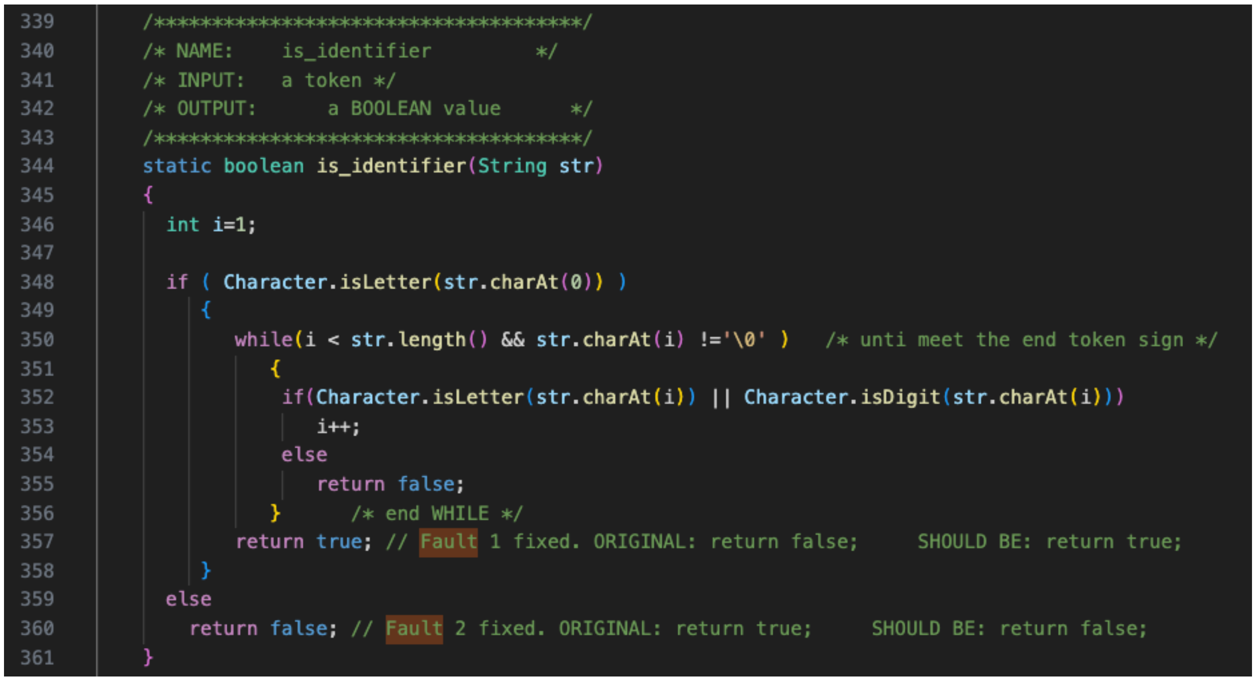
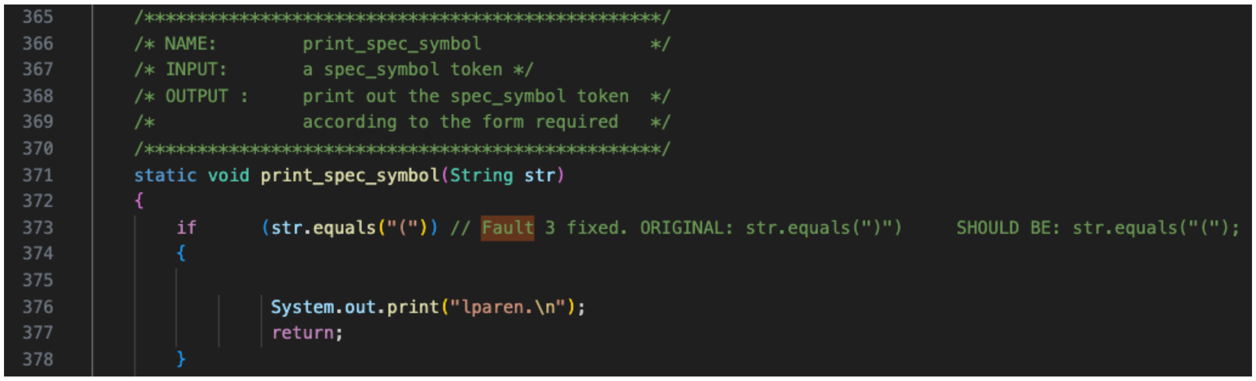
Software Testing Project Faults and Corrections

Uriel Lujan & Elvin Palushi



```
339     /***************************************/
340     /* NAME:    is_identifier         */
341     /* INPUT:   a token */
342     /* OUTPUT:      a BOOLEAN value      */
343     /***************************************/
344     static boolean is_identifier(String str)
345     {
346       int i=1;
347
348       if ( Character.isLetter(str.charAt(0)) )
349          {
350             while(i < str.length() && str.charAt(i) !='\0' )   /* unti meet the end token sign */
351                {
352                 if(Character.isLetter(str.charAt(i)) || Character.isDigit(str.charAt(i)))
353                    i++;
354                 else
355                    return false;
356                }      /* end WHILE */
357             return true; // Fault 1 fixed. ORIGINAL: return false;     SHOULD BE: return true;
358          }
359       else
360         return false; // Fault 2 fixed. ORIGINAL: return true;     SHOULD BE: return false;
361     }
```

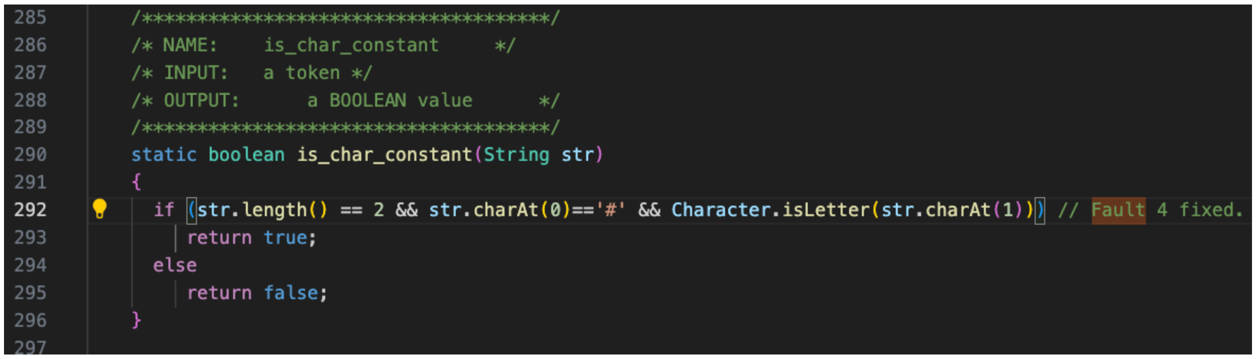
Here, the faults we're returning incorrect Boolean values. If we returned False for line 357, true identifiers would not be marked as identifiers and for line 360, the opposite would be true where anything else by default would be an identifier. We made sure to correct these faults by changing them to the correct Boolean values.



```
365     /**************************************************/
366     /* NAME:        print_spec_symbol               */
367     /* INPUT:       a spec_symbol token */
368     /* OUTPUT :     print out the spec_symbol token  */
369     /*              according to the form required   */
370     /**************************************************/
371     static void print_spec_symbol(String str)
372     {
373         if      (str.equals("(")) // Fault 3 fixed. ORIGINAL: str.equals(")")     SHOULD BE: str.equals("(");
374         {
375
376                 System.out.print("lparen.\n");
377                 return;
378         }
```

For fault 3 we noticed that the system.out statement had lparen in quotes and this clearly means left parenthesis so we corrected the ')' to a '('.

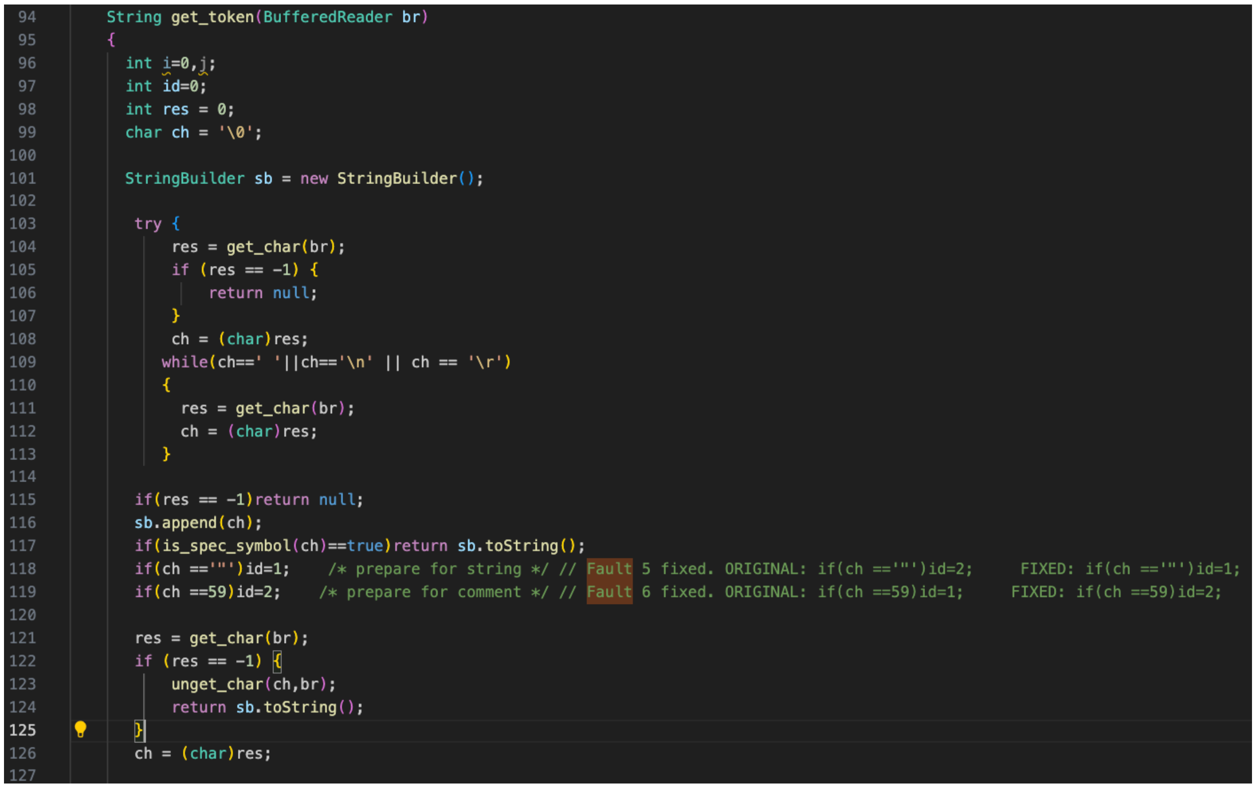
Software Testing Project Faults and Corrections
Uriel Lujan & Elvin Palushi



```java
/*****************************************/
/* NAME:    is_char_constant     */
/* INPUT:   a token */
/* OUTPUT:      a BOOLEAN value      */
/*****************************************/
static boolean is_char_constant(String str)
{
  if (str.length() == 2 && str.charAt(0)=='#' && Character.isLetter(str.charAt(1))) // Fault 4 fixed.
     return true;
  else
     return false;
}
```

```
// Fault 4 fixed. ORIGINAL: str.length() > 2 || str.charAt(0)=='#'      FIXED:
str.length() == 2 && str.charAt(0)=='#'
```

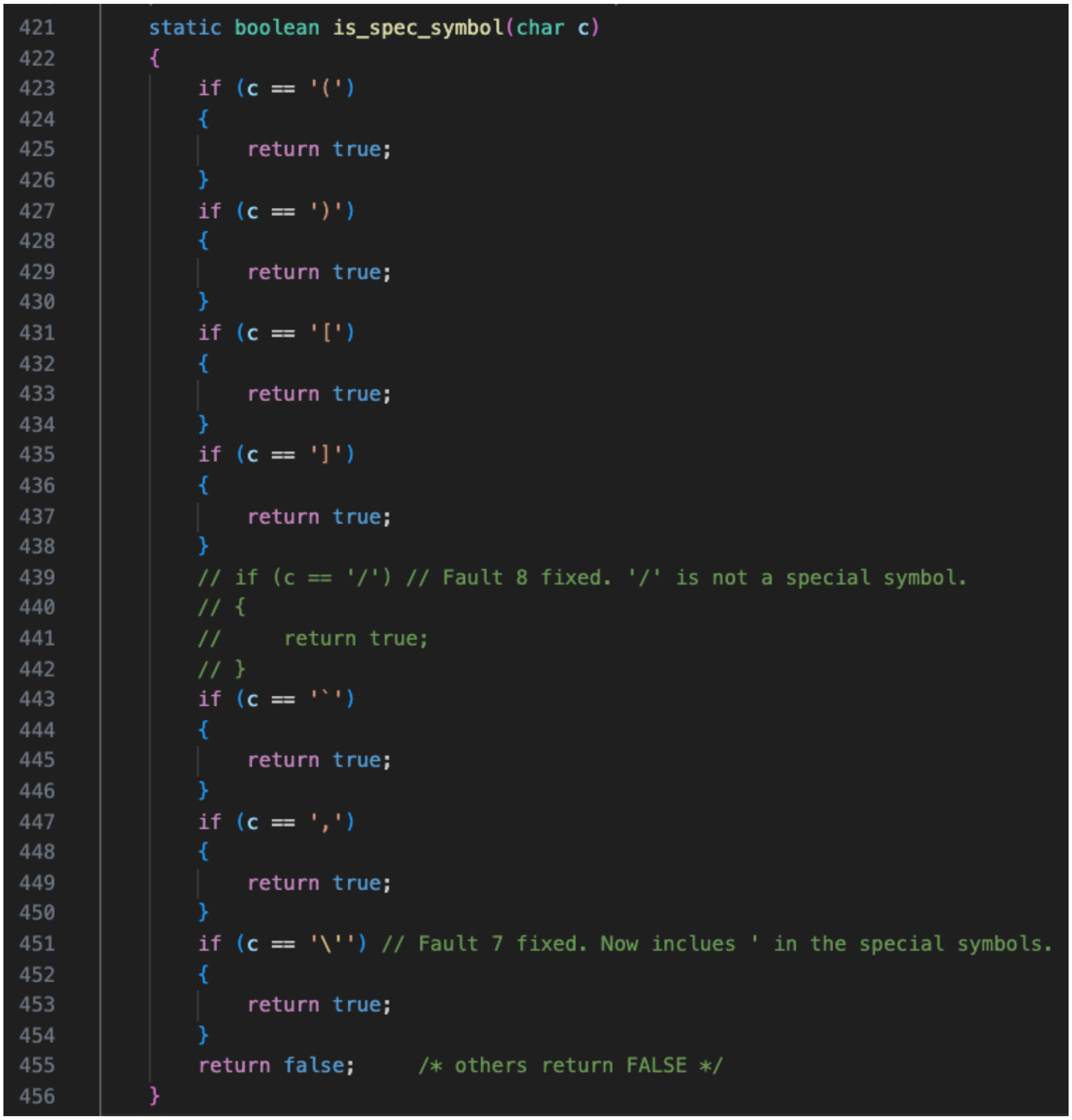
For fault 4 the if statement contains an OR when it should contain an AND statement. Character constants should be two characters, and the original was checking if it was 3 characters or greater. The first character is a # and the second character is a letter. Both must be true to be a char constant.



```java
String get_token(BufferedReader br)
{
  int i=0,j;
  int id=0;
  int res = 0;
  char ch = '\0';

  StringBuilder sb = new StringBuilder();

   try {
      res = get_char(br);
      if (res == -1) {
         return null;
      }
      ch = (char)res;
     while(ch==' '||ch=='\n' || ch == '\r')
     {
       res = get_char(br);
       ch = (char)res;
     }

   if(res == -1)return null;
   sb.append(ch);
   if(is_spec_symbol(ch)==true)return sb.toString();
   if(ch =='"')id=1;    /* prepare for string */ // Fault 5 fixed. ORIGINAL: if(ch =='"')id=2;     FIXED: if(ch =='"')id=1;
   if(ch ==59)id=2;    /* prepare for comment */ // Fault 6 fixed. ORIGINAL: if(ch ==59)id=1;     FIXED: if(ch ==59)id=2;

   res = get_char(br);
   if (res == -1) {
      unget_char(ch,br);
      return sb.toString();
   }
   ch = (char)res;
```

For fault 5 and 6 we can see that the id’s are switched when they should not be.

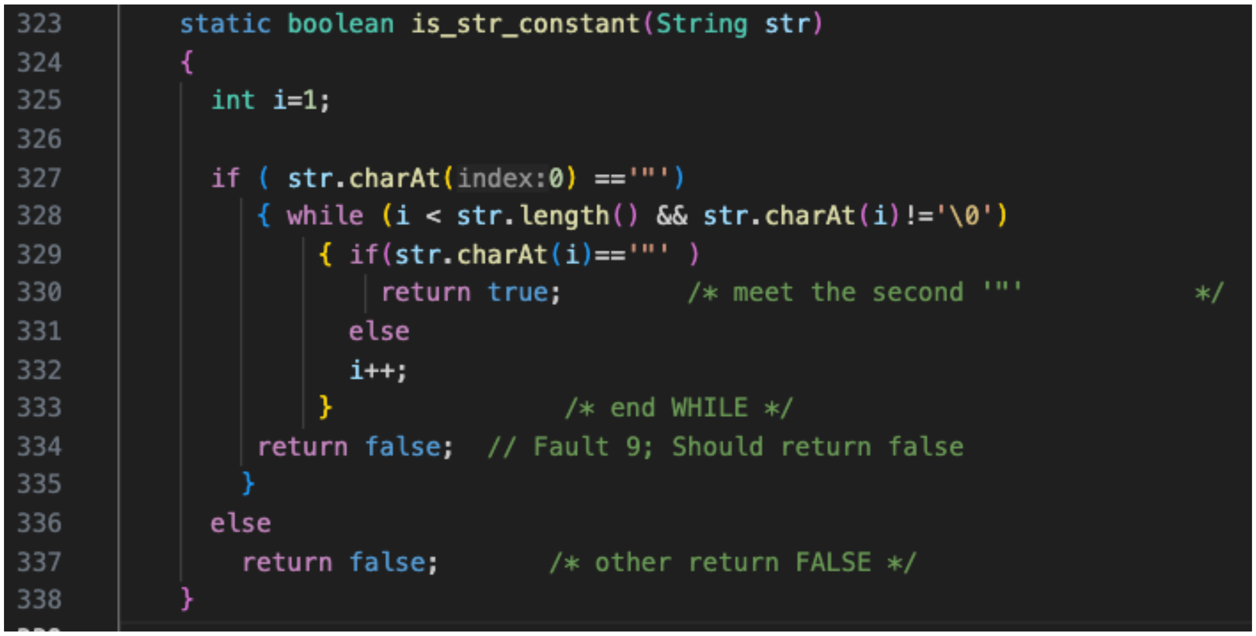
Software Testing Project Faults and Corrections
Uriel Lujan & Elvin Palushi



```
static boolean is_spec_symbol(char c)
{
    if (c == '(')
    {
        return true;
    }
    if (c == ')')
    {
        return true;
    }
    if (c == '[')
    {
        return true;
    }
    if (c == ']')
    {
        return true;
    }
    // if (c == '/') // Fault 8 fixed. '/' is not a special symbol.
    // {
    //     return true;
    // }
    if (c == '`')
    {
        return true;
    }
    if (c == ',')
    {
        return true;
    }
    if (c == '\'') // Fault 7 fixed. Now inclues ' in the special symbols.
    {
        return true;
    }
    return false;     /* others return FALSE */
}
```

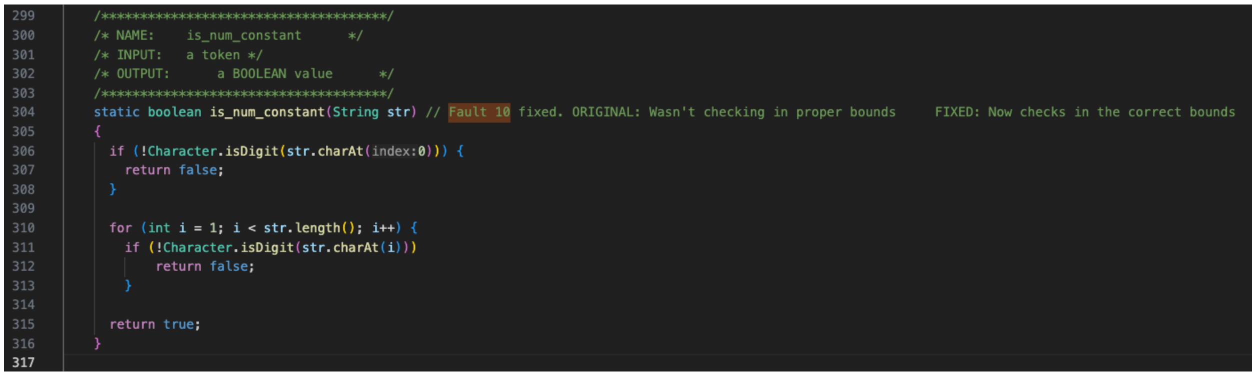
For faults 7 and 8: For fault 8 ‘/’ is not considered a special symbol and as for fault 7, the quote was missing from the list of special symbols.

Software Testing Project Faults and Corrections
Uriel Lujan & Elvin Palushi



```
static boolean is_str_constant(String str)
{
  int i=1;

  if ( str.charAt(index:0) =='"')
    { while (i < str.length() && str.charAt(i)!='\0')
        { if(str.charAt(i)=='"' )
            return true;         /* meet the second '"'           */
          else
          i++;
        }                /* end WHILE */
      return false;  // Fault 9; Should return false
    }
  else
    return false;       /* other return FALSE */
}
```

For Fault 9 originally it was returning true when it should have returned false because if it ran out of characters to loop through, then it would break out of the while loop.



```
/***************************************/
/* NAME:    is_num_constant      */
/* INPUT:   a token */
/* OUTPUT:      a BOOLEAN value      */
/***************************************/
static boolean is_num_constant(String str) // Fault 10 fixed. ORIGINAL: Wasn't checking in proper bounds     FIXED: Now checks in the correct bounds
{
  if (!Character.isDigit(str.charAt(index:0))) {
    return false;
  }

  for (int i = 1; i < str.length(); i++) {
    if (!Character.isDigit(str.charAt(i)))
        return false;
    }

  return true;
}
```

Fault 10 was discovered not checking proper bounds but this has now been corrected by checking if the digit is there in the string.