$\ensuremath{\mathsf{CSE}}\textsc{-4321}$ Software Testing Final Project - Test Cases

Maximum Effort: Elliot Mai & Benjamin Niccum

December 2, 2024

Contents

1	Uni	t Tests		2
	1.1	Method:	openCharacterStream	2
	1.2	Method:	getChar	4
	1.3		ungetChar	
	1.4	Method:	openTokenStream	7
	1.5	Method:	getToken	9
	1.6	Method:	isTokenEnd	12
	1.7	Method:	tokenType	14
	1.8	Method:	printToken	17
	1.9	Method:	isComment	19
	1.10	Method:	isKeyword	21
	1.11	Method:	isCharConstant	23
	1.12	Method:	isNumConstant	24
	1.13	Method:	isStrConstant	26
	1.14	Method:	isIdentifier	28
			printSpecSymbol	
	1.16	Method:	isSpecSymbol	32
2	Pro	gram Te	${f sts}$	35

1 Unit Tests

ormalsize1.1 Method: openCharacterStream

```
BufferedReader open_character_stream(String fname) {
22
23
             BufferedReader br = null;
24
             if (fname == null) {
                  br = new BufferedReader(new InputStreamReader(System.in));
25
26
               else {
27
                  try {
                      FileReader fr = new FileReader(fname);
28
                      br = new BufferedReader(fr);
29
30
                  } catch (FileNotFoundException e) {
                      System.out.print("The file " + fname +" doesn't exists\n");
31
32
                      e.printStackTrace();
33
34
35
36
             return br;
37
```

Figure 1: Code Snippet for openCharacterStream

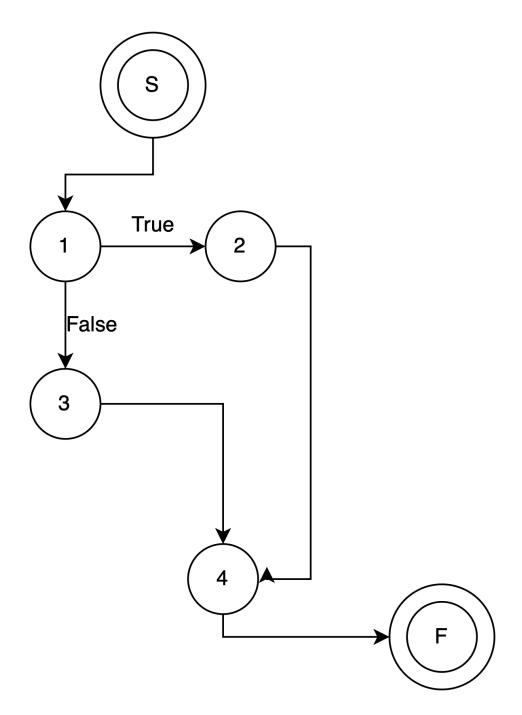


Figure 2: Control Flow Graph for openCharacterStream

Test Path	Test Data	Expected Output
1, 2, 4	null	notNull
1, 3, 4	file.txt	True

Table 1: Test Cases for openCharacterStream

ormalsize1.2 Method: getChar

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

Figure 3: Code Snippet for getChar

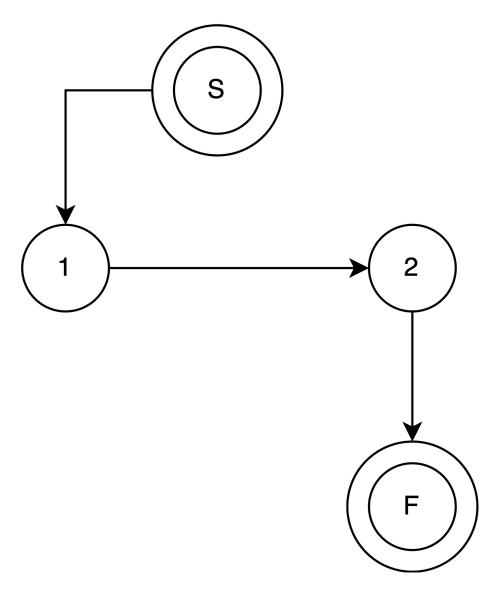


Figure 4: Control Flow Graph for getChar

Test Path	Test Data	Expected Output
1, 2	abc	a

Table 2: Test Cases for getChar

${\bf ormal size 1.3} \quad {\bf Method: \ ungetChar}$

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

Figure 5: Code Snippet for ungetChar

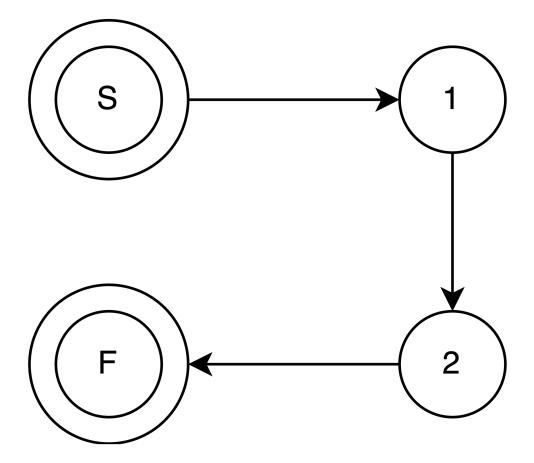


Figure 6: Control Flow Graph for unget Char

Test Path	Test Data	Expected Output
1, 2	abc	0

Table 3: Test Cases for ungetChar

ormalsize1.4 Method: openTokenStream

```
BufferedReader open_token_stream(String fname)

{
BufferedReader br;

if(fname==null || fname.equals(anObject:""))

br=open_character_stream(fname:null);

else

br=open_character_stream(fname);

return br;

}
```

Figure 7: Code Snippet for openTokenStream

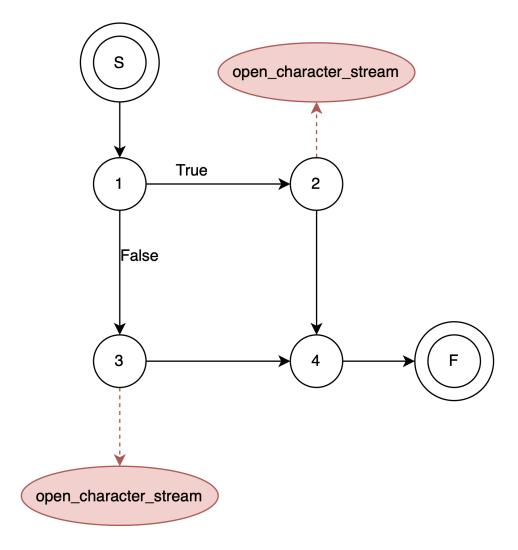


Figure 8: Control Flow Graph for openTokenStream

Test Path	Test Data	Expected Output
1, 2, 4	test.txt	notNull
1, 3, 4	null	Null

Table 4: Test Cases for openTokenStream

ormalsize1.5 Method: getToken

```
94
          String get_token(BufferedReader br)
 95
            int i=0,j;
 96
 97
            int id=0;
98
            int res = 0;
            char ch = '\0';
 99
100
            StringBuilder sb = new StringBuilder();
101
102
103
             try {
104
                 res = get_char(br);
                 if (res == -1) {
105
106
                    return null;
107
                 ch = (char)res;
108
                while(ch==' '||ch=='\n' || ch == '\r')
109
110
111
                  res = get_char(br);
                  ch = (char)res;
112
113
114
             if(res == -1)return null;
115
116
             sb.append(ch);
117
             if(is_spec_symbol(ch)==true)return sb.toString();
118
             if(ch =='"')id=2; /* prepare for string */
             if(ch ==59)id=1; /* prepare for comment */
119
120
121
             res = get_char(br);
122
             if (res == -1) {
123
                 unget_char(ch,br);
                 return sb.toString();
124
125
             }
             ch = (char)res;
126
127
             while (is_token_end(id,res) == false)/* until meet the end character */
128
129
130
                 sb.append(ch);
131
                 br.mark(readAheadLimit:4);
132
                 res = get char(br);
133
                 if (res == -1) {
134
                    break;
135
                 ch = (char)res;
136
137
138
139
                             /* if end character is eof token */
             if(res == -1)
140
                { unget_char(ch,br);
                                          /* then put back eof on token_stream */
141
                  return sb.toString();
               }
142
143
             if(is_spec_symbol(ch)==true) /* if end character is special_symbol */
144
145
               { unget char(ch,br);
                                            /* then put back this character
146
                  return sb.toString();
147
                                       /* if end character is " and is string */
148
             if(id==1)
149
               {
                 if (ch == '"') {
150
                    sb.append(ch);
151
152
153
                 return sb.toString();
```

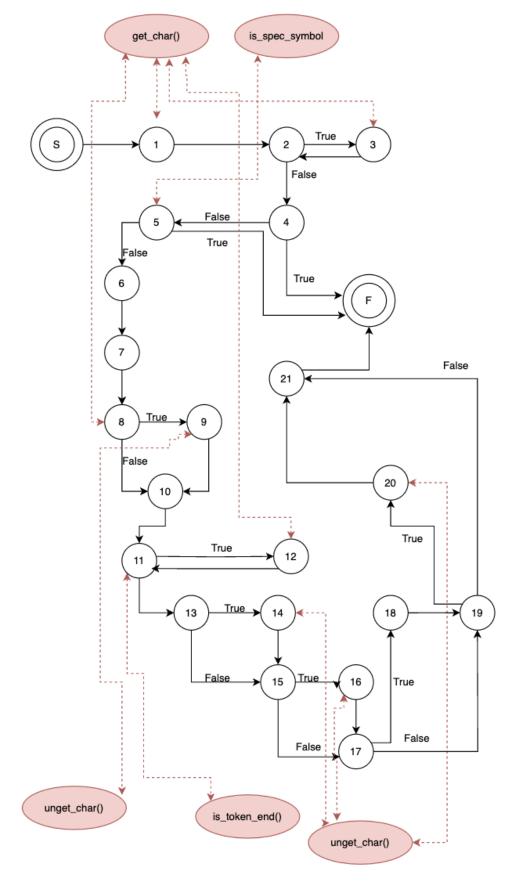


Figure 10: Control Flow Graph for getToken

Test Path	Test Data	Expected Output
1	11 11	Null
1, 2, 3, 2, 4	"\n\n"	Null
1, 2, 4, 5	"("	"("
1, 2, 4, 5, 6, 7, 8, 9	"a"	"a"
1, 2, 4, 5, 6, 7, 8, 9,	"abc"	"abc"
10, 11, 12, 11, 13		
1, 2, 4, 5, 6, 7, 8, 9,	"abc)"	"abc"
10, 11, 13, 14, 15		
1, 2, 4, 5, 6, 7, 8, 9,	""abc ⁱⁱ	""abc""
10, 11, 13, 14, 15, 16,		
17		
1, 2, 4, 5, 6, 7, 8, 9,	";abc;"	";abc;"
10, 11, 13, 14, 15, 16,		
17, 18, 19		
1, 2, 4, 5, 6, 7, 8, 9,	"hello"	"hello"
10, 11, 13, 14, 15, 16,		
17, 18, 19, 20, 21		

Table 5: Test Cases for getToken

ormalsize1.6 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 */
             { if(ch=='"' || ch=='\n' || ch == '\r' || ch=='\t') /* for string until meet another " */
177
178
                return true;
179
               else
                return false;
180
181
182
           if(str_com_id==2) /* is comment token */
183
            { if(ch=='\n' || ch == '\r' || ch=='\t') /* for comment until meet end of line */
184
185
                return true;
186
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;
           /* others until meet blank or tab or 59 */
192
193
           return false;
                                    /* other case,return FALSE */
194
```

Figure 11: Code Snippet for isTokenEnd

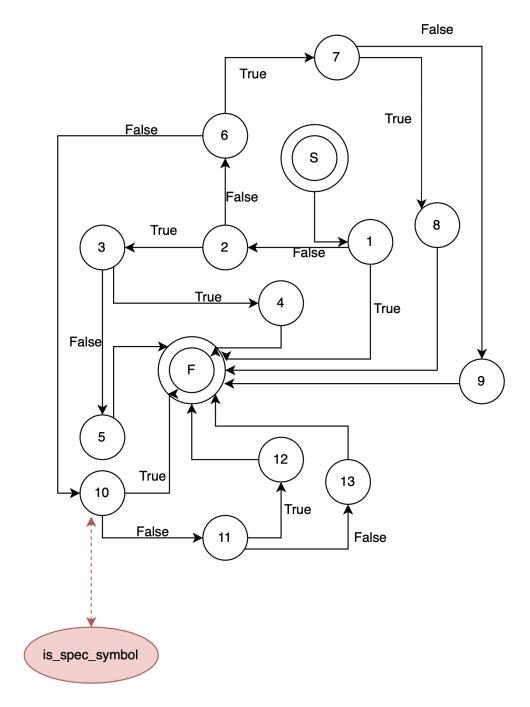


Figure 12: Control Flow Graph for isTokenEnd

Test Path	Test Data	Expected Output
1	0, -1	true
1, 2, 3, 4	1, 34	true
1, 2, 3, 5	1, 97	false
1, 2, 6, 7, 9	2, 97	false
1, 2, 6, 7, 8	2, 10	true
1, 2, 6, 10	0, 40	true
1, 2, 6, 11, 12	0, 32	true
1, 2, 6, 10, 11, 13	0, 97	false

Table 6: Test Cases for isTokenEnd

ormalsize1.7 Method: tokenType

```
203
          static int token_type(String tok)
204
205
           if(is_keyword(tok))return(keyword);
           if(is_spec_symbol(tok.charAt(index:0)))return(spec_symbol);
206
207
           if(is_identifier(tok))return(identifier);
           if(is_num_constant(tok))return(num_constant);
208
209
           if(is_str_constant(tok))return(str_constant);
210
           if(is_char_constant(tok))return(char_constant);
           if(is_comment(tok))return(comment);
211
212
           return(error);
                                              /* else look as error token */
213
```

Figure 13: Code Snippet for tokenType

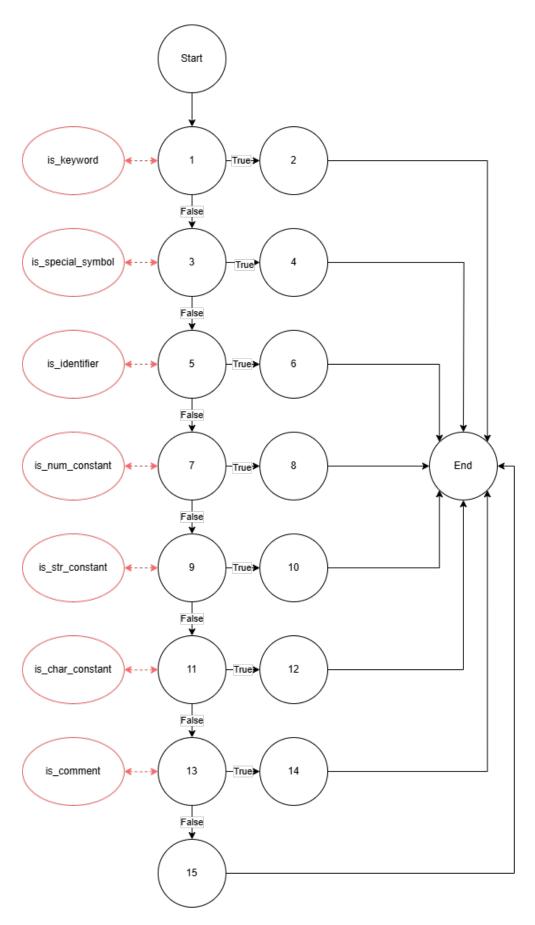


Figure 14: Control Flow Graph for tokenType

Test Path	Test Data	Expected Output
1, 2	"and"	1
1, 2	"or"	1
1, 2	"if"	1
1, 2	"xor"	1
1, 2	"lambda"	1
1, 2	"=>"	1
1, 3, 4	"("	2
1, 3, 4	")"	2
1, 3, 4	"["	2
1, 3, 4	"]"	2
1, 3, 4	11) 11	2
1, 3, 4	11 (11	2
1, 3, 4	","	2
1, 3, 5, 6	"variableName"	3
1, 3, 5, 6	"a"	3
1, 3, 5, 6	"aa"	3
1, 3, 5, 6	"a1"	3
1, 3, 5, 6	"a2"	3
1, 3, 5, 7, 8	"123"	41
1, 3, 5, 7, 8	"1"	41
1, 3, 5, 7, 8	"321"	41
1, 3, 5, 7, 9, 10	""Hello""	42
1, 3, 5, 7, 9, 10	""asd""	42
1, 3, 5, 7, 9, 10	""123""	42
1, 3, 5, 7, 9, 11, 13, 14	";comment"	5
1, 3, 5, 7, 9, 11, 12	"#a"	43
1, 3, 5, 7, 9, 11, 12	"#b"	43
1, 3, 5, 7, 9, 11, 13, 15	"*^&"	0

Table 7: Test Cases for tokenType

ormalsize1.8 Method: printToken

```
219
          void print_token(String tok)
220
           { int type;
221
            type=token_type(tok);
           if(type==error)
222
223
              System.out.print("error,\"" + tok + "\".\n");
224
225
226
227
           if(type==keyword)
228
             System.out.print("keyword,\"" + tok + "\".\n");
229
230
231
232
           if(type==spec symbol)print spec symbol(tok);
           if(type==identifier)
233
234
             System.out.print("identifier,\"" + tok + "\".\n");
235
236
           if(type==num_constant)
237
238
             System.out.print("numeric," + tok + ".\n");
239
240
241
           if(type==str_constant)
242
243
             System.out.print("string," + tok + ".\n");
244
245
           if(type==char_constant)
246
              System.out.print("character,\"" + tok.charAt(index:1) + "\".\n");
247
248
249
           if(type==comment)
250
              System.out.print("comment,\"" + tok + "\".\n");
251
252
253
```

Figure 15: Code Snippet for printToken

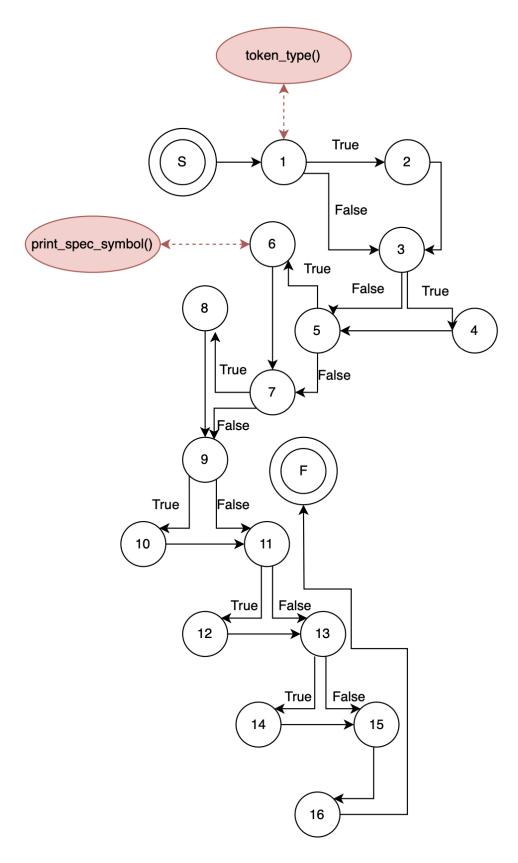


Figure 16: Control Flow Graph for printToken

Test Path	Test Data	Expected Output
1, 2, 3, 5, 7, 9, 11, 13,	"*^&"	"error,"*^&"."
15		
1, 3, 4, 5, 7, 9, 11, 13,	"if"	"keyword,"if"."
15		
1, 3, 5, 6, 7, 9, 11, 13,	"("	"lparen."
15		
1, 3, 5, 7, 8, 9, 11, 13,	"variableName"	identifier,"variableName
15		
1, 3, 5, 7, 9, 10, 11,	"123"	numeric, 123.
13, 15		
1, 3, 5, 7, 9, 11, 12,	""Hello""	"string,
13, 15		"Hello"."
1, 3, 5, 7, 9, 11, 13,	"#a"	"character, "a"."
14, 15		
1, 3, 5, 7, 9, 11, 13,	";comment"	"comment,
15, 16		";comment"."

Table 8: Test Cases for printToken

ormalsize1.9 Method: isComment

Figure 17: Code Snippet for isComment

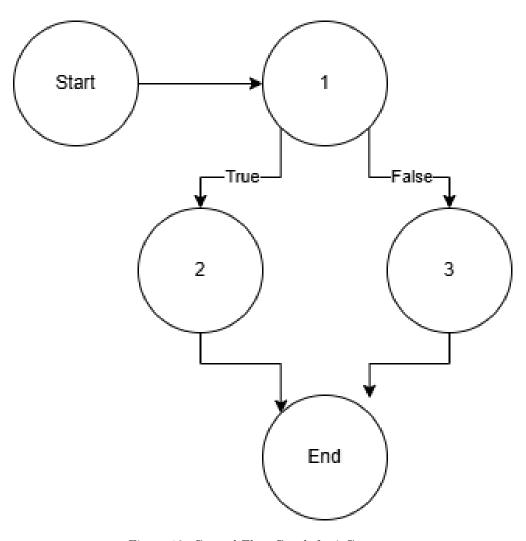


Figure 18: Control Flow Graph for isComment

Test Path	Test Data	Expected Output
1, 2	","	true
1, 3	"x"	false

Table 9: Test Cases for isComment

ormalsize1.10 Method: isKeyword

```
static boolean is_keyword(String str)

{

if (str.equals(anObject:"and") || str.equals(anObject:"or") || str.equals(anObject:"if") ||

str.equals(anObject:"xor")||str.equals(anObject:"lambda")||str.equals(anObject:"=>"))

return true;
else

return false;
}
```

Figure 19: Code Snippet for isKeyword

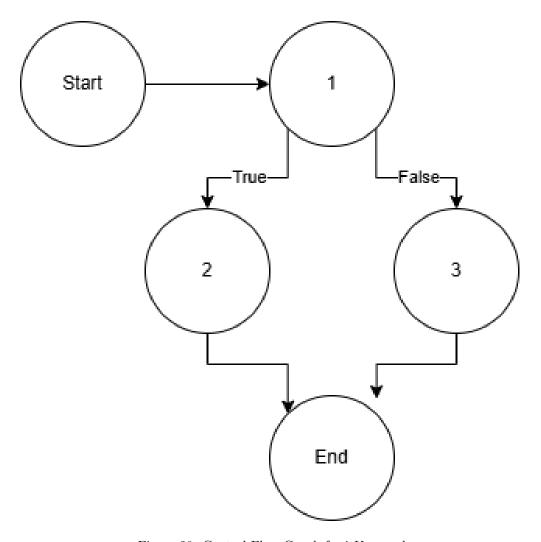


Figure 20: Control Flow Graph for isKeyword

Test Path	Test Data	Expected Output
1, 2	"and"	true
1, 3	"hello"	false

Table 10: Test Cases for isKeyword

ormalsize1.11 Method: isCharConstant

```
static boolean is_char_constant(String str)

{

if (str.length() > 2 || str.charAt(index:0)=='#' && Character.isLetter(str.charAt(index:1)))

return true;
else
| return false;
}
```

Figure 21: Code Snippet for isCharConstant

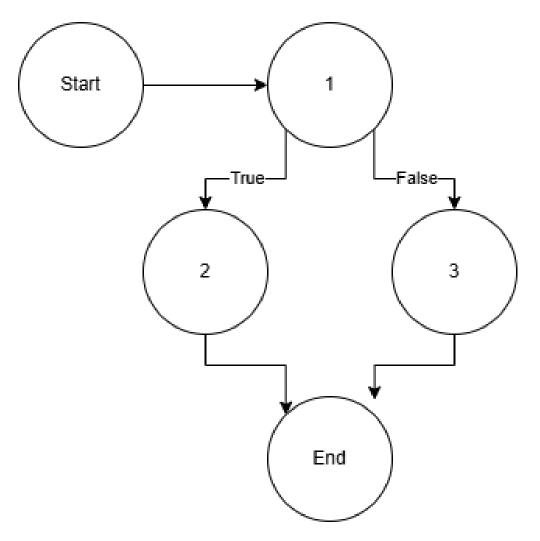


Figure 22: Control Flow Graph for isCharConstant

Test Path	Test Data	Expected Output
1, 2	"#a"	true
1, 3	"abc"	false

Table 11: Test Cases for isCharConstant

${\rm ormalsize 1.12} \quad Method: \ is Num Constant$

```
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' )</pre>
310
                 if(Character.isDigit(str.charAt(i+1)))
311
                  i++;
312
                  else
313
                  return false;
314
                                          /* end WHILE */
315
316
              return true;
317
318
            else
319
             return false;
                                          /* other return FALSE */
320
```

Figure 23: Code Snippet for isNumConstant

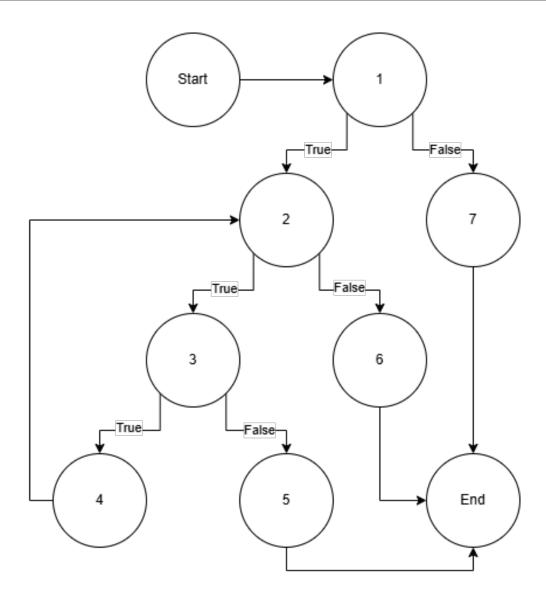


Figure 24: Control Flow Graph for isNumConstant

Test Path	Test Data	Expected Output
1, 7	str = "abc"	false
1, 2, 3, 4, 2, 6	"123"	true
1, 2, 3, 5	"12a"	false
1, 2, 6	"1"	false

Table 12: Test Cases for isNumConstant

ormalsize1.13 Method: isStrConstant

```
static boolean is_str_constant(String str)
327
328
329
             int i=1;
330
             if ( str.charAt(index:0) =='"')
331
               { while (i < str.length() && str.charAt(i)!='\0')
332
                   { if(str.charAt(i)=='"' )
    return true;  /* meet the second '"'
333
                                                                            */
334
335
                      else
336
                     i++;
337
                   }
                                    /* end WHILE */
338
                return true;
339
340
             else
                                /* other return FALSE */
341
              return false;
342
```

Figure 25: Code Snippet for isStrConstant

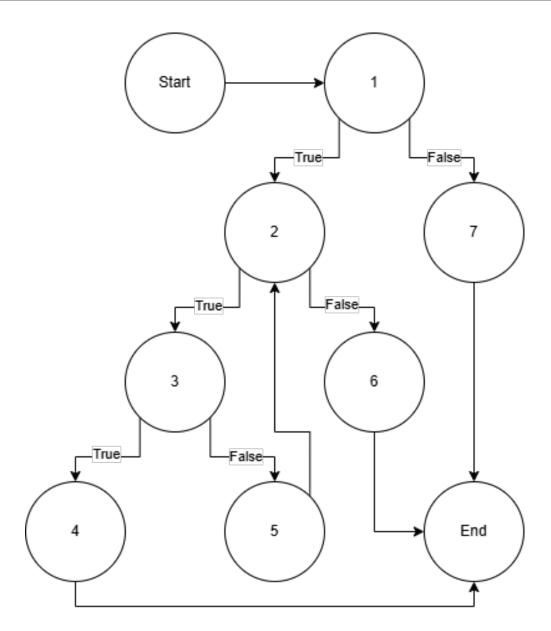


Figure 26: Control Flow Graph for is StrConstant

Test Path	Test Data	Expected Output
1, 7	"abc"	false
1, 2, 3, 4	11 11 11	false
1, 2, 3, 5, 2, 6	""a""	true
1, 2, 6	11 11 11	false

Table 13: Test Cases for ${\tt isStrConstant}$

${\bf ormalsize 1.14} \quad {\bf Method: is Identifier}$

```
349
          static boolean is_identifier(String str)
350
351
            int i=1;
352
            if ( Character.isLetter(str.charAt(index:0)) )
353
354
355
                  \label{eq:while(i < str.length() && str.charAt(i) !='\0' ) /* until meet the end token sign */} \\
356
357
                      if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
358
                      i++;
359
                      else
                      return false;
360
361
                          /* end WHILE */
362
                  return false;
363
364
            else
365
              return true;
366
```

Figure 27: Code Snippet for isIdentifier

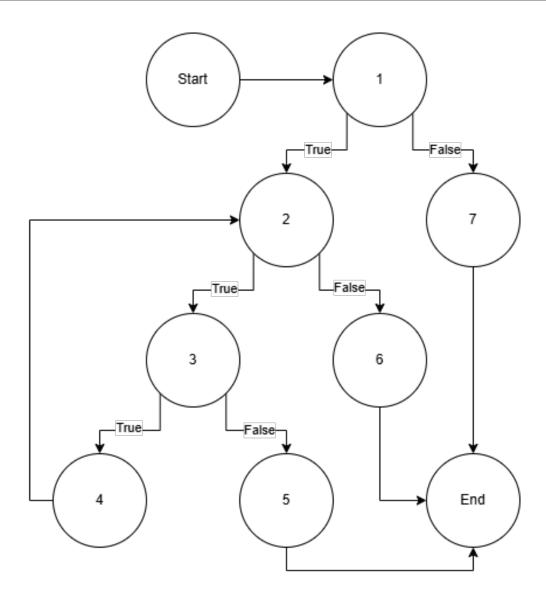


Figure 28: Control Flow Graph for isIdentifier

Test Path	Test Data	Expected Output
1, 7	"1"	false
1, 2, 3, 4, 2, 6	"a1"	true
1, 2, 3, 5	"output"	true
1, 2, 6	"1output"	false

Table 14: Test Cases for isIdentifier

ormalsize1.15 Method: printSpecSymbol

```
376
          static void print_spec_symbol(String str)
377
378
               if
                       (str.equals(anObject:")"))
379
380
                        System.out.print(s:"lparen.\n");
381
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
                        System.out.print(s: "quote.\n");
403
404
                        return;
405
406
               if (str.equals(anObject:"`"))
407
408
                        System.out.print(s:"bquote.\n");
409
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

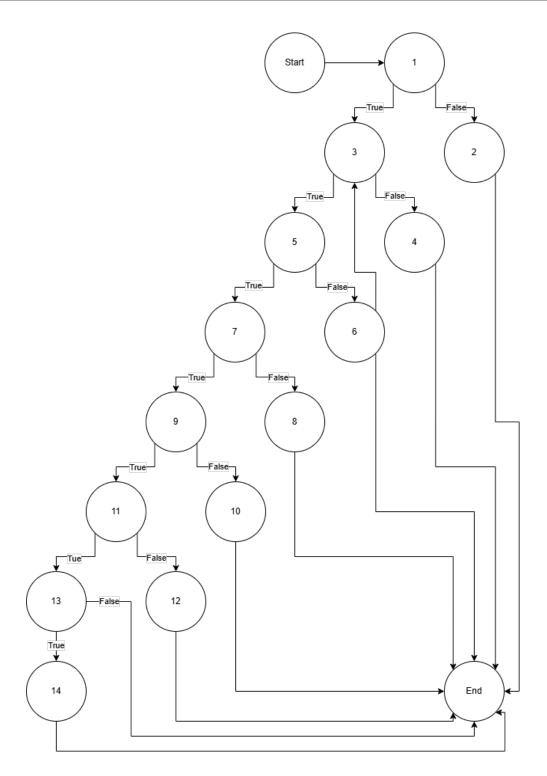


Figure 30: Control Flow Graph for printSpecSymbol

Test Path	Test Data	Expected Output
1, 2	")"	"lparen."
1, 3, 4	"("	"rparen."
1, 3, 5, 6	"["	"lsquare."
1, 3, 5, 7, 8	"]"	"rsquare."
1, 3, 5, 7, 9, 10	11) 11	"quote."
1, 3, 5, 7, 9, 11, 12	11 (11	"bquote."
1, 3, 5, 7, 9, 11, 13	","	"comma."
1, 3, 5, 7, 9, 11, 13, 14	"unknown"	error

Table 15: Test Cases for printSpecSymbol

ormalsize1.16 Method: isSpecSymbol

```
425
          static boolean is_spec_symbol(char c)
426
               if (c == '(')
427
428
429
                   return true;
430
               if (c == ')')
431
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
               if (c == '`')
447
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

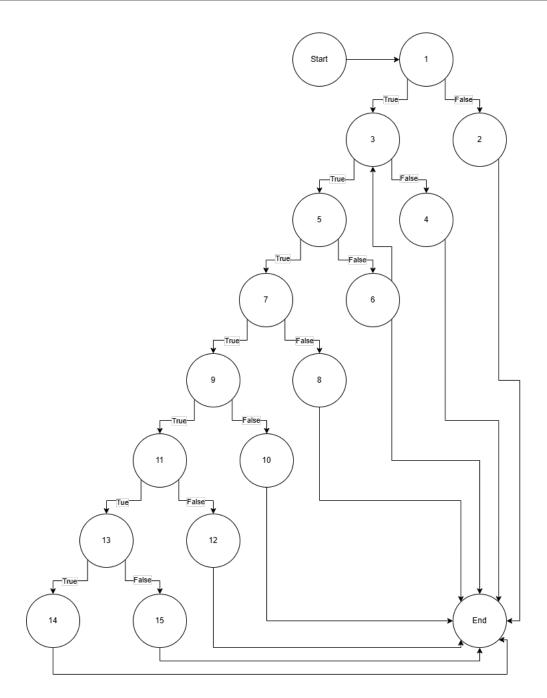


Figure 32: Control Flow Graph for isSpecSymbol

Test Path	Test Data	Expected Output
1, 2	c = '('	true
1, 3, 4	c = ')'	true
1, 3, 5, 6	c = '['	true
1, 3, 5, 7, 8	c = ']'	true
1, 3, 5, 7, 9, 10	c = ','	true
1, 3, 5, 7, 9, 11, 12	c = '(')	true
1, 3, 5, 7, 9, 11, 13, 14	c = ','	true
1, 3, 5, 7, 9, 11, 13, 15	c = 'x'	false

Table 16: Test Cases for isSpecSymbol

2 Program Tests

- Test Case 1
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 3 \rightarrow openCharacterStream[1, 3, 4]], 8 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- **Input:** fname = "file.txt", EOF
- Output: ""
- Test Case 2
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 2 \rightarrow openCharacterStream[1, 2, 4]], 8 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- **Input:** fname = "", EOF
- Output: ""
- Test Case 3
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 3 \rightarrow openCharacterStream[1, 3, 4]], 8 \rightarrow getToken[1, 5 \rightarrow isSpecSymbol[1], 15], 9, 10 \rightarrow printToken[1, 6 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- Input: fname = "file.txt", "("
- Output: "lparen."
- Test Case 4
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 3 \rightarrow openCharacterStream[1, 3, 4]], 8 \rightarrow getToken[1, 6 \rightarrow isKeyword[1], 9 \rightarrow ungetChar[1], 11 \rightarrow isTokenEnd[1, 10 \rightarrow isSpecSymbol[1], 11, 12], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7], 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- Input: fname = "file.txt", "if lambda"
- Output: "keyword," if". keyword, "lambda"."
- Test Case 5
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 2 \rightarrow openCharacterStream[1, 2, 4]], 8 \rightarrow getToken[1, 6 \rightarrow isIdentifier[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 3], 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- **Input:** fname = "", ""abc""
- Output: "identifier," abc"."
- Test Case 6
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 3 \rightarrow openCharacterStream[1, 3, 4]], 8 \rightarrow getToken[1, 5 \rightarrow isSpecSymbol[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 6 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow getToken[1, 5 \rightarrow isSpecSymbol[1], 9 \rightarrow ungetChar[1], 15], 9, 12]
- Input: fname = "file.txt", "()"
- Output: "lparen. rparen."
- Test Case 7
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 3 \rightarrow openCharacterStream[1, 3, 4]], 8 \rightarrow getToken[1, 6 \rightarrow isStrConstant[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 5], 3], 11 \rightarrow getToken[1, 6 \rightarrow isNumConstant[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 4], 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- Input: fname = "file.txt", ""HelloWorld" 123"
- Output: "string," Hello World". numeric, 123."
- Test Case 8
- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 2 \rightarrow openCharacterStream[1, 2, 4]], 8 \rightarrow getToken[1, 6 \rightarrow isNumConstant[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 4], 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]

- **Input:** fname = "", "123"
- Output: "numeric,123."

Test Case 9

- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 2 \rightarrow openCharacterStream[1, 2, 4]], 8 \rightarrow getToken[1, 10 \rightarrow isComment[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7], 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- Input: fname = "", ";comment"
- Output: "comment, ";comment"."

Test Case 10 Test Case 10

- Path: main[1, 2, 6, 7 \rightarrow openTokenStream[1, 2 \rightarrow openCharacterStream[1, 2, 4]], 8 \rightarrow getToken[1, 10 \rightarrow isCharConstant[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 6], 3], 11 \rightarrow getToken[1, 2, 9 \rightarrow ungetChar[1], 15], 9, 12]
- **Input:** fname = "", "#a"
- Output: "character,"a"."

Test Case 11

- Path: main[1, 2, 6, 7]
- $\rightarrow openTokenStream[1,2 \rightarrow openCharacterStream[1,2,4]], 8 \rightarrow getToken[1,6 \rightarrow isKeyword[1],9 \rightarrow ungetChar[1],15], 9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1,3]],11 \rightarrow getToken[1,6 \rightarrow isIdentifier[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,3 \rightarrow isIdentifier[1]],3],11 \rightarrow getToken[1,6 \rightarrow isNumConstant[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,4 \rightarrow isNumConstant[1]],3],11 \rightarrow getToken[1,6 \rightarrow isStrConstant[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,5 \rightarrow isStrConstant[1]],3],11 \rightarrow getToken[1,10 \rightarrow isComment[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isComment[1]],3],11 \rightarrow getToken[1,6 \rightarrow isKeyword[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1],15],11,12]$
- Input: fname = "",
 "if (x 123 "HelloWorld"
 ;comment
 lambda)"
- Output: "keyword," if". lparen. identifier," x". numeric,123. string," HelloWorld". comment,"; comment". keyword, "lambda".

Test Case 12

• **Path:** main[1, 2, 6, 7]

rparen."

 $\rightarrow openTokenStream[1,2 \rightarrow openCharacterStream[1,2,4]], 8 \rightarrow getToken[1,6 \rightarrow isKeyword[1],9 \rightarrow ungetChar[1],15], 9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1,3]],11 \rightarrow getToken[1,6 \rightarrow isKeyword[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,6 \rightarrow isIdentifier[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,3 \rightarrow isIdentifier[1]],3],11 \rightarrow getToken[1,6 \rightarrow isError[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,0],3],11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1,3]],11,12]$

```
• Input: fname = "", "and and j 112A)"
```

• Output: "keyword," and". bquote. keyword," and". identifier," j". error," 112A". rparen."

Test Case 13 • **Path:** main[1, 2, 6, 7]

 $\rightarrow openTokenStream[1,2 \rightarrow openCharacterStream[1,2,4]], 8 \rightarrow getToken[1,10 \rightarrow$ $isComment[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7 \rightarrow tokenType]]$ $isComment[1], 3, 11 \rightarrow getToken[1, 6 \rightarrow isKeyword[1], 9 \rightarrow ungetChar[1], 15, 9, 10 \rightarrow isKeyword[1], 9 \rightarrow ungetChar[1], 15, 9, 10 \rightarrow isKeyword[1], 10 \rightarrow$ $printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,5 \rightarrow tokenType],100 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,5 \rightarrow tokenType],100 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow getToken[1,5 \rightarrow tokenType],100 \rightarrow tokenType[1,7 \rightarrow tokenType],100 \rightarrow to$ $isSpecSymbol[1], 15], 9, 10 \rightarrow printToken[1, 6 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow printSpecSymbol[1, 3], 11 \rightarrow printSp$ $getToken[1,6 \rightarrow isIdentifier[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow isIdentifier[1],10)$ $tokenType[1, 3 \rightarrow isIdentifier[1]], 3], 11 \rightarrow getToken[1, 6 \rightarrow isNumConstant[1], 9 \rightarrow isIdentifier[1], 3 \rightarrow isIdentif$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 4 \rightarrow$ $isNumConstant[1], 3, 11 \rightarrow qetToken[1, 6 \rightarrow isStrConstant[1], 9 \rightarrow$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 5 \rightarrow$ $isStrConstant[1]], 3], 11 \rightarrow getToken[1, 10 \rightarrow isComment[1], 9 \rightarrow$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7 \rightarrow$ $isComment[1]], 3], 11 \rightarrow getToken[1, 6 \rightarrow isKeyword[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow ungetChar[1], 15], 10 \rightarrow ungetChar[1], 10$ $printToken[1,2 \rightarrow tokenType[1,7 \rightarrow isKeyword[1]],3],11 \rightarrow qetToken[1,5 \rightarrow tokenType],100 + tokenType[1,7 \rightarrow tokenType],100 + tokenType[$ $isSpecSymbol[1], 15], 9, 10 \rightarrow printToken[1, 6 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow printSpecSymbol[1, 3], 11 \rightarrow printSp$ $getToken[1,6 \rightarrow isCharConstant[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow isCharConstant[1],9)$ $tokenType[1,6 \rightarrow isCharConstant[1]], 3], 11 \rightarrow getToken[1,6 \rightarrow isStrConstant[1], 9 \rightarrow isCharConstant[1], 9 \rightarrow isCharC$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 5 \rightarrow$ $isStrConstant[1]], 3], 11 \rightarrow getToken[1, 6 \rightarrow isNumConstant[1], 9 \rightarrow isNumConstant[1], 9$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 4 \rightarrow$ $isNumConstant[1]], 3], 11 \rightarrow getToken[1, 5 \rightarrow isSpecSymbol[1], 15], 9, 10 \rightarrow$ $printToken[1,6 \rightarrow printSpecSymbol[1,3]], 11 \rightarrow qetToken[1,6 \rightarrow isIdentifier[1],9 \rightarrow qetToken[1],9 \rightarrow qetT$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 3 \rightarrow$ $isIdentifier[1]], 3], 11 \rightarrow getToken[1, 5 \rightarrow isSpecSymbol[1], 15], 9, 10 \rightarrow$ $printToken[1, 6 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow getToken[1, 10 \rightarrow isComment[1], 9 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow getToken[1, 10 \rightarrow isComment[1], 11 \rightarrow getToken[1, 10 \rightarrow isComment[1], 12 \rightarrow getToken[1, 10 \rightarrow isComment[1], 12 \rightarrow getToken[1, 10 \rightarrow isComment[1], 13 \rightarrow getToken[1], 13 \rightarrow getToken[1, 10 \rightarrow isComment[1], 13 \rightarrow getToken[1, 10 \rightarrow isComment[1], 13 \rightarrow getToken[1], 13 \rightarrow getToke$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7 \rightarrow$ $printToken[1,2 \rightarrow tokenType[1,0],3],11 \rightarrow qetToken[1,6 \rightarrow isError[1],9 \rightarrow token[1,2 \rightarrow tokenType[1,0],3],11 \rightarrow qetToken[1,6 \rightarrow isError[1],9 \rightarrow token[1,6 \rightarrow isError[1],9 \rightarrow token[1],9 \rightarrow token[1,6 \rightarrow isError[1],9 \rightarrow token[1],9 \rightarrow$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 0], 3], 11 \rightarrow getToken[1, 6 \rightarrow tokenType[1, 0], 3], 11 \rightarrow getToken[1, 0], 11 \rightarrow$ $isKeyword[1], 9 \rightarrow ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7 \rightarrow tokenType]]$ $printToken[1,6 \rightarrow printSpecSymbol[1,3]], 11 \rightarrow qetToken[1,5 \rightarrow$ $isSpecSymbol[1], 15], 9, 10 \rightarrow printToken[1, 6 \rightarrow printSpecSymbol[1, 3]], 11 \rightarrow$ $getToken[1,5 \rightarrow isSpecSymbol[1],15], 9,10 \rightarrow printToken[1,6 \rightarrow$ $printSpecSymbol[1,3]], 11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15], 9,10 \rightarrow$ $printToken[1,6 \rightarrow printSpecSymbol[1,3]], 11 \rightarrow getToken[1,10 \rightarrow isComment[1],9 \rightarrow printSpecSymbol[1,3]], 11 \rightarrow getToken[1,3]], 11 \rightarrow getToken[1,3]]$ $ungetChar[1], 15], 9, 10 \rightarrow printToken[1, 2 \rightarrow tokenType[1, 7 \rightarrow$ isComment[1]], 3], 11, 12]

• Input: fname = "", "; this is a comment if (x 123 "hello" ;another comment lambda) #c "string with spaces" 456 ' quote , comma ;comment unknown_token 123abc

```
"unclosed_string
  and []();end_comment"
• Output: "comment,"; this is a comment".
  keyword,"if".
  lparen.
 identifier,"x".
  numeric, 123.
 string,"hello".
  comment,";another comment".
  keyword,"lambda".
  rparen.
  character,'c'.
  string,"string with spaces".
  numeric, 456.
  bquote.
  identifier,"quote".
  comma.
  identifier," comma".
  comment,";comment".
  error,"unknown_token".
  error,"123abc".
  error,""unclosed_string".
  keyword," and".
  lsquare.
  rsquare.
 lparen.
  rparen.
  comment,";end_comment"."
```

Test Case 14 • **Path:** main[1, 2, 6, 7]

 $\rightarrow openTokenStream[1,2 \rightarrow openCharacterStream[1,2,4]], 8 \rightarrow getToken[1,6 \rightarrow isIdentifier[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,3 \rightarrow isIdentifier[1]],3],11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1,3]],11 \rightarrow getToken[1,5 \rightarrow isSpecSymbol[1],15],9,10 \rightarrow printToken[1,6 \rightarrow printSpecSymbol[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,5 \rightarrow isStrConstant[1],9 \rightarrow ungetChar[1],15],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,5],9,10 \rightarrow printToken[1,2 \rightarrow tokenType[1,0],3],11,12]$

- **Input:** fname = "", ""string"; '"extr"a""
- Output: "identifier," string". comment,";". quote. string," extr". error, "a"."