```
 1 package Project_1;
 2
 3 import java.io.BufferedReader;
 4 import java.io.FileNotFoundException;
 5 import java.io.FileReader;
 6 import java.io.IOException;
 7 import java.io.InputStreamReader;
 8
 9 public class Printtokens2 {
10     static int error = 0;
11     static int keyword = 1;
12     static int spec_symbol = 2;
13     static int identifier = 3;
14     static int num_constant = 41;
15     static int str_constant = 42;
16     static int char_constant = 43;
17     static int comment = 5;
18
19     /*********************************************/
20     /* NAME:    open_character_stream            */
21     /* INPUT:       a filename                   */
22     /* OUTPUT:      a BufferedReader             */
23     /* DESCRIPTION: when not given a filename,   */
24     /*              open stdin,otherwise open    */
25     /*              the existed file             */
26
27     /*********************************************/
28     BufferedReader open_character_stream(String fname) {
29         BufferedReader br = null;
30
31         if (fname == null) /*BUG fname.equals(NULL)*/
32         {
33             br = new BufferedReader(new InputStreamReader(System.in));
34         } else {
35             try {
36                 FileReader fr = new FileReader(fname);
37                 br = new BufferedReader(fr);
38             } catch (FileNotFoundException e) {
39                 System.out.print("The file " + fname + " doesn't
   exists\n");
40                 e.printStackTrace();
41             }
42         }
43
44         return br;
45     }
46
47     /*********************************************/
48     /* NAME:        get_char                     */
49     /* INPUT:       a BufferedReader             */
50     /* OUTPUT:      a character              */
51
52     /*********************************************/
53     int get_char(BufferedReader br) {
```

```
54          int ch = 0;
55          try {
56              br.mark(4);         //marks a position, why spot 4? Add
    test case with longer array
57              ch = br.read();
58          } catch (IOException e) {
59              e.printStackTrace();
60          }
61          return ch;
62      }
63
64      /*****************************************************/
65      /* NAME:       unget_char                            */
66      /* INPUT:      a BufferedReader,a character */
67      /* OUTPUT:     a character                           */
68      /* DESCRIPTION: move backward  */
69
70      /*****************************************************/
71      char unget_char(int ch, BufferedReader br) {
72          try {
73              br.reset();             //resets the stream to the current
     mark
74          } catch (IOException e) {
75              e.printStackTrace();
76          }
77          return 0;
78      }
79
80      /*****************************************************/
81      /* NAME:     open_token_stream                       */
82      /* INPUT:        a filename                              */
83      /* OUTPUT:       a BufferedReader              */
84      /* DESCRIPTION: when filename is EMPTY,choice standard  */
85      /*               input device as input source           */
86
87      /*****************************************************/
88      BufferedReader open_token_stream(String fname) {
89          BufferedReader br;
90          if (fname == null)
91              br = open_character_stream(null);
92          else
93              br = open_character_stream(fname);
94          return br;
95      }
96
97      /*****************************************************/
98      /* NAME :    get_token                               */
99      /* INPUT:    a BufferedReader          */
100     /* OUTPUT:       a token string                                */
101     /* DESCRIPTION: according the syntax of tokens,dealing  */
102     /*               with different case  and get one token  */
103
104     /*****************************************************/
105     String get_token(BufferedReader br) {
```

```
106         int i = 0, j; // bug i and j are never used
107         int id = 0;
108         int res = 0;
109         char ch = '\0';
110 
111         //Creates a new builder with a capacity of 16 characters.
    Even if string is 'Hello', length() will
112         //still return 16, 0-15.
113         StringBuilder sb = new StringBuilder();
114 
115         try {
116             //get_char returns a char while rest is an int
117             res = get_char(br);
118             //can i return -1 from get_Char?
119             if (res == -1) {
120                 return null;
121             }
122             ch = (char) res;
123             while (ch == ' ' || ch == '\n' || ch == '\r')       /*
    strip all blanks until meet characters */ {
124                 res = get_char(br);
125                 ch = (char) res;
126             }
127 
128             if (res == -1) {
129                 System.out.println("ch is: " + ch + "res is: "+res);
130                 return null;
131             }
132 
133             sb.append(ch);
134 
135             if (is_spec_symbol(ch) == true) {
136                 return sb.toString();
137             }
138             if (ch == '"') id = 1;     /* BUG ID WAS 2   prepare for
    string */
139             if (ch == ';') id = 2;     /* BUG ID WAS 1   prepare for
    comment */
140 
141             res = get_char(br);
142             if (res == -1) {
143                 unget_char(ch, br);
144                 //BUG, ADDED UNGET_ERROR INCASE CH == \0
145                 if(ch == '\0') {
146                     unget_error(br);
147                 }
148                 return sb.toString();
149             }
150             ch = (char) res;
151 
152             while (is_token_end(id, res) == false)/* until meet the
    end character */ {
153                 System.out.println("Start is_token_End");
154                 sb.append(ch);
```

```
155                 br.mark(4);
156                 res = get_char(br);
157                 ch = (char) res;
158             }
159             //This will never reach due to -1 being returned null
    above
160             if (res == -1)       /* if end character is eof token
        */ {
161                 unget_char(ch, br);        /* then put back eof on
    token_stream */
162                 //BUG, ADDED UNGET_ERROR INCASE CH == \0
163                 if(ch == '\0') {
164                     unget_error(br);
165                 }
166                 return sb.toString();
167             }
168 
169             if (is_spec_symbol(ch) == true)     /* if end character
    is special_symbol */ {
170                 unget_char(ch, br);        /* then put back this
    character       */
171                 //BUG, ADDED UNGET_ERROR INCASE CH == \0
172                 if(ch == '\0') {
173                     unget_error(br);
174                 }
175                 return sb.toString();
176             }
177             if (id == 1)                  /* if end character is "
    and is string */ {
178                 sb.append(ch);
179                 return sb.toString();
180             }
181             if (id == 0 && ch == 59)
182             {
183                 unget_char(ch, br);       /* then put back this
    character         */
184                 //BUG, ADDED UNGET_ERROR INCASE CH == \0
185                 if(ch == '\0') {
186                     unget_error(br);
187                 }
188                 return sb.toString();
189             }
190         } catch (IOException e) {
191             e.printStackTrace();
192         }
193 
194         return sb.toString();                   /* return nomal case
    token             */
195     }
196 
197     /*****************************************************/
198     /* NAME:    is_token_end                          */
199     /* INPUT:       a character,a token status             */
200     /* OUTPUT:  a BOOLEAN value                       */
```

```
201
202     /*****************************************************/
203     static boolean is_token_end(int str_com_id, int res) {
204         if (res == -1) return (true); /* BUG - unreachable is eof
    token? */
205         char ch = (char) res;
206         if (str_com_id == 1)          /* is string token */ {
207             if (ch == '"' | ch == '\n' || ch == '\r')   /* for string
     until meet another " */
208                 return true;
209             else
210                 return false;
211         }
212
213         if (str_com_id == 2)    /* is comment token */ {
214             if (ch == '\n' || ch == '\r') //BUG "|| ch == '\t'" tab
    is not end of token    /* for comment until meet end of line */
215                 return true;
216             else
217                 return false;
218         }
219
220         if (is_spec_symbol(ch) == true) return true; /* is
    special_symbol? */
221         if (ch == ' ' || ch == '\n' || ch == '\r' || ch == 59) return
     true;
222
223         return false;               /* other case,return FALSE */
224     }
225
226     /*****************************************************/
227     /* NAME :   token_type                               */
228     /* INPUT:       a token              */
229     /* OUTPUT:      an integer value                     */
230     /* DESCRIPTION: the integer value is corresponding   */
231     /*              to the different token type          */
232
233     /*****************************************************/
234     static int token_type(String tok) {
235         if (is_keyword(tok)) return (keyword);
236         if (is_spec_symbol(tok.charAt(0))) return (spec_symbol);
237         if (is_identifier(tok)) return (identifier);
238         if (is_num_constant(tok)) return (num_constant);
239         if (is_str_constant(tok)) return (str_constant);
240         if (is_char_constant(tok)) return (char_constant);
241         if (is_comment(tok)) return (comment);
242         return (error);                     /* else look as error
    token */
243     }
244
245     /*****************************************************/
246     /* NAME:    print_token                              */
247     /* INPUT:   a token                                  */
248
```

```
249     /*****************************************************/
250     void print_token(String tok) {
251         int type;
252         type = token_type(tok);
253         if (type == error) {
254             System.out.print("error,\"" + tok + "\".\n");
255         }
256 
257         if (type == keyword) {
258             System.out.print("keyword,\"" + tok + "\".\n");
259         }
260 
261         if (type == spec_symbol) print_spec_symbol(tok);
262         if (type == identifier) {
263             System.out.print("identifier,\"" + tok + "\".\n");
264         }
265         if (type == num_constant) {
266             System.out.print("numeric," + tok + ".\n");
267         }
268         if (type == str_constant) {
269             System.out.print("string," + tok + ".\n");
270         }
271         if (type == char_constant) {
272             System.out.print("character,\"" + tok.charAt(1) + "\".\n"
    );
273         }
274 
275     }
276 
277     /* the code for tokens judgment function */
278 
279 
280     /*************************************/
281     /* NAME:    is_comment           */
282     /* INPUT:   a token */
283     /* OUTPUT:      a BOOLEAN value      */
284 
285     /*************************************/
286     static boolean is_comment(String ident) {
287         if (ident.charAt(0) == 59)   /* the ; char is u0059   */
288             return true;
289         else
290             return false;
291     }
292 
293     /*************************************/
294     /* NAME:    is_keyword           */
295     /* INPUT:   a token */
296     /* OUTPUT:      a BOOLEAN value      */
297 
298     /*************************************/
299     static boolean is_keyword(String str) {
300         if (str.equals("and") || str.equals("or") || str.equals("if")
    ||
```

```
301                 str.equals("xor") || str.equals("lambda") || str.
    equals("=>"))
302             return true;
303         else
304             return false;
305     }
306 
307     /*************************************/
308     /* NAME:    is_char_constant     */
309     /* INPUT:   a token */
310     /* OUTPUT:      a BOOLEAN value      */
311 
312     /*************************************/
313     static boolean is_char_constant(String str) {
314         if (str.length() == 2 && str.charAt(0) == '#' && Character.
    isLetter(str.charAt(1))) //BUG SHOULD BE ==
315             return true;
316         else
317             return false;
318     }
319 
320     /*************************************/
321     /* NAME:    is_num_constant      */
322     /* INPUT:   a token */
323     /* OUTPUT:      a BOOLEAN value      */
324     /*************************************/
325     static boolean is_num_constant(String str) {
326         int i = 1;
327 
328         if (Character.isDigit(str.charAt(0))) {
329             while (i < str.length() && str.charAt(i) != '\0')    /*
    until meet token end sign */ {
330                 if (Character.isDigit(str.charAt(i)))    //BUG was at
     (i+1)
331                     i++;
332                 else
333                     return false;
334             }                           /* end WHILE */
335             return true;
336         } else
337             return false;               /* other return FALSE */
338     }
339 
340     /*************************************/
341     /* NAME:    is_str_constant      */
342     /* INPUT:   a token */
343     /* OUTPUT:      a BOOLEAN value      */
344     /*************************************/
345     static boolean is_str_constant(String str)
346     {
347         int i=1;
348 
349         if ( str.charAt(0) =='"')
350         { while (i < str.length() && str.charAt(0)!='\0')  /* until
```

```
350 meet the token end sign */
351         { if(str.charAt(i)=='"')
352             return true;        /* meet the second '"'           */
353         else
354             i++;
355         }               /* end WHILE */
356             return false; /*BUG was true*/
357         }
358         else
359             return false;       /* other return FALSE */
360     }
361
362     /************************************/
363     /* NAME:    is_identifier         */
364     /* INPUT:   a token */
365     /* OUTPUT:      a BOOLEAN value      */
366
367     /************************************/
368     static boolean is_identifier(String str) {
369         int i = 1;
370
371         if (Character.isLetter(str.charAt(0))) {
372             while (i < str.length() && str.charAt(i) != '\0')    /*
    until meet the end token sign */ {
373                 if (Character.isLetter(str.charAt(i)) || Character.
    isDigit(str.charAt(i)) || str.charAt(i) == '_' || str.charAt(i) == '$
    ' ) //Bug, added _ $ for identifiers
374                     i++;
375                 else
376                     return false;
377             }      /* end WHILE */
378             return true;
379         } else
380             return false;
381     }
382
383     /****************************************/
384     /* NAME:    unget_error                 */
385     /* INPUT:      a BufferedReader */
386     /* OUTPUT:  print error message       */
387
388     /****************************************/
389     static void unget_error(BufferedReader br) {
390         System.out.print("It can not get charcter\n");
391     }
392     //Bug - never called
393     /************************************************/
394     /* NAME:        print_spec_symbol                */
395     /* INPUT:       a spec_symbol token */
396     /* OUTPUT :     print out the spec_symbol token  */
397     /*              according to the form required   */
398
399     /************************************************/
400     static void print_spec_symbol(String str) {
```

```
401         if (str.equals("(")) {
402 
403             System.out.print("lparen.\n");
404             return;
405         }
406         if (str.equals(")")) {
407 
408             System.out.print("rparen.\n");
409             return;
410         }
411         if (str.equals("[")) {
412             System.out.print("lsquare.\n");
413             return;
414         }
415         if (str.equals("]")) {
416 
417             System.out.print("rsquare.\n");
418             return;
419         }
420         if (str.equals("'")) {
421             System.out.print("quote.\n");
422             return;
423         }
424         if (str.equals("`")) {
425 
426             System.out.print("bquote.\n");
427             return;
428         }
429 
430         System.out.print("comma.\n");
431     }
432 
433     /*************************************/
434     /* NAME:         is_spec_symbol      */
435     /* INPUT:        a token */
436     /* OUTPUT:       a BOOLEAN value     */
437 
438     /*************************************/
439     static boolean is_spec_symbol(char c) {
440         if (c == '(') {
441             return true;
442         }
443         if (c == ')') {
444             return true;
445         }
446         if (c == '[') {
447             return true;
448         }
449         if (c == ']') {
450             return true;
451         }
452         if (c == '\'') {
453             return true;
454             //bug - not in print_spec_symbols
```

```
455         }
456         if (c == '`') {
457             return true;
458         }
459         if (c == ',') {
460             return true;
461         }
462         return false;     /* others return FALSE */
463     }
464 
465     public static void main(String[] args) throws IOException {
466         String fname = null;
467         if (args.length == 0) { /* if not given filename,take as
    '"'" */
468             fname = new String();
469         } else if (args.length == 1) {
470             System.out.print(args[0]); //BUG
471             fname = args[0];
472         } else {
473             System.out.print("Error!,please give the token stream\n")
    ;
474             System.exit(0);
475         }
476         Printtokens2 t = new Printtokens2();
477         BufferedReader br = t.open_token_stream(fname); /* open token
    stream */
478         String tok = t.get_token(br);
479         while (tok != null) {   /* take one token each time until eof
    */
480             t.print_token(tok);
481             tok = t.get_token(br);
482         }
483 
484 
485         System.exit(0);
486     }
487 }
488 
```