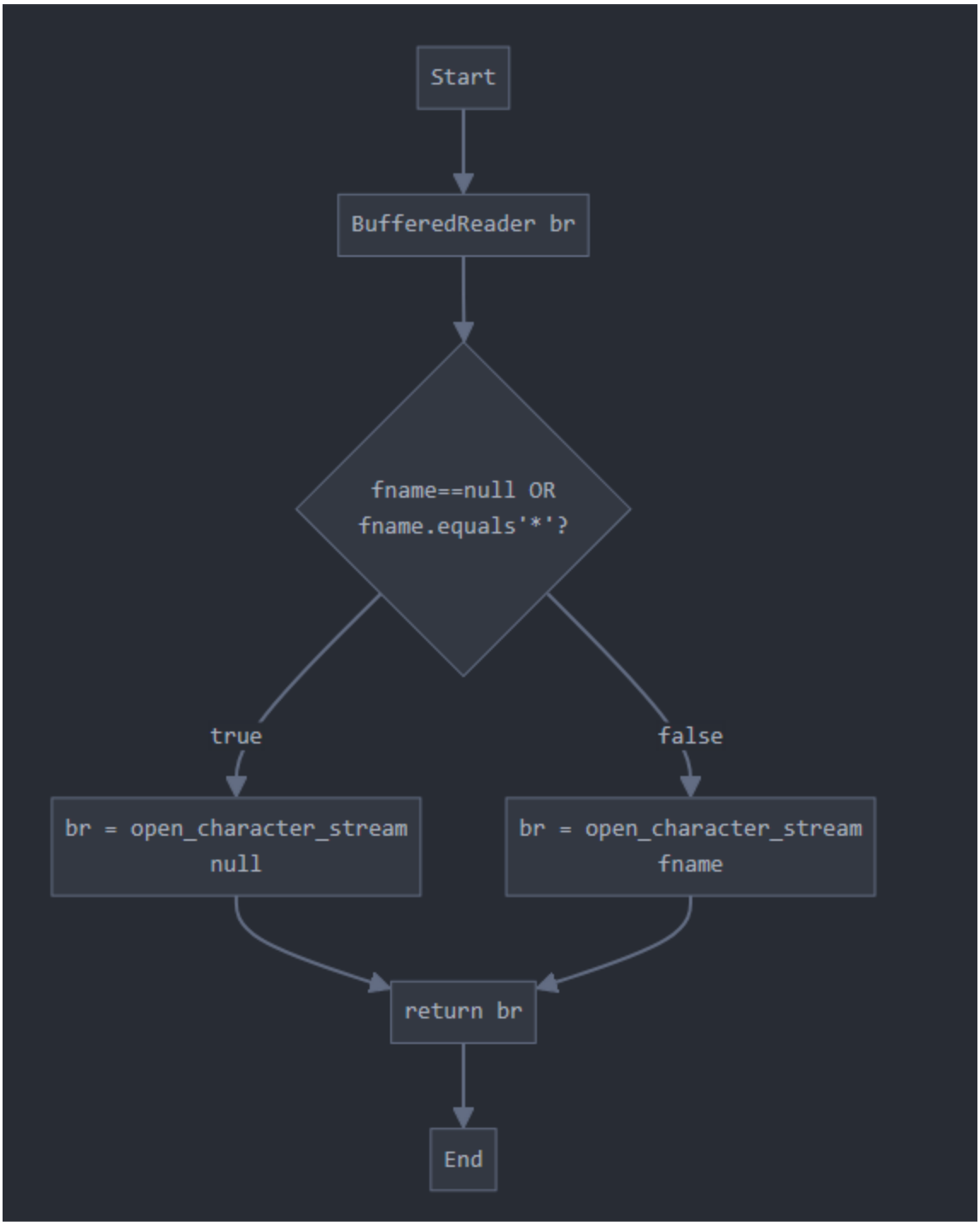
Block Number	Lines Number
1	62
2	66



4. BufferedReader open_token_stream(String fname)

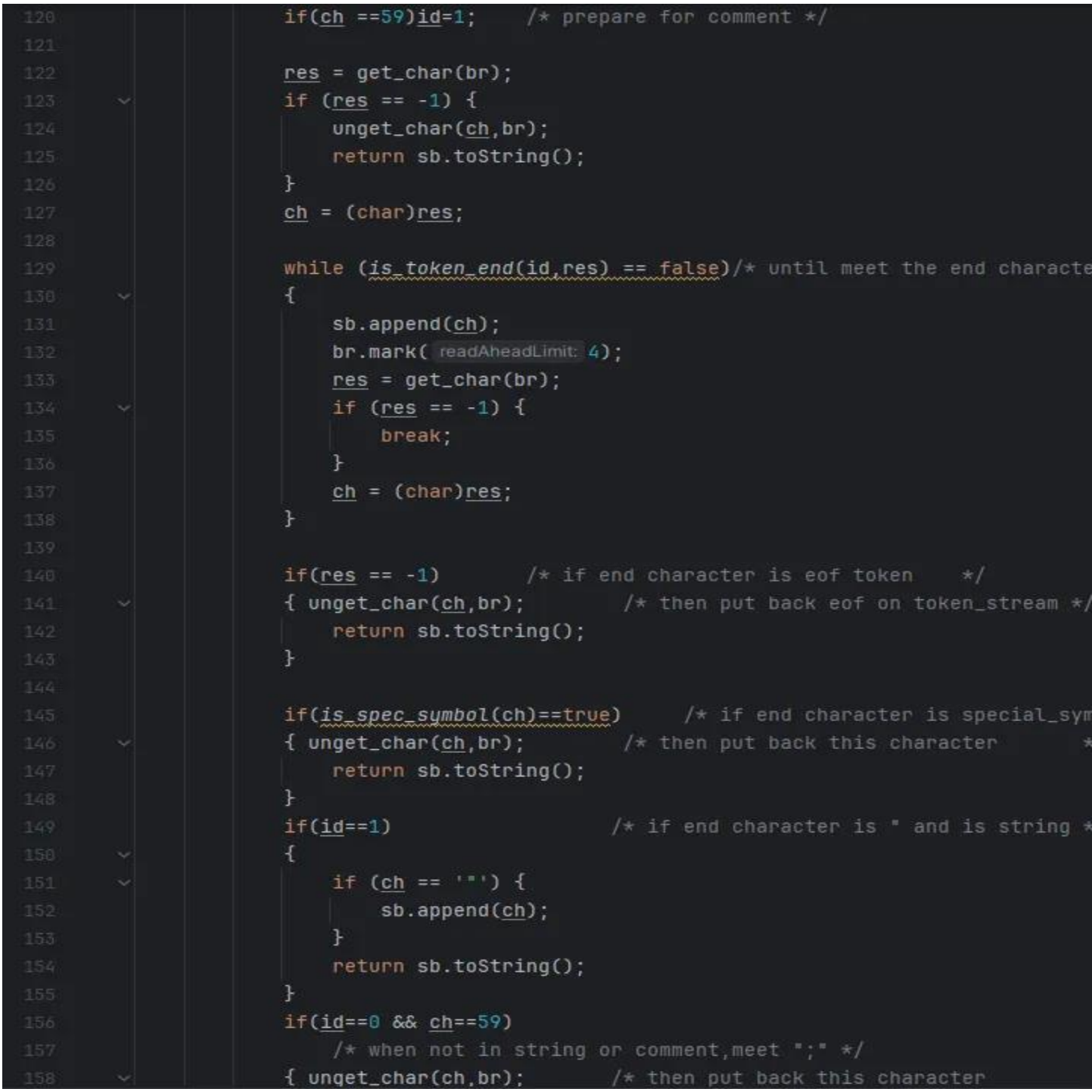
```
78      BufferedReader open_token_stream(String fname)  1 usage
79      {
80          BufferedReader br;
81          if(fname==null || fname.equals(""))
82              br=open_character_stream( fname: null);
83          else
84              br=open_character_stream(fname);
85          return br;
86      }
87
```

Block Number	Lines Number
1	80,81
2	82
3	84
4	85



5. String get_token(BufferedReader br)

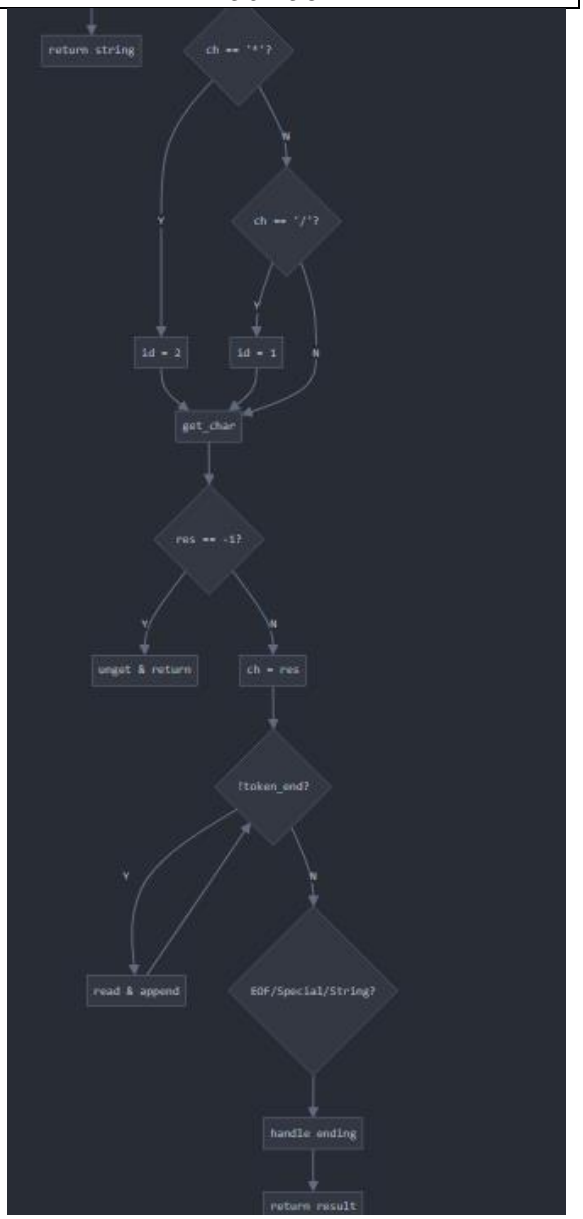
```
95      String get_token(BufferedReader br)  2 usages
96      {
97          int i=0,j;
98          int id=0;
99          int res = 0;
100         char ch = '\0';
101
102         StringBuilder sb = new StringBuilder();
103
104         try {
105             res = get_char(br);
106             if (res == -1) {
107                 return null;
108             }
109             ch = (char)res;
110             while(ch==' '||ch=='\n' || ch == '\r')
111             {
112                 res = get_char(br);
113                 ch = (char)res;
114             }
115
116             if(res == -1)return null;
117             sb.append(ch);
118             if(is_spec_symbol(ch)==true)return sb.toString();
119             if(ch =='"')id=2;    /* prepare for string */
120             if(ch ==59)id=1;    /* prepare for comment */
```



```
120             if(ch ==59)id=1;    /* prepare for comment */
121
122             res = get_char(br);
123             if (res == -1) {
124                 unget_char(ch,br);
125                 return sb.toString();
126             }
127             ch = (char)res;
128
129             while (is_token_end(id,res) == false)/* until meet the end characte
130             {
131                 sb.append(ch);
132                 br.mark( readAheadLimit: 4);
133                 res = get_char(br);
134                 if (res == -1) {
135                     break;
136                 }
137                 ch = (char)res;
138             }
139
140             if(res == -1)       /* if end character is eof token    */
141             { unget_char(ch,br);        /* then put back eof on token_stream */
142                 return sb.toString();
143             }
144
145             if(is_spec_symbol(ch)==true)     /* if end character is special_sym
146             { unget_char(ch,br);        /* then put back this character       *
147                 return sb.toString();
148             }
149             if(id==1)                   /* if end character is " and is string *
150             {
151                 if (ch == '"') {
152                     sb.append(ch);
153                 }
154                 return sb.toString();
155             }
156             if(id==0 && ch==59)
157                 /* when not in string or comment,meet ";" */
158             { unget_char(ch,br);        /* then put back this character
```

Block Number	Lines Number
1	97-101
2	102
3	104-108
4	109-114
5	115-117

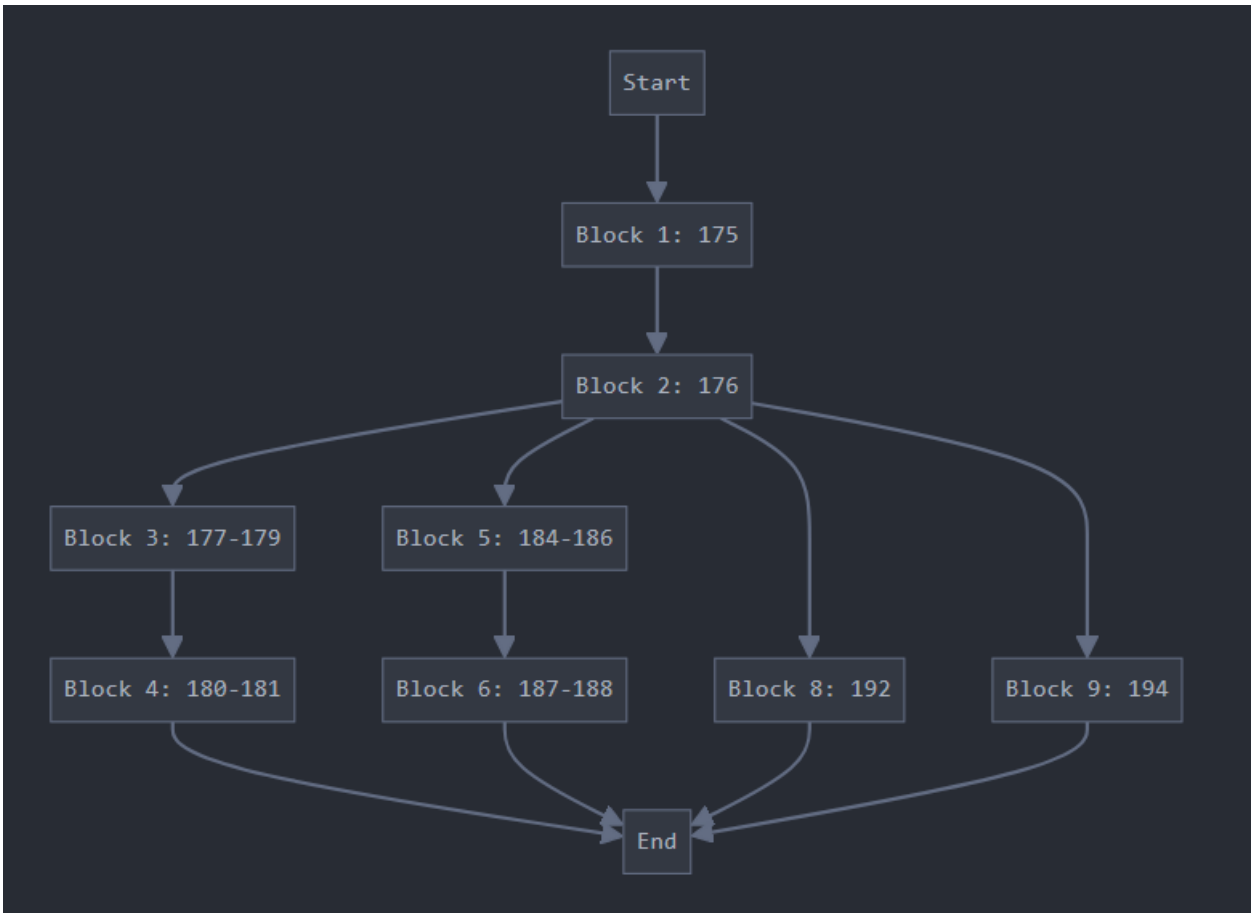
6	118
7	119-120
8	121-127
9	129-138
10	140-143
11	145-148
12	149-154
13	156-158



6. Static Boolean is_token_end(int str_com_id, int res)

```
173     static boolean is_token_end(int str_com_id, int res)  8 usages
174     {
175         if(res==-1)return(true); /* is eof token? */
176         char ch = (char)res;
177         if(str_com_id==1)          /* is string token */
178         { if(ch=='"' || ch=='\n' || ch == '\r' || ch=='\t')   /* for string until meet
179             return true;
180         else
181             return false;
182         }
183
184         if(str_com_id==2)    /* is comment token */
185         { if(ch=='\n' || ch == '\r' || ch=='\t')     /* for comment until meet end of l
186             return true;
187         else
188             return false;
189         }
190
191         if(is_spec_symbol(ch)==true) return true; /* is special_symbol? */
192         if(ch ==' ' || ch=='\n'|| ch=='\r' || ch==59) return true;
193         /* others until meet blank or tab or 59 */
194         return false;               /* other case,return FALSE */
195     }
```

Block Number	Lines Number
1	175
2	176
3	177-179
4	180-181
5	184-186
6	187-188
7	191
8	192
9	194

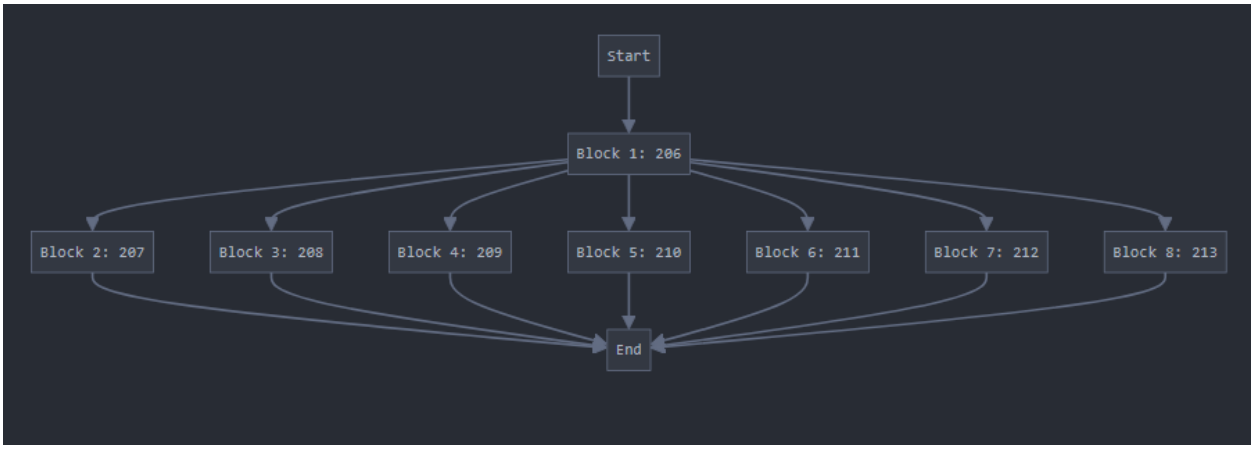


7. Static int token_type(String tok)

```
204     static int token_type(String tok)  1 usage
205     {
206         if(is_keyword(tok))return(keyword);
207         if(is_spec_symbol(tok.charAt(0)))return(spec_symbol);
208         if(is_identifier(tok))return(identifier);
209         if(is_num_constant(tok))return(num_constant);
210         if(is_str_constant(tok))return(str_constant);
211         if(is_char_constant(tok))return(char_constant);
212         if(is_comment(tok))return(comment);
213         return(error);                  /* else look as error token */
214     }
```

Block Number	Lines Number
1	206
2	207
3	208
4	209
5	210
6	211

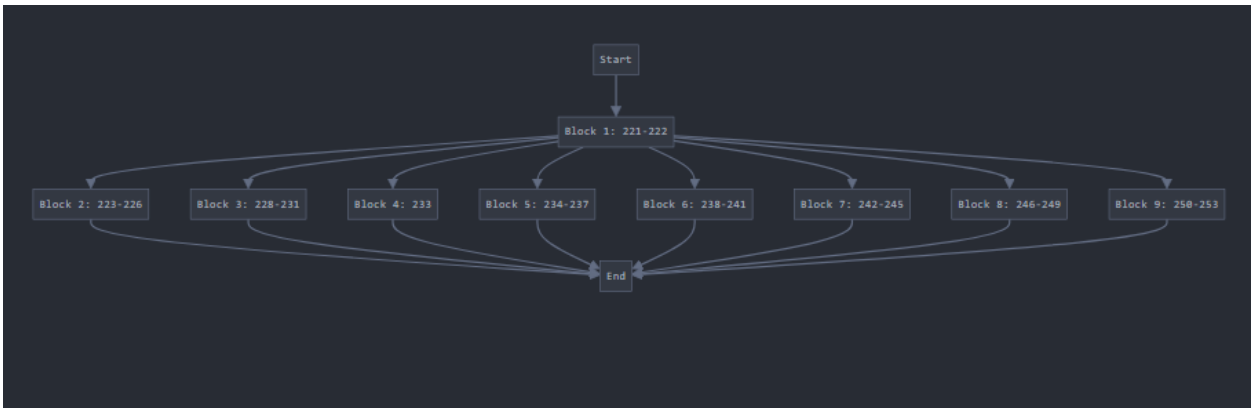
7	212
8	213



8. Void print_token(String tok)

```
220     void print_token(String tok)  1 usage
221     { int type;
222         type=token_type(tok);
223         if(type==error)
224         {
225             System.out.print("error,\"" + tok + "\".\n");
226         }
227
228         if(type==keyword)
229         {
230             System.out.print("keyword,\"" + tok + "\".\n");
231         }
232
233         if(type==spec_symbol)print_spec_symbol(tok);
234         if(type==identifier)
235         {
236             System.out.print("identifier,\"" + tok + "\".\n");
237         }
238         if(type==num_constant)
239         {
240             System.out.print("numeric," + tok + ".\n");
241         }
242         if(type==str_constant)
243         {
244             System.out.print("string," + tok + ".\n");
245         }
246         if(type==char_constant)
247         {
248             System.out.print("character,\"" + tok.charAt(1) + "\".\n");
249         }
250         if(type==comment)
251         {
252             System.out.print("comment,\"" + tok + "\".\n");
253         }
254     }
```

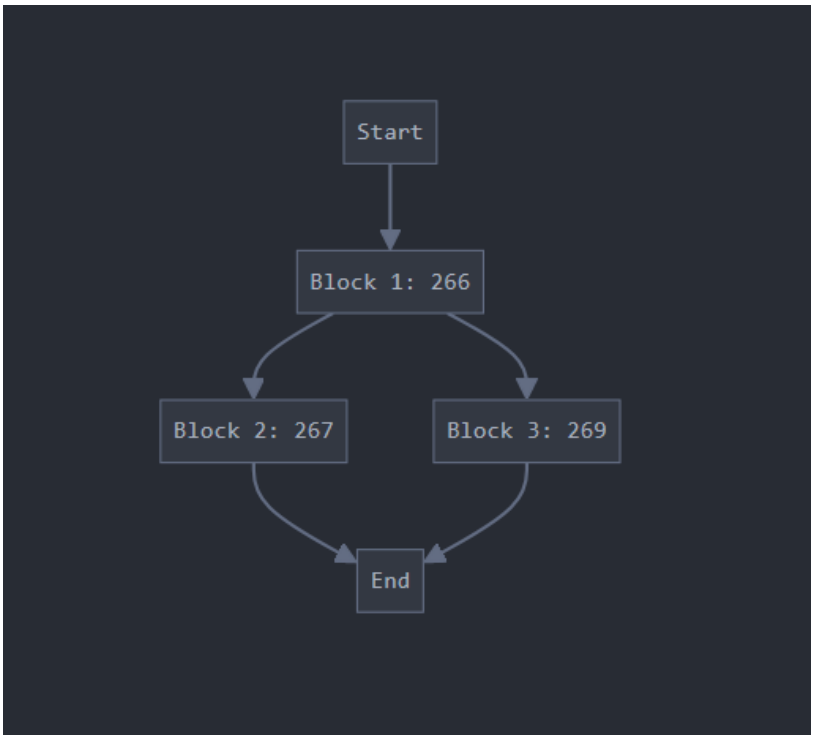
Block Number	Lines Number
1	206
2	207
3	208
4	209
5	210
6	211
7	212
8	213



9. Static Boolean is_comment(String ident)

```
264  @      static boolean is_comment(String ident)  3 usages
265         {
266             if( ident.charAt(0) ==59 )   /* the char is 59   */
267                 return true;
268             else
269                 return false;
270         }
```

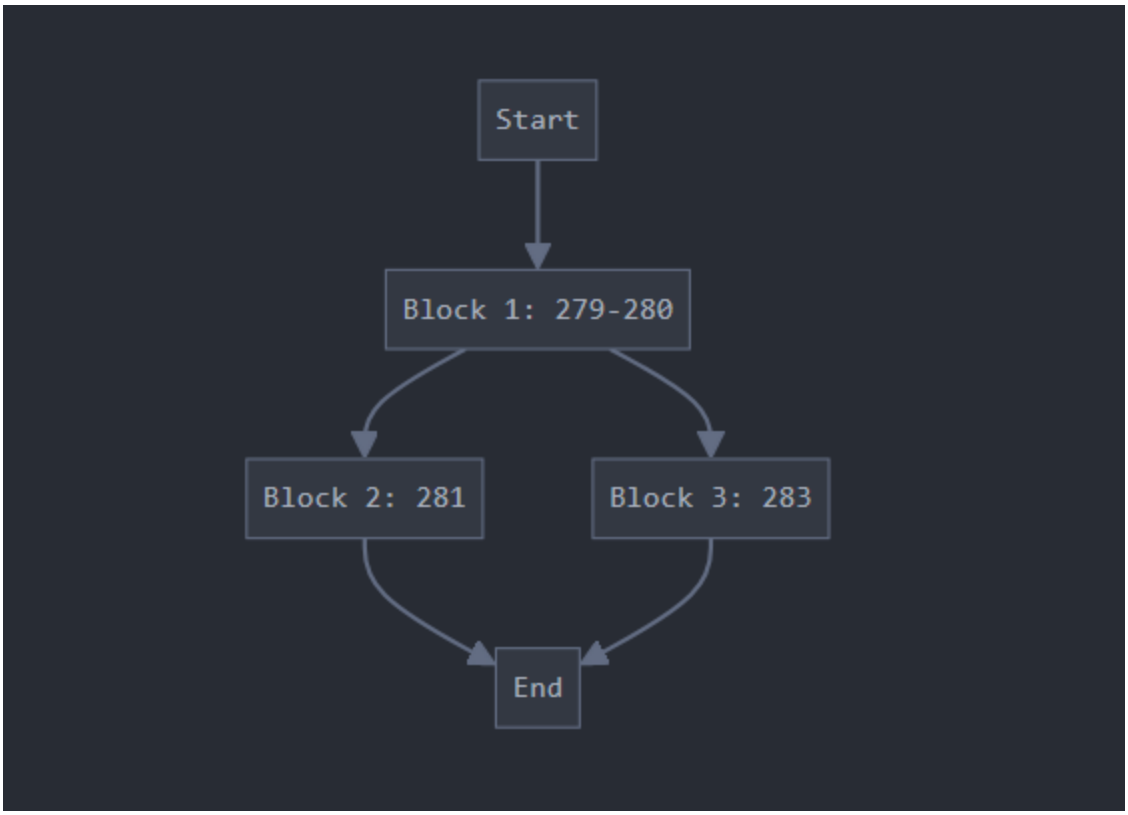
Block Number	Lines Number
1	266
2	267
3	269



10. Static Boolean is_keyword(String str)

```
277  @      static boolean is_keyword(String str)  8 usages
278         {
279             if (str.equals("and") || str.equals("or") || str.equals("if") ||
280                     str.equals("xor")||str.equals("lambda")||str.equals("=>"))
281                 return true;
282             else
283                 return false;
284         }
```

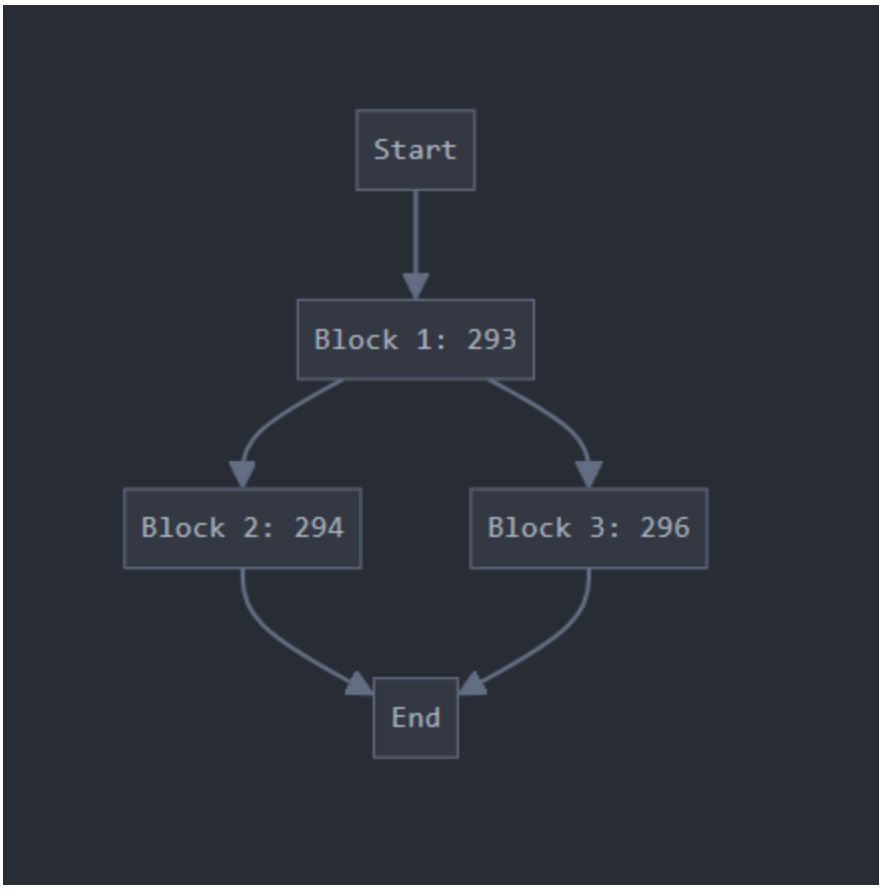
Block Number	Lines Number
1	279-280
2	281
3	283



11. Static Boolean is_char_constant(String str)

```
291 @   static boolean is_char_constant(String str)  5 usages
292     {
293         if (str.length() > 2 || str.charAt(0)=='#' && Character.isLetter(str.charAt(1)))
294             return true;
295         else
296             return false;
297     }
298
```

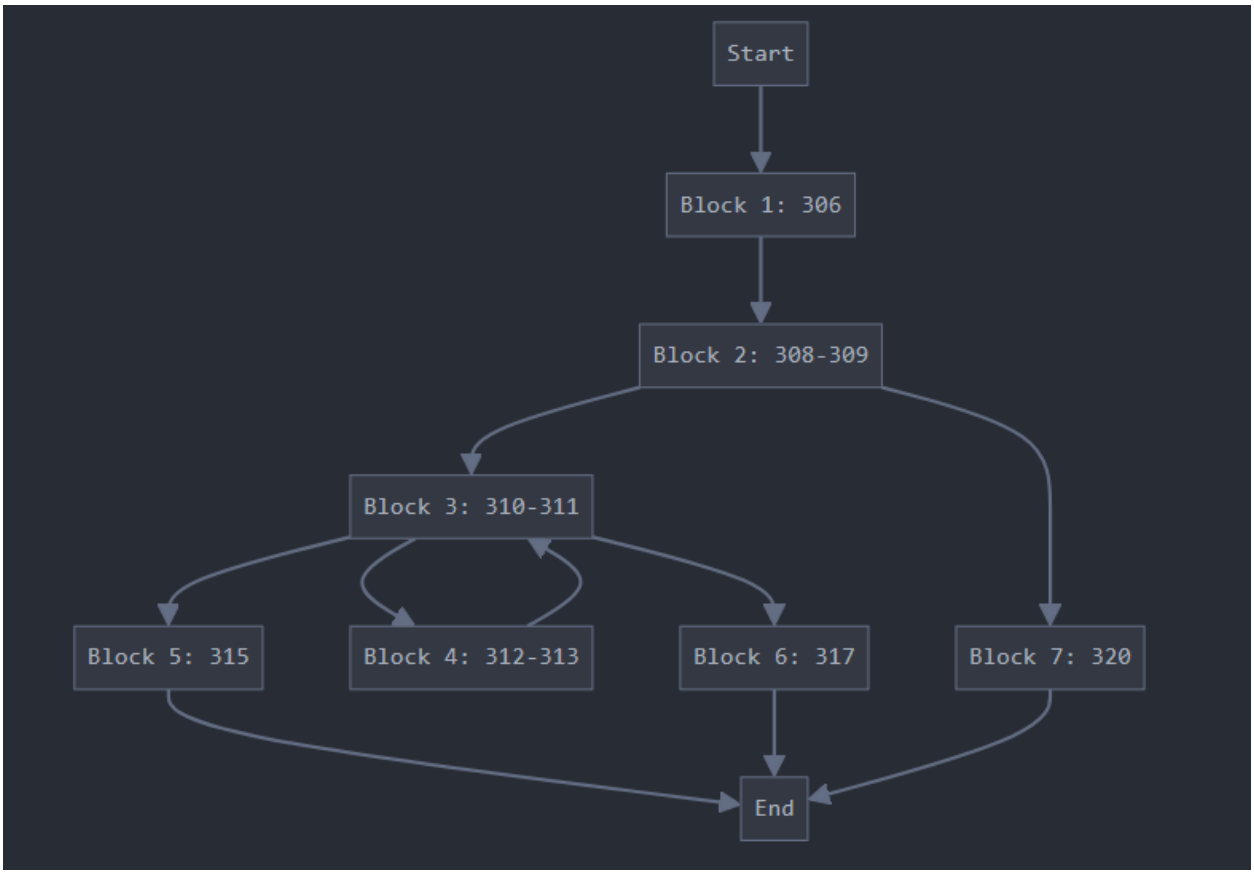
Block Number	Lines Number
1	293
2	294
3	296



12. Static Boolean is_num_constant(String str)

```
304    @    static boolean is_num_constant(String str)  5 usages
305         {
306             int i=1;
307
308             if ( Character.isDigit(str.charAt(0)))
309             {
310                 while ( i < str.length() && str.charAt(i) != '\0' )
311                 {
312                     if(Character.isDigit(str.charAt(i+1)))
313                         i++;
314                     else
315                         return false;
316                 }                          /* end WHILE */
317                 return true;
318             }
319             else
320                 return false;               /* other return FALSE */
321         }
322
```

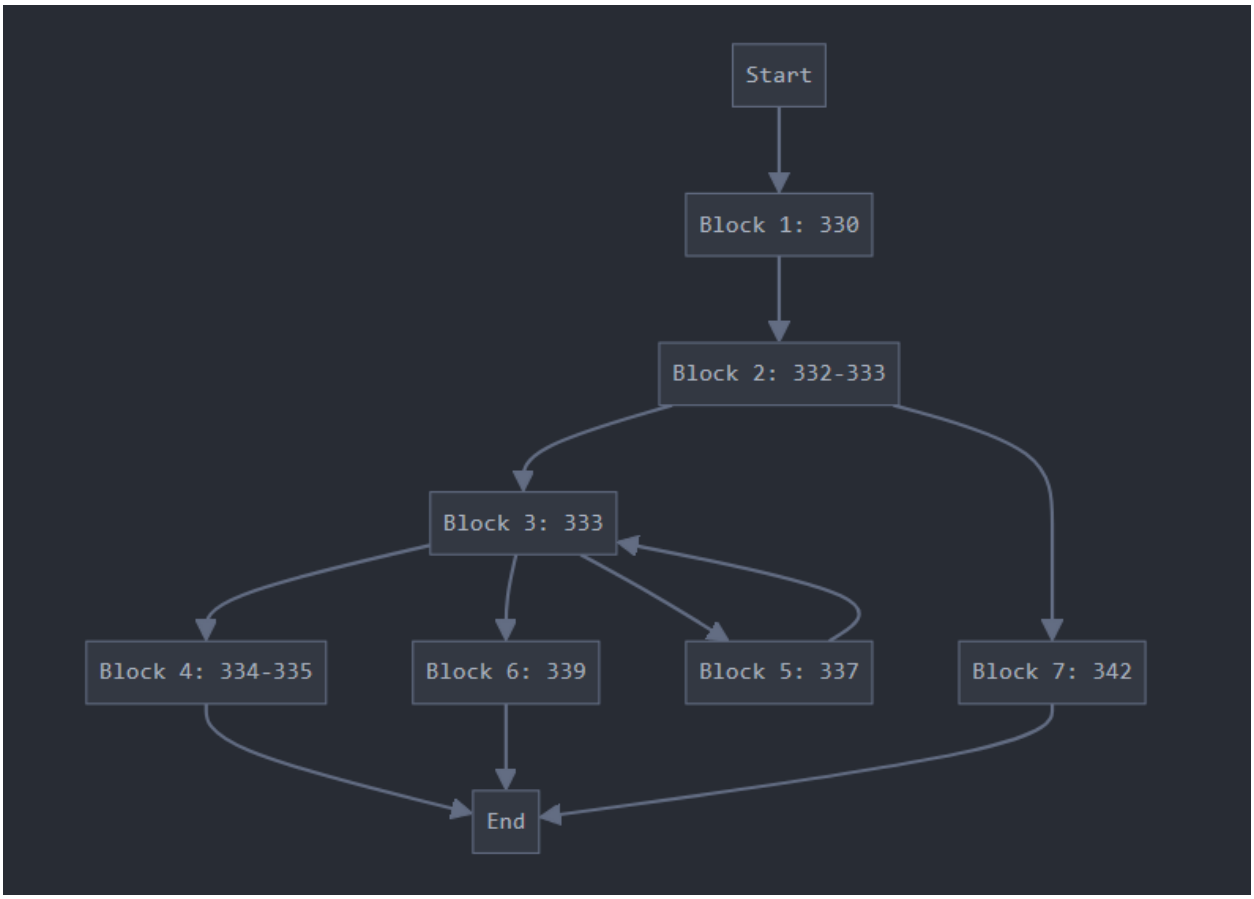
Block Number	Lines Number
1	306
2	308-309
3	310-311
4	312-313
5	315
6	317
7	320



13. Static Boolean is_str_constant(String str)

```
328 @   static boolean is_str_constant(String str)  5 usages
329     {
330         int i=1;
331
332         if ( str.charAt(0) =='"')
333         { while (i < str.length() && str.charAt(i)!='\0')
334         { if(str.charAt(i)=='"' )
335             return true;        /* meet the second '"'          */
336         else
337             i++;
338         }               /* end WHILE */
339             return true;
340         }
341         else
342             return false;       /* other return FALSE */
343     }
344
```

Block Number	Lines Number
1	330
2	332-333
3	333
4	334-335
5	337
6	339
7	342

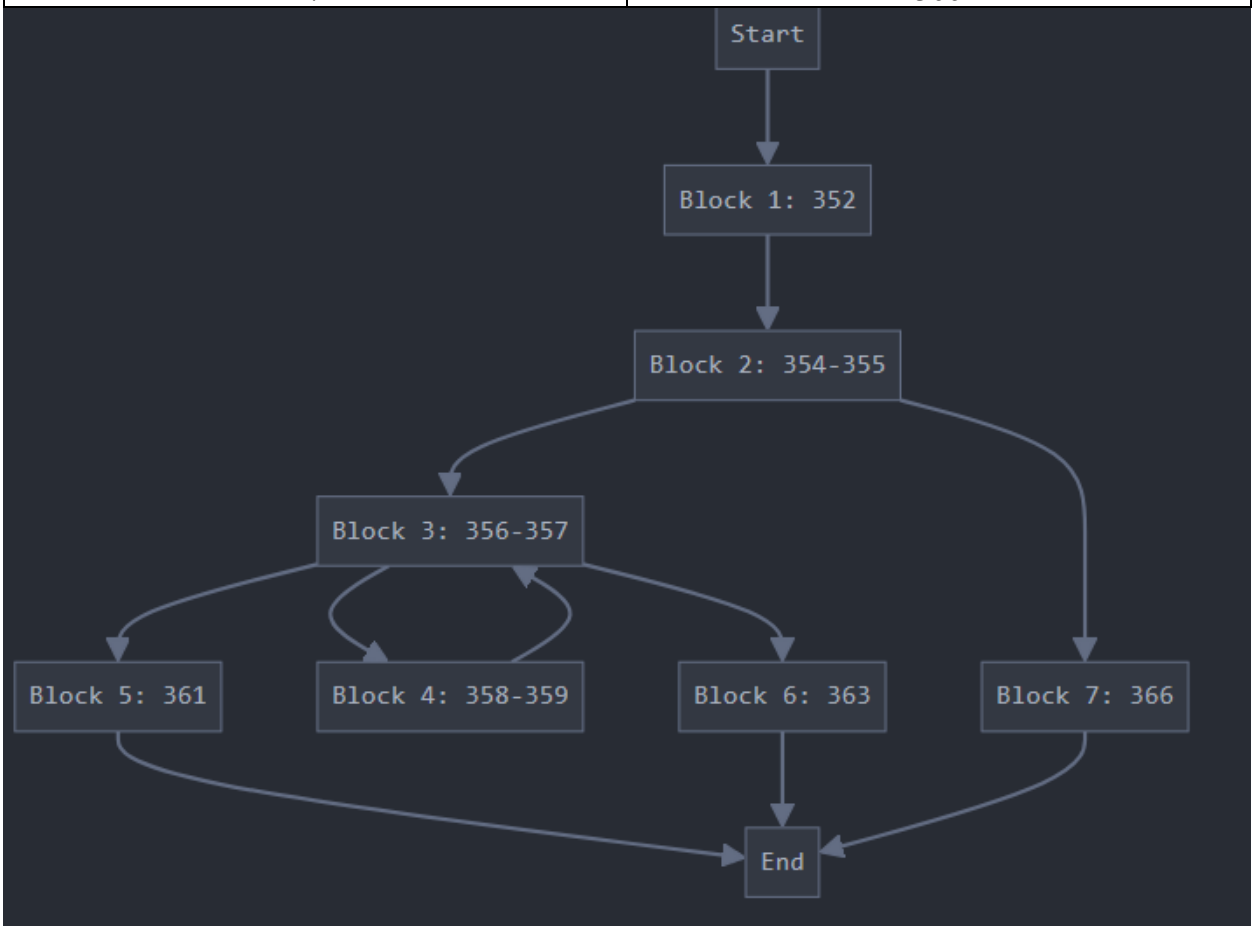


14. Static Boolean is_identifier

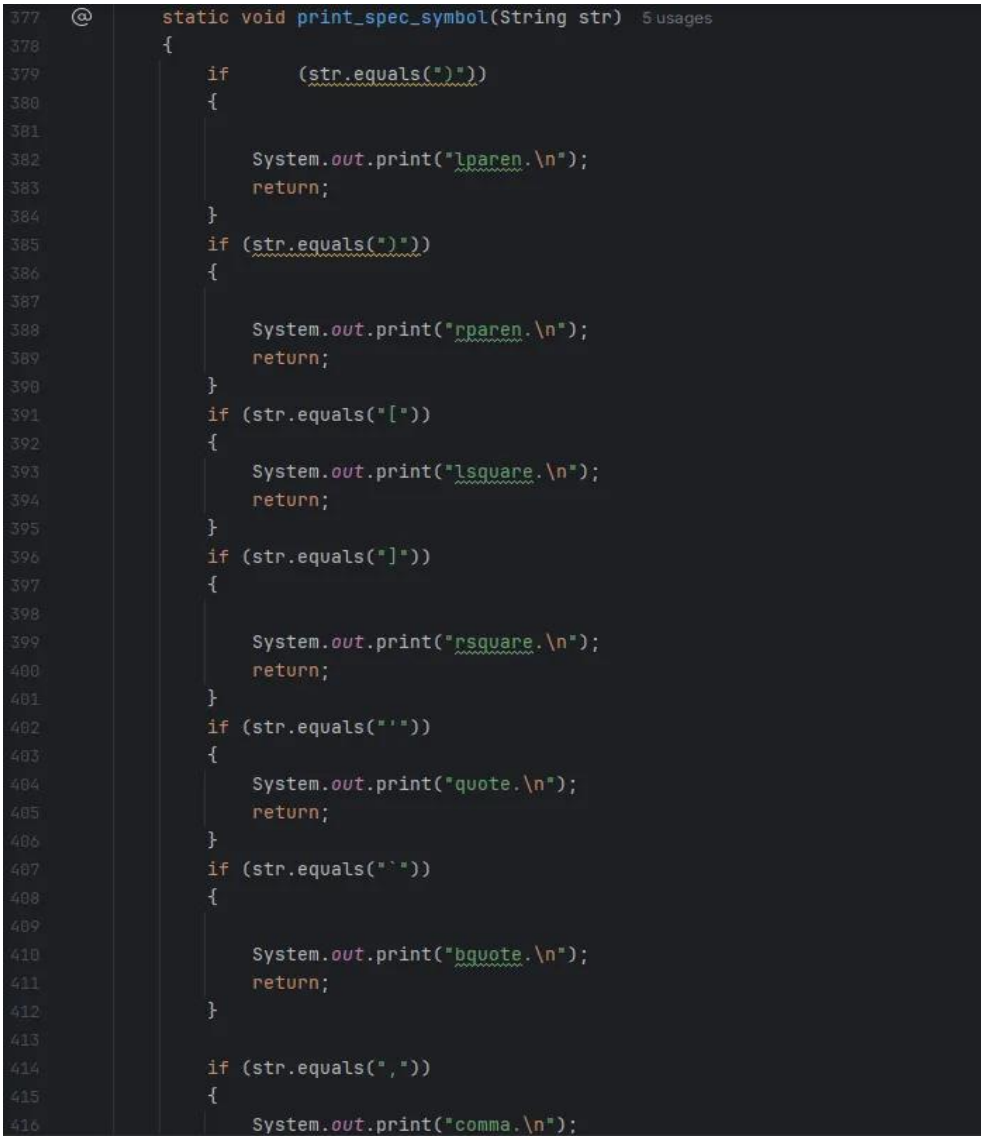
```
350 @   static boolean is_identifier(String str)  6 usages
351     {
352         int i=1;
353
354         if ( Character.isLetter(str.charAt(0)) )
355         {
356             while(i < str.length() && str.charAt(i) !='\0' )   /* unti meet the end token sign */
357             {
358                 if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
359                     i++;
360                 else
361                     return false;
362             }      /* end WHILE */
363             return false;
364         }
365         else
366             return true;
367     }
368
369
```

Block Number	Lines Number
1	352
2	354-355

3	356-357
4	358-359
5	361
6	363
7	366



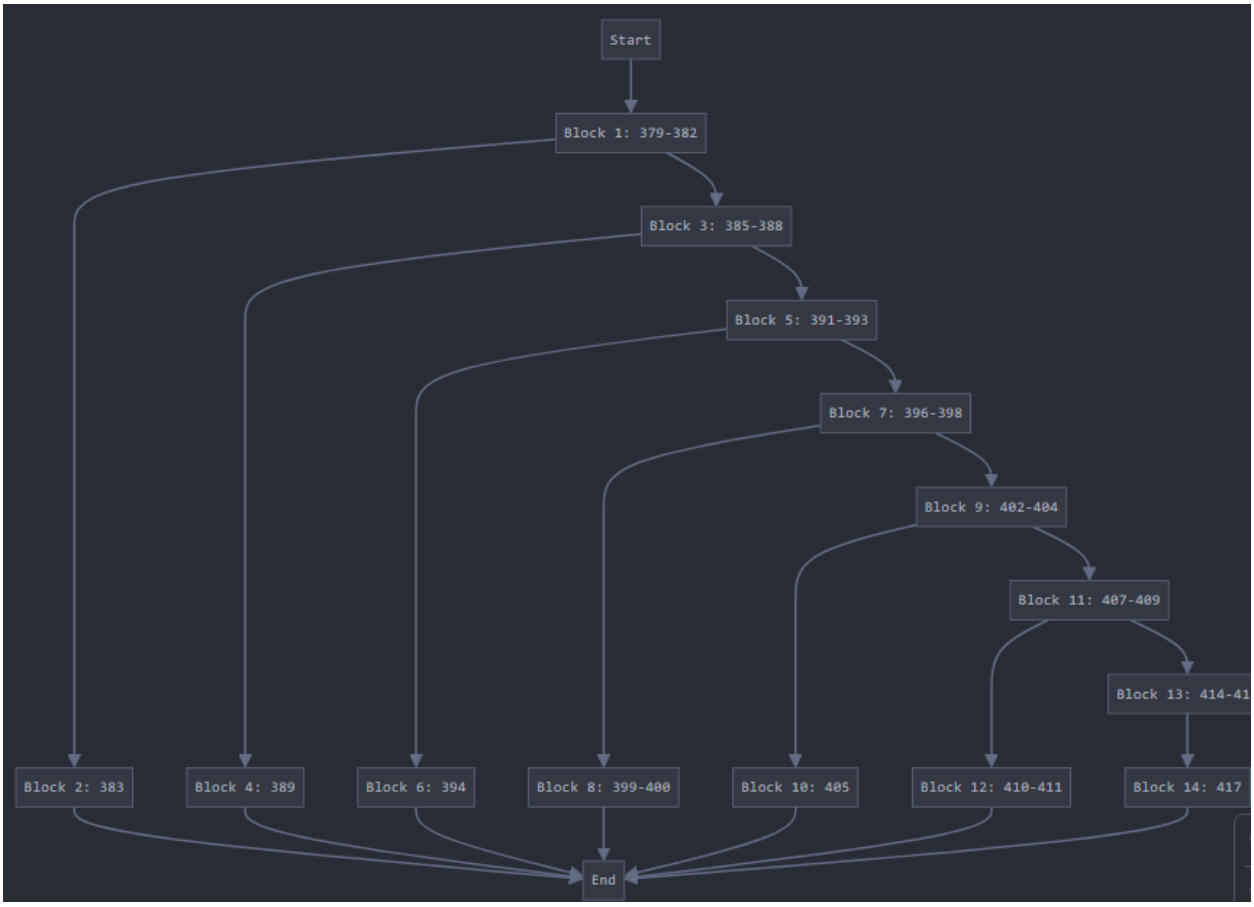
15. Static void print_spec_symbol(String str)



```
377 @       static void print_spec_symbol(String str)  5 usages
378         {
379             if      (str.equals("("))
380             {
381
382                 System.out.print("lparen.\n");
383                 return;
384             }
385             if (str.equals(")"))
386             {
387
388                 System.out.print("rparen.\n");
389                 return;
390             }
391             if (str.equals("["))
392             {
393                 System.out.print("lsquare.\n");
394                 return;
395             }
396             if (str.equals("]"))
397             {
398
399                 System.out.print("rsquare.\n");
400                 return;
401             }
402             if (str.equals("'"))
403             {
404                 System.out.print("quote.\n");
405                 return;
406             }
407             if (str.equals("`"))
408             {
409
410                 System.out.print("bquote.\n");
411                 return;
412             }
413
414             if (str.equals(","))
415             {
416                 System.out.print("comma.\n");
```

Block Number	Lines Number
1	379-382
2	383
3	385-388
4	389
5	391-393
6	394
7	396-398
8	399-400
9	402-404
10	405
11	407-409
12	410-411
13	414-416

14	417

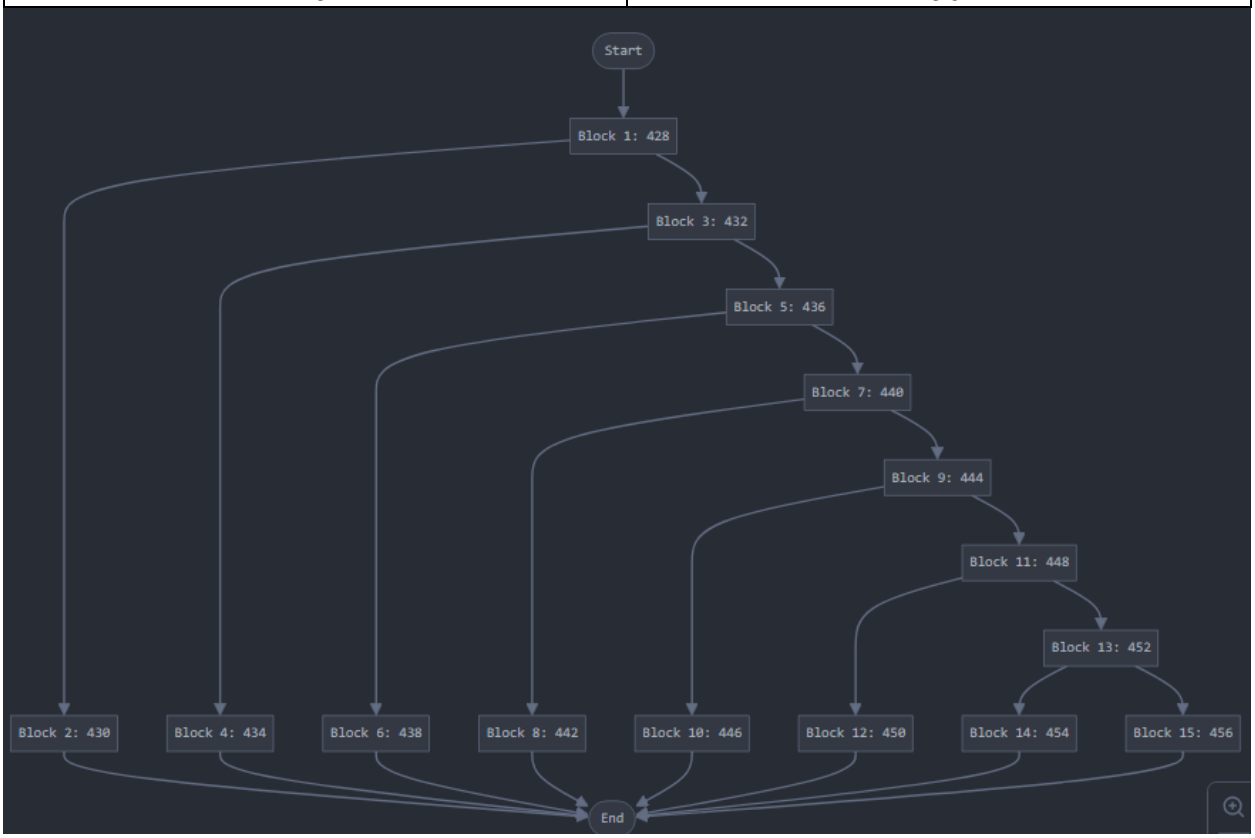


16. Static Boolean is_spec_symbol(char c)

```
426     static boolean is_spec_symbol(char c)  4 usages
427     {
428         if (c == '(')
429         {
430             return true;
431         }
432         if (c == ')')
433         {
434             return true;
435         }
436         if (c == '[')
437         {
438             return true;
439         }
440         if (c == ']')
441         {
442             return true;
443         }
444         if (c == '/')
445         {
446             return true;
447         }
448         if (c == '`')
449         {
450             return true;
451         }
452         if (c == ',')
453         {
454             return true;
455         }
456         return false;     /* others return FALSE */
457     }
```

Block Number	Lines Number
1	428
2	430
3	432
4	434
5	436
6	438
7	440
8	442
9	444

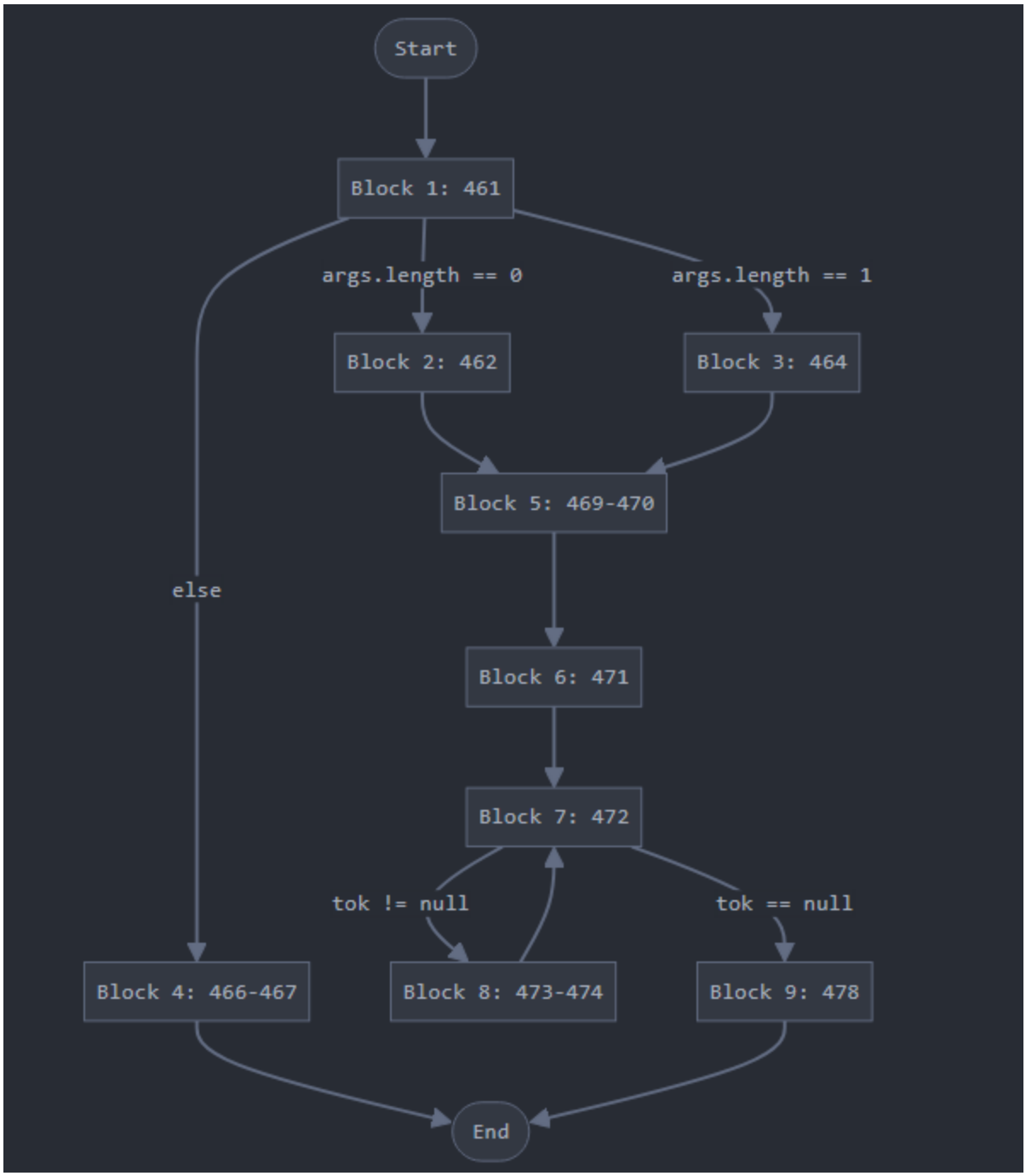
10	446
11	448
12	450
13	452
14	454
15	456



17. Public static void main(String[] args)

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

Block Number	Lines Number
1	461
2	462
3	464
4	466-467
5	469-470
6	471
7	472
8	473-474
9	478


Start
Block 1: 461
args.length == 0
args.length == 1
Block 2: 462
Block 3: 464
Block 5: 469-470
else
Block 6: 471
Block 7: 472
tok != null
tok == null
Block 4: 466-467
Block 8: 473-474
Block 9: 478
End