File - C:\Users\Damian\Dropbox\Projects & Academia\Classes\CSE 4321\Project_1\src\test\java\Project_1\Tests.java

```
 1 package Project_1;
 2
 3 import static org.junit.Assert.assertEquals;
 4 import static org.junit.Assert.assertFalse;
 5 import static org.junit.Assert.assertNull;
 6 import static org.junit.Assert.assertTrue;
 7 import static org.junit.Assert.fail;
 8
 9 import java.io.BufferedReader;
10 import java.io.ByteArrayInputStream;
11 import java.io.IOException;
12 import java.io.InputStreamReader;
13 import java.util.Arrays;
14 import java.util.List;
15
16 import org.junit.Before;
17 import org.junit.Rule;
18 import org.junit.Test;
19 import org.junit.contrib.java.lang.system.ExpectedSystemExit;
20 import org.junit.rules.ExpectedException;
21 import static org.mockito.Mockito.*;
22
23
24 public class Tests{
25     @Rule
26     public final ExpectedException exception = ExpectedException.none(
   );
27     @Rule
28     public final ExpectedSystemExit exit = ExpectedSystemExit.none();
29
30     private final static String DEFAULT_TEST_FILE = "testFile0.txt";
31     private final static List<String> TEST_FILE_LIST = Arrays.asList("
   testFile0.txt", "testFile1.txt",
32             "testFile2.txt", "testFile3.txt", "testFile4.txt", "
   testFile5.txt", "testFile6.txt", "testFile7.txt",
33             "testFile8.txt", "testFile9.txt", "testFile10.txt");
34     private final static List<String> TEST_CHAR_LIST = Arrays.asList("
   a", "b", "c", "1", "2", "A", "B", "C",
35             "3", "!", "@", "#", "$", "%", "^", "&", "*", "(", ")", "-"
   , "_", "=", "+", "`", "~", "[", "{", "]",
36             "}", ";", ":", "'", ",", "<", ".", ">", "/", "?", "\\",
   "|", "\t", "\r", "\n", "\0", " ", "\b", "\f",
37             "and", "or", "if", "xor","lambda","=>","#a");
38     private Printtokens2 pt2;
39
40     @Before
41     //Run before every test is run for generic steps: assign test
   variables
42     //or create objects
43     public void setup() {
44         pt2 = new Printtokens2();
45     }
46
47     @Test
```

```
48      // If open_character_stream doesn't receive a filename,
49      // it should open stdin
50      public void test_open_character_stream_no_filename() {
51          BufferedReader br = pt2.open_character_stream(null);
52          assertTrue(br != null);
53      }
54
55      @Test
56      //Using assertTrue, if line is expected, pass True. else, fail
57      public void test_open_character_stream_file_exists() {
58          try {
59              BufferedReader br = pt2.open_character_stream(
    DEFAULT_TEST_FILE);
60              assertTrue(br.readLine().compareTo("Test File") == 0);
61          } catch (IOException e) {
62              System.out.println(e);
63              fail();
64          }
65      }
66
67      @Test
68      public void test_open_character_stream_file_does_not_exist() {
69          BufferedReader br = pt2.open_character_stream("nonexistant");
70          assertNull(br);
71      }
72
73      @Test
74      public void test_get_char() {
75          byte [] bytes = {(byte) 'a'};
76          BufferedReader br = new BufferedReader(new InputStreamReader(
    new ByteArrayInputStream(bytes)));
77          assertEquals(pt2.get_char(br), (char)((byte)'a'));
78      }
79
80      @Test
81      public void test_get_char_io_exception() throws IOException {
82          BufferedReader br = mock(BufferedReader.class);
83          doThrow(new IOException()).when(br).read();
84          pt2.get_char(br);
85      }
86
87      @Test
88      public void test_get_char_empty() {
89          byte [] bytes = {};
90          BufferedReader br = new BufferedReader(new InputStreamReader(
    new ByteArrayInputStream(bytes)));
91          assertEquals(pt2.get_char(br), -1);
92      }
93
94
95      @Test
96      public void test_unget_char_io_exception() throws IOException {
97          BufferedReader br = mock(BufferedReader.class);
98          doThrow(new IOException()).when(br).reset();
```

```
 99          pt2.unget_char(0, br);
100      }
101
102      @Test
103      public void test_open_token_stream_null() {
104          BufferedReader br = pt2.open_token_stream(null);
105          assertTrue(br != null);
106      }
107
108      @Test
109      public void test_open_token_stream() {
110          BufferedReader br = pt2.open_token_stream(DEFAULT_TEST_FILE);
111          assertTrue(br != null);
112      }
113
114      @Test
115      public void test_unget_error() {
116          byte [] bytes = {};
117          BufferedReader br = new BufferedReader(new InputStreamReader(
    new ByteArrayInputStream(bytes)));
118          pt2.unget_error(br);
119      }
120
121      @Test
122      public void test_string_get_token() {
123          for(String testFile: TEST_FILE_LIST) {
124              String result = pt2.get_token(pt2.open_token_stream(
    testFile));
125              if(result != null) {
126                  assertFalse(result.equals(""));
127              }
128              else {
129                  assertNull(result);
130              }
131          }
132      }
133
134      @Test
135      public void test_print_token() {
136          for(String testString: TEST_CHAR_LIST) {
137              pt2.print_token(testString);
138          }
139      }
140
141      @Test
142      public void test_token_type() {
143          for(String testString: TEST_CHAR_LIST) {
144              Integer result;
145              result = pt2.token_type(testString);
146              System.out.print(result);
147          }
148      }
149
150      @Test
```

```
151     public void test_main_file_as_arg() throws IOException {
152         exit.expectSystemExitWithStatus(0);
153         for(String testFile: TEST_FILE_LIST) {
154             pt2.main(new String[]{testFile});
155         }
156     }
157 
158     @Test
159     public void test_main_no_args() throws IOException {
160         try {
161             pt2.main(new String[]{});
162         }
163         catch(NullPointerException e) {
164             // Expecting this to happen
165         }
166     }
167 
168     @Test
169     public void test_main_too_many_args() throws IOException {
170         exit.expectSystemExitWithStatus(0);
171         pt2.main(new String[] {"testFile0.txt", "testFile0.txt", "
    testFile0.txt"});
172     }
173 
174     //this may be a bug, why does it check for # at index 0?
175     @Test
176     public void test_is_char_constant(){
177         String str = "#h";
178         assertTrue(pt2.is_char_constant(str));
179     }
180 
181     @Test
182     public void test_is_num_constant(){
183         String str = "12345a";
184         assertEquals(pt2.is_num_constant(str), false);
185     }
186 
187     //possible bug. should return true.
188     @Test
189     public void test1_is_num_constant(){
190         String str = "12345";
191         assertEquals(pt2.is_num_constant(str), true);
192     }
193 
194     //possible bug. should return true
195     @Test
196     public void test_is_str_constant(){
197         String str = "\"string\"";
198         assertEquals(pt2.is_str_constant(str),true);
199     }
200 
201     @Test
202     public void test_is_str_constant_alt(){
203         String str = "\"string";
```

```
204         assertEquals(pt2.is_str_constant(str),false);
205     }
206 
207     @Test
208     public void test_false_is_str_constant(){
209         String str = "S23ring";
210         assertEquals(pt2.is_str_constant(str),false);
211     }
212 
213     @Test
214     public void test_false_is_comment(){
215         String str = ";test";
216         assertEquals(pt2.is_comment(str),true);
217     }
218 
219     @Test
220     public void test_true_is_char_constant(){
221         String str = "#A";
222         assertEquals(pt2.is_char_constant(str),true);
223     }
224 
225     @Test
226     public void test_false_is_char_constant(){
227         String str = "$test";
228         assertEquals(pt2.is_char_constant(str),false);
229     }
230 
231     @Test
232     public void test_empty_string_is_char_constant(){
233         String str = "";
234         assertEquals(pt2.is_char_constant(str),false);
235     }
236 
237     @Test
238     public void test_is_token_end(){
239         int com_id = 1;
240         int res = -1;
241         assertEquals(pt2.is_token_end(com_id,res), true);
242     }
243 }
244 
245 
246 
```