



دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

آزمایش سوم

مهندسی نرم افزار

محمد رضا دولتی
۹۷۱۱۰۴۱۱

محمد مهدی به نصر
۹۷۱۰۵۷۹۳

۱۴۰۲ فروردین

در اولین قدم پروژه را config کردیم و برای initial config ها را depeenency کردیم و برای build را initial config کردیم و برای source target فایل pom را سمت قسمت source و target می کنیم:

```

Activities Terminal 21:30 14 آوريل en
nvim ./ README.md
nvim ./ README.md
# SE_LAB - آزمایش مهندسی سیستم
**m (use "git restore <file>..." to discard changes in working directory)
modified: README.md
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  json-simple/
no changes added to commit (use "git add" and/or "git commit -a")
→ SE_LAB git:(master) X git add README.md
→ SE_LAB git:(master) X git commit -n 'feature: add EXPT3'
[master 1eefc9a] feature: add EXPT3
  1 file changed, 1 insertion(+)
→ SE_LAB git:(master) X git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 307.00 KiB | 307.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:mbehnrasr/SE_LAB.git
  8b8dc9a..0bfef5f master -> master
→ SE_LAB git:(master) X sudo update-alternatives --config java
[sudo] password for mbehnrasr:
There are 4 choices for the alternative java (providing /usr/bin/java).
Selection Path Priority Status
----- -
0 /usr/lib/jvm/java-18-openjdk-amd64/bin/java 1811 auto mode
1 /usr/lib/jvm/java-11-openjdk-amd64/bin/java 1111 manual mode
* 2 /usr/lib/jvm/java-17-openjdk-amd64/bin/java 1711 manual mode
3 /usr/lib/jvm/java-18-openjdk-amd64/bin/java 1811 manual mode
4 /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java 1801 manual mode

Press <enter> to keep the current choice[*], or type selection number: 0
update-alternatives: using /usr/lib/jvm/java-18-openjdk-amd64/bin/java to provide /usr/bin/java (java) in auto mode
→ SE_LAB git:(master) X ls
json-simple README.md
→ SE_LAB git:(master) X java --version
openjdk 18.0.2-ea 2022-07-19
OpenJDK Runtime Environment (build 18.0.2-ea+9-Ubuntu-222.04)
OpenJDK 64-Bit Server VM (build 18.0.2-ea+9-Ubuntu-222.04, mixed mode, sharing)
→ SE_LAB git:(master) X

```

سپس تست‌های مربوطه را اجرا می‌کنیم:

Activities IntelliJ IDEA Ultimate Edition 21:46 14 آوريل en

File Edit View Navigate Code Refactor Build Run Tools Git Window Help

Project json-simple /-Univers pom.xml (json-simple) TestJson.java

Run: TestJson Tests passed: 2 of 2 tests - 27 ms

TestJson (org.json.simple) 27 ms /usr/lib/jvm/java-1.18.0-openjdk-amd64/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/opt/idea-IU-222.33x5.118/lib/idea_rt.jar=59945:/opt/idea-IU-222.33x5.118/bin

testJSONArrayCollection 17 ms =====decode=====

testDecode 10 ms =====the 2nd element of array=====

10

Process finished with exit code 0

ابتدا از کد اولیه coverage می‌گیریم و نتیجه را مشاهده می‌کنیم:

Element	Class, %	Method, %	Line, %
org.json.simple	71% (5/7)	32% (30/91)	25% (204/786)

```

Tests passed: 2 (moments ago)
13:14 CRLF UTF-8 Tab* EXPT3 %

```

Element	Class, %	Method, %	Line, %
org.json.simple	71% (5/7)	32% (30/91)	25% (204/786)
parser	75% (3/4)	40% (20/49)	33% (166/496)
JSONArray	100% (1/1)	20% (2/25)	11% (19/161)
JSONObject	100% (0/0)	100% (0/0)	100% (0/0)
JSONStreamAware	0% (0/1)	0% (0/9)	0% (0/6)
JSONValue	100% (0/0)	100% (0/0)	100% (0/0)

```

Tests passed: 2 (moments ago)
13:14 CRLF UTF-8 Tab* EXPT3 %

```

سپس، از نتیجه خروجی HTML نیز می‌گیریم:

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	33.3% (31/93)	25.9% (204/787)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
org.json.simple.parser	75% (34)	42% (21/50)	33.5% (166/496)
org.json.simple	66.7% (2/3)	23.3% (10/43)	13.1% (38/291)

generated on 2023-04-14 22:52

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	33.3% (31/93)	25.9% (204/787)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
org.json.simple	66.7% (2/3)	23.3% (10/43)	13.1% (38/291)
org.json.simple.parser	75% (34)	42% (21/50)	33.5% (166/496)

generated on 2023-04-14 22:52

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	33.3% (31/93)	25.9% (204/787)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
org.json.simple.parser	75% (34)	42% (21/50)	33.5% (166/496)
org.json.simple	66.7% (2/3)	23.3% (10/43)	13.1% (38/291)

generated on 2023-04-14 22:52

Screenshot captured
You can paste the image from the clipboard.

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	33.3% (31/93)	25.9% (204/787)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
org.json.simple	66.7% (2/3)	23.3% (10/43)	13.1% (38/291)
org.json.simple.parser	75% (34)	42% (21/50)	33.5% (166/496)

generated on 2023-04-14 22:52

Screenshot captured
You can paste the image from the clipboard.

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	33.3% (31/93)	25.9% (204/787)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
org.json.simple.parser	75% (34)	42% (21/50)	33.5% (166/496)
org.json.simple	66.7% (2/3)	23.3% (10/43)	13.1% (38/291)

generated on 2023-04-14 22:52

Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	71.4% (5/7)	33.3% (31/93)	25.9% (204/787)

Coverage Breakdown

Package	Class, %	Method, %	Line, %
org.json.simple	66.7% (2/3)	23.3% (10/43)	13.1% (38/291)
org.json.simple.parser	75% (34)	42% (21/50)	33.5% (166/496)

generated on 2023-04-14 22:52

در صورت کلیک بر روی هر یک از لینک‌های فایل گزارش، خطوط پوشش داده شده هر تست به رنگ سبز و خطوط پوشش داده نشده به رنگ قرمز نشان داده می‌شود:

کلاس JSONArray

Class	Class, %	Method, %	Line, %
JSONArray	100% (1/1)	20% (5/25)	11.8% (19/161)

```

1 /**
2 * $Id: JSONArray.java,v 1.1 2006/04/15 14:19:48 platform Exp
3 * Created on 2006-4-10
4 */
5 package org.json.simple;
6
7 import java.io.IOException;
8 import java.io.StringWriter;
9 import java.io.Writer;
10 import java.util.ArrayList;
11 import java.util.Collection;
12 import java.util.Iterator;
13
14 /**
15 * A JSONArray array. JSONObject supports java.util.List interface.
16 *
17 * @author FangYidong<fangyidong@yahoo.com.cn>
18 */
19 public class JSONArray extends ArrayList implements JSONAware, JSONStreamAware {
20     private static final long serialVersionUID = 3957988303675231981L;
21
22     /**
23      * Constructs an empty JSONArray.
24      */
25     public JSONArray(){
26         super();
27     }
28
29     /**
30      * Constructs a JSONArray containing the elements of the specified
31      * collection, in the order they are returned by the collection's iterator.
32      *
33      * @param c the collection whose elements are to be placed into this JSONArray
34      */
35     public JSONArray(Collection c){
36         super(c);
37     }
38
39     /**
40      * Encode a list into JSON text and write it to out.
41      * If this list is also a JSONStreamAware or a JSONAware, JSONStreamAware and JSONAware specific behaviours will be ignored at this top level.
42      *
43      * @see org.json.simple.JSONObject#writeJSONString(Writer)
44      *
45      * @param collection
46      * @param out
47      */
48     public static void writeJSONString(Collection collection, Writer out) throws IOException{
49         if(collection == null){
50             out.write("null");
51             return;
52         }
53
54         boolean first = true;
55         Iterator iter=collection.iterator();
56
57         out.write("[");
58         while(iter.hasNext()){
59             if(first == true)
60                 first = false;
61
62             out.write(",");
63             out.write(JSONObject.toJSONString(iter.next()));
64
65         }
66         out.write("]");
67     }
68
69 }

```

Class	Class, %	Method, %	Line, %
JSONArray	100% (1/1)	20% (5/25)	11.8% (19/161)

```

87     public static String toJSONString(Collection collection){
88         final StringWriter writer = new StringWriter();
89
90         try {
91             writeJSONString(collection, writer);
92             return writer.toString();
93         } catch(IOException e){
94             // This should never happen for a StringWriter
95             throw new RuntimeException(e);
96         }
97     }
98
99     public static void writeJSONString(byte[] array, Writer out) throws IOException{
100
101         if(array == null)
102             out.write("null");
103         else if(array.length == 0)
104             out.write("[]");
105         else {
106             out.write("[");
107             out.write(String.valueOf(array[0]));
108
109             for(int i = 1; i < array.length; i++){
110                 out.write(",");
111                 out.write(String.valueOf(array[i]));
112             }
113             out.write("]");
114         }
115     }
116
117     public static String toJSONString(byte[] array){
118         final StringWriter writer = new StringWriter();
119
120         try {
121             writeJSONString(array, writer);
122             return writer.toString();
123         } catch(IOException e){
124             // This should never happen for a StringWriter
125             throw new RuntimeException(e);
126         }
127     }
128
129     public static void writeJSONString(short[] array, Writer out) throws IOException{
130         if(array == null)
131             out.write("null");
132         else if(array.length == 0)
133             out.write("[]");
134         else {
135             out.write("[");
136             out.write(String.valueOf(array[0]));
137
138             for(int i = 1; i < array.length; i++){
139                 out.write(",");
140                 out.write(String.valueOf(array[i]));
141             }
142             out.write("]");
143         }
144     }
145
146     public static String toJSONString(short[] array){
147         final StringWriter writer = new StringWriter();
148
149         try {
150             writeJSONString(array, writer);
151             return writer.toString();
152         } catch(IOException e){
153             // This should never happen for a StringWriter
154             throw new RuntimeException(e);
155         }
156     }

```

```
Activities Google Chrome
Software_Eng_Lab/Sessi... Coverage Report>JSON... + 21:56 14 Ju... en
File /home/mmbenhas/University/software/LAB/SE_LAB/SE_LAB/first_coverage/ns/1/ Screenshot captured
You can paste the image from the clipboard.

155     // This should never happen for a StringWriter
156     throw new RuntimeException();
157 }
158 }
159 
160 public static void writeJSONObject(int[] array, Writer out) throws IOException{
161     if(array == null){
162         out.write("null");
163     } else if(array.length == 0) {
164         out.write("[ ]");
165     } else {
166         out.write("[");
167         out.write(String.valueOf(array[0]));
168         for(int i = 1; i < array.length; i++){
169             out.write(",");
170             out.write(String.valueOf(array[i]));
171         }
172     }
173     out.write("]");
174 }
175 }
176 
177 public static String toJSONString(int[] array){
178     final StringWriter writer = new StringWriter();
179 
180     try {
181         writeJSONObject(array, writer);
182         return writer.toString();
183     } catch(IOException e){
184         // This should never happen for a StringWriter
185         throw new RuntimeException(e);
186     }
187 }
188 
189 public static void writeJSONObject(long[] array, Writer out) throws IOException{
190     if(array == null){
191         out.write("null");
192     } else if(array.length == 0) {
193         out.write("[ ]");
194     } else {
195         out.write("[");
196         out.write(String.valueOf(array[0]));
197         for(int i = 1; i < array.length; i++){
198             out.write(",");
199             out.write(String.valueOf(array[i]));
200         }
201     }
202     out.write("]");
203 }
204 
205 }
206 
207 public static String toJSONString(long[] array){
208     final StringWriter writer = new StringWriter();
209 
210     try {
211         writeJSONObject(array, writer);
212         return writer.toString();
213     } catch(IOException e){
214         // This should never happen for a StringWriter
215         throw new RuntimeException(e);
216     }
217 }
218 
219 public static void writeJSONObject(float[] array, Writer out) throws IOException{
220     if(array == null){
221         out.write("null");
222     } else if(array.length == 0) {
223         out.write("[ ]");
224     }
```

```
Activities Google Chrome
Software_Eng_Lab/Sessi... Coverage Report>JSON... + 21:56 14 Ju... en
File /home/mmbenhas/University/software/LAB/SE_LAB/SE_LAB/first_coverage/ns/1/ Screenshot captured
You can paste the image from the clipboard.

262     }
263     out.write("]");
264 }
265 }
266 
267 public static String toJSONString(double[] array){
268     final StringWriter writer = new StringWriter();
269 
270     try {
271         writeJSONObject(array, writer);
272         return writer.toString();
273     } catch(IOException e){
274         // This should never happen for a StringWriter
275         throw new RuntimeException(e);
276     }
277 }
278 
279 public static void writeJSONObject(boolean[] array, Writer out) throws IOException{
280     if(array == null){
281         out.write("null");
282     } else if(array.length == 0) {
283         out.write("[ ]");
284     } else {
285         out.write("[");
286         out.write(String.valueOf(array[0]));
287         for(int i = 1; i < array.length; i++){
288             out.write(",");
289             out.write(String.valueOf(array[i]));
290         }
291     }
292     out.write("]");
293 }
294 
295 }
296 
297 public static String toJSONString(boolean[] array){
298     final StringWriter writer = new StringWriter();
299 
300     try {
301         writeJSONObject(array, writer);
302         return writer.toString();
303     } catch(IOException e){
304         // This should never happen for a StringWriter
305         throw new RuntimeException(e);
306     }
307 }
308 
309 public static void writeJSONObject(char[] array, Writer out) throws IOException{
310     if(array == null){
311         out.write("null");
312     } else if(array.length == 0) {
313         out.write("[ ]");
314     } else {
315         out.write("[\"");
316         out.write(String.valueOf(array[0]));
317         out.write("\"]");
318         for(int i = 1; i < array.length; i++){
319             out.write(",");
320             out.write(String.valueOf(array[i]));
321         }
322     }
323     out.write("]");
324 }
325 
326 }
327 
328 public static String toJSONString(char[] array){
329     final StringWriter writer = new StringWriter();
330 
331     try {
```

Activities Google Chrome

Software_Eng_Lab/Sessi... Coverage_Report>JSON

File | /home/mmbehnas/University/software/LAB/SE LAB/SE LAB/first_coverage/ns-1

21:56 14 جولی en

Screenshot captured You can paste the image from the clipboard.

```

316         out.write("\\\"");
317         out.write(String.valueOf(array[0]));
318
319         for(int i = 1; i < array.length; i++){
320             out.write(",");
321             out.write(String.valueOf(array[i]));
322         }
323
324         out.write("\\\"");
325     }
326 }
327 public static String toJSONObject(char[] array){
328     final StringWriter writer = new StringWriter();
329
330     try {
331         writeJSONObject(array, writer);
332         return writer.toString();
333     } catch(IOException e){
334         // This should never happen for a StringWriter
335         throw new RuntimeException(e);
336     }
337 }
338
339 public static void writeJSONObject(Object[] array) throws IOException{
340     if(array == null)
341         out.write("null");
342     else if(array.length == 0) {
343         out.write("[]");
344     } else {
345         out.write("[");
346         JSONValue.writeJSONObject(array[0], out);
347
348         for(int i = 1; i < array.length; i++){
349             out.write(",");
350             JSONValue.writeJSONObject(array[i], out);
351         }
352     }
353
354     out.write("]");
355 }
356
357 public static String toJSONObject(Object[] array){
358     final StringWriter writer = new StringWriter();
359
360     try {
361         writeJSONObject(array, writer);
362         return writer.toString();
363     } catch(IOException e){
364         // This should never happen for a StringWriter
365         throw new RuntimeException(e);
366     }
367 }
368
369 public String toJSONString(){
370     return toJSONString(this);
371 }
372 }
373
374 /**
375 * Returns a string representation of this array. This is equivalent to
376 * calling {@link JSONArray#toJSONString()}.
377 */
378 public String toString() {
379     return toJSONString();
380 }
381 }
```

generated on 2023-04-14 22:52

کلاس JSONValue

Activities Google Chrome

Software_Eng_Lab/Sessi... Coverage_Report>JSON

File | /home/mmbehnas/University/software/LAB/SE LAB/SE LAB/first_coverage/ns-1

21:57 14 جولی en

Screenshot captured You can paste the image from the clipboard.

Current scope: all classes | org.json.simple

Coverage Summary for Class: JSONValue (org.json.simple)

Class	Class, %	Method, %	Line, %
JSONValue	100% (1/1)	55.6% (5/9)	20.2% (19/94)

```

1 /*
2 * $Id: JSONValue.java,v 1.1 2006/04/15 14:37:04 platform Exp $
3 * Created on 2006-04-15
4 */
5 package org.json.simple;
6
7 import java.io.IOException;
8 import java.io.Reader;
9 import java.io.StringReader;
10 import java.io.StringWriter;
11 import java.io.Writer;
12 import java.util.Collection;
13 import java.util.List;
14 import java.util.Map;
15
16 import org.json.simple.parser.JSONParser;
17 import org.json.simple.parser.ParseException;
18
19 /**
20 * Author FangYidong<Fangyidong@yahoo.com.cn>
21 */
22
23 public class JSONValue {
24
25     /**
26      * Parse JSON text into Java object from the input source.
27      * Please use parseWithException() if you don't want to ignore the exception.
28      *
29      * @see org.json.simple.parser.JSONParser#parse(Reader)
30      * @see #parseWithException(Reader)
31      *
32      * @param in
33      * @return Instance of the following:
34      *         org.json.simple.JSONObject,
35      *         org.json.simple.JSONArray,
36      *         java.lang.String,
37      *         java.lang.Number,
38      *         java.lang.Boolean,
39      *         null
40      *
41      * @deprecated This method may throw an ErrorCode Error instead of returning
42      * (code null); please use {@link JSONValue#parseWithException(Reader)}
43      */
44     public static Object parse(Reader in){
45         try{
46             JSONParser parser=new JSONParser();
47             return parser.parse(in);
48         }
49         catch(Exception e){
50             return null;
51         }
52     }
53
54
55     /**
56      * Parse JSON text into Java object from the given string.
57      * Please use parseWithException() if you don't want to ignore the exception.
58      *
59      * @see org.json.simple.parser.JSONParser#parse(Reader)
60      * @see #parseWithException(Reader)
61     }
```

Activities Google Chrome

Software_Eng_Lab/Sess1 Coverage Report - JSONParser

File /home/mmbehnas/University/software/LAB/SE_LAB/first_coverage/ns-1/so

Screenshot captured
You can paste the image from the clipboard.

```

88     + java.lang.String;
89     + java.lang.Number;
90     + java.lang.Boolean;
91     + null;
92
93     @throws IOException
94     @throws ParseException
95
96     public static Object parseWithException(Reader in) throws IOException, ParseException{
97         JSONParser parser=new JSONParser();
98         return parser.parse(in);
99     }
100
101    public static Object parseWithException(String s) throws ParseException{
102        JSONParser parser=new JSONParser();
103        return parser.parse(s);
104    }
105
106    /**
107     * Encode an object into JSON text and write it to out.
108     * If this object is a Map or a List, and it's also a JSONStreamAware or a JSONAware, JSONStreamAware or JSONAware will be considered firstly.
109     * <P>
110     * NOT call this method from writeJSONObject(Writer) of a class that implements both JSONStreamAware and (Map or List) with
111     * "this" as the first parameter, use JSONObject.writeJSONObject(Map, Writer) or JSONArray.writeJSONArray(List, Writer) instead.
112     *
113     * @see org.json.simple.JSONObject#writeJSONObject(Map, Writer)
114     * @see org.json.simple.JSONArray#writeJSONArray(List, Writer)
115     */
116    @JSONAware
117    @OutputStreamWriter
118    public static void writeJSONObject(Object value, Writer out) throws IOException {
119        if(value == null){
120            out.write("null");
121            return;
122        }
123
124        if(value instanceof String){
125            out.write(value);
126            out.write("\\\"");
127            out.write(escape((String)value));
128            out.write("\\\"");
129            return;
130        }
131
132        if(value instanceof Double){
133            if(((Double)value).isInfinite() || ((Double)value).isNaN()){
134                out.write("null");
135            } else
136                out.write(value.toString());
137            return;
138        }
139
140        if(value instanceof Float){
141            if(((Float)value).isInfinite() || ((Float)value).isNaN())
142                out.write("null");
143            else
144                out.write(value.toString());
145            return;
146        }
147
148        if(value instanceof Number){
149            out.write(value.toString());
150            return;
151        }
152
153        if(value instanceof Boolean){
154            out.write(value.toString());
155            return;
156        }
157    }

```

:JSONParser کلاس

Activities Google Chrome

Software_Eng_Lab/Sess1 Coverage Report - JSONParser

File /home/mmbehnas/University/software/LAB/SE_LAB/first_coverage/ns-2/sources/source-1.html

Coverage Summary for Class: JSONParser (org.json.simple.parser)

Class	Class, %	Method, %	Line, %
JSONParser	100% (1/1)	43.8% (7/16)	14.7% (37/251)

```

1  /*
2   * Std: JSONParser.java.v 1.1 2006/04/15 14:10:48 platform Exp $
3   * Created on 2006-4-15
4   */
5 package org.json.simple.parser;
6
7 import java.io.IOException;
8 import java.io.Reader;
9 import java.io.StringReader;
10 import java.util.LinkedList;
11 import java.util.List;
12 import java.util.Map;
13
14 import org.json.simple.JSONArray;
15 import org.json.simple.JSONObject;
16
17 /**
18  * Parser for JSON text. Please note that JSONParser is NOT thread-safe.
19  */
20
21 /* @author FangYidong<fangyidong@yahoo.com.cn>
22 */
23 public class JSONParser {
24     public static final int S_INIT=0;
25     public static final int S_IN_FINISHED_VALUE=1;//string,number,boolean,null,object,array
26     public static final int S_IN_OBJECT=2;
27     public static final int S_IN_PAIR=3;
28     public static final int S_PASSED_PAIR_KEY=4;
29     public static final int S_IN_PAIR_VALUE=5;
30     public static final int S_IN_ERROR=-1;
31
32     private LinkedList handlerStatusStack;
33     private Ytoken lexer = new Ytoken((Reader)null);
34     private Ytoken token = null;
35     private int status = S_INIT;
36
37     private int peekStatus(LinkedList statusStack){
38         if(statusStack.size()==0)
39             Integer status=(Integer)statusStack.getFirst();
40         return status.intValue();
41     }
42
43     /**
44      * Reset the parser to the initial state without resetting the underlying reader.
45      */
46     public void reset(){
47         token = null;
48         status = S_INIT;
49         handlerStatusStack = null;
50     }
51
52     /**
53      * Reset the parser to the initial state with a new character reader.
54      */
55     public void in ( Reader in ) throws IOException {
56         if(in==null)
57             throw new IOException("null in");
58         if(in instanceof JSONParser)
59             throw new IOException("already a parser");
60     }

```

Activities Google Chrome 21:57 14 Juillet 2023

Screenshot captured You can paste the image from the clipboard.

```
107
108     * @throws IOException
109     * @throws ParseException
110
111    public Object parse(Reader in, ContainerFactory containerFactory) throws IOException, ParseException{
112        reset();
113        LinkedList statusStack = new LinkedList();
114        LinkedList valueStack = new LinkedList();
115
116        try{
117            do{
118                nextToken();
119                switch(token.type){
120                    case S_INIT:
121                        switch(token.type){
122                            case Yttoken.TYPE_VALUE:
123                                status=S_IN_FINISHED_VALUE;
124                                statusStack.addFirst(new Integer(status));
125                                valueStack.addFirst(token.value);
126                                break;
127
128                            case Yttoken.TYPE_LEFT_BRACE:
129                                status=S_IN_OBJECT;
130                                statusStack.addFirst(new Integer(status));
131                                valueStack.addFirst(createObjectContainer(containerFactory));
132                                break;
133
134                            case Yttoken.TYPE_LEFT_SQUARE:
135                                status=S_IN_ARRAY;
136                                statusStack.addFirst(new Integer(status));
137                                valueStack.addFirst(createArrayContainer(containerFactory));
138                                break;
139
140                            default:
141                                status=S_IN_ERROR;
142
143                            } //inner switch
144                            break;
145
146                        case S_IN_FINISHED_VALUE:
147                            if(token.type==Yttoken.TYPE_EOF)
148                                return valueStack.removeFirst();
149                            else
150                                throw new ParseException(getPosition(), ParseException.ERROR_UNEXPECTED_TOKEN, token);
151
152                        case S_IN_OBJECT:
153                            switch(token.type){
154                                case Yttoken.TYPE_COMMA:
155                                    break;
156
157                                case Yttoken.TYPE_VALUE:
158                                    if(token.value instanceof String){
159                                        String key=(String)token.value;
160                                        if(statusStack.size()>1)
161                                            statusStack.removeFirst();
162                                        statusStack.addFirst(new Integer(status));
163                                        status=S_PASSED_PAIR_KEY;
164                                        statusStack.addFirst(new Integer(status));
165                                    }
166                                    else{
167                                        status=S_IN_ERROR;
168
169                                    }
170                                    break;
171
172                                default:
173                                    status=S_IN_ERROR;
174
175                            } //inner switch
176
177                } //inner loop
178            } //outer loop
179
180        } //try
181
182        finally{
183            close();
184        }
185    }
186
187    /**
188     * Escape quotes, \, /, \r, \n, \b, \f, \t and other control characters (U+0000 through U+001F).
189     * @param s
190     * @return sb
191     */
192    public static String escape(String s){
193        if(s==null)
194            return null;
195        StringBuffer sb = new StringBuffer();
196        escape(s, sb);
197        return sb.toString();
198    }
199
200    /**
201     * Argument s - Must not be null.
202     * Argument sb
203     */
204    static void escape(String s, StringBuffer sb) {
205        final int len = s.length();
206        for(int i=0;i<len;i++){
207            char ch=s.charAt(i);
208            switch(ch){
209                case '\n':
210                    sb.append("\\\\n");
211                    break;
212                case '\\':
213                    sb.append("\\\\");
214                    break;
215                case '\r':
216                    sb.append("\\\\r");
217                    break;
218                case '\t':
219                    sb.append("\\\\t");
220                    break;
221                case '\b':
222                    sb.append("\\\\b");
223                    break;
224                case '\f':
225                    sb.append("\\\\f");
226                    break;
227                case '\u0000'..'\u001F':
228                    if((ch>='\u0000' && ch<='\u0007' || (ch>='\u000F' && ch<='\u000F') || (ch>='\u0080' && ch<='\u00FF')){
229                        String ss=Integer.toHexString(ch);
230                        sb.append("\\\\u");
231                        for(int j=ss.length();j>0;j--){
232                            sb.append(ss.substring(j,j));
233                        }
234                    }
235                    else{
236                        sb.append(ss.toUpperCase());
237                    }
238            }
239        }
240    }
241
242 } //for
243
244
245 }
```

Activities Google Chrome 21:57 14 Juillet 2023

Screenshot captured You can paste the image from the clipboard.

```
251
252
253    /**
254     * Escape quotes, \, /, \r, \n, \b, \f, \t and other control characters (U+0000 through U+001F).
255     * @param s
256     * @return sb
257     */
258    public static String escape(String s){
259        if(s==null)
260            return null;
261        StringBuffer sb = new StringBuffer();
262        escape(s, sb);
263        return sb.toString();
264    }
265
266    /**
267     * Argument s - Must not be null.
268     * Argument sb
269     */
270    static void escape(String s, StringBuffer sb) {
271        final int len = s.length();
272        for(int i=0;i<len;i++){
273            char ch=s.charAt(i);
274            switch(ch){
275                case '\n':
276                    sb.append("\\\\n");
277                    break;
278                case '\\':
279                    sb.append("\\\\");
280                    break;
281                case '\r':
282                    sb.append("\\\\r");
283                    break;
284                case '\t':
285                    sb.append("\\\\t");
286                    break;
287                case '\b':
288                    sb.append("\\\\b");
289                    break;
290                case '\f':
291                    sb.append("\\\\f");
292                    break;
293                case '\u0000'..'\u001F':
294                    if((ch>='\u0000' && ch<='\u0007' || (ch>='\u000F' && ch<='\u000F') || (ch>='\u0080' && ch<='\u00FF')){
295                        String ss=Integer.toHexString(ch);
296                        sb.append("\\\\u");
297                        for(int j=ss.length();j>0;j--){
298                            sb.append(ss.substring(j,j));
299                        }
300                    }
301                    else{
302                        sb.append(ss.toUpperCase());
303                    }
304            }
305        }
306    }
307
308
309 } //for
310
311
312
313
314
315 }
```

generated on 2023-04-14 22:52

Activities Google Chrome 21:57 14 جولی en □ ×

Software_Eng_Lab/Sessi... Coverage Report > JSON! +

File | /home/mmbehnaas/University/software/LAB/SE_LAB/first_coverage/ns-2/sources/source-1.html

```
206     status=S_IN_OBJECT;
207     statusStack.addFirst(new Integer(status));
208     valueStack.addFirst(newObject);
209     break;
210   default:
211     status=S_IN_ERROR;
212     break;
213 }
214 case S_IN_ARRAY:
215   switch(token.type){
216     case Ytoken.TYPE_COMMA:
217       break;
218     case Ytoken.TYPE_VALUE:
219       List val=(List)valueStack.getFirst();
220       val.add(token.value);
221       break;
222     case Ytoken.TYPE_RIGHT_SQUARE:
223       if(valueStack.size()>1){
224         statusStack.removeFirst();
225         valueStack.removeFirst();
226         statusStack.setStatus(statusStack);
227       }
228       else{
229         status=S_IN_FINISHED_VALUE;
230       }
231       break;
232     case Ytoken.TYPE_LEFT_BRACE:
233       val=(List)valueStack.getFirst();
234       Map newObject=createObjectContainer(containerFactory);
235       val.put(newObject);
236       status=S_IN_OBJECT;
237       statusStack.addFirst(new Integer(status));
238       valueStack.addFirst(newObject);
239       break;
240     case Ytoken.TYPE_LEFT_SQUARE:
241       val=(List)valueStack.getFirst();
242       List newArray=parseCreateArrayContainer(containerFactory);
243       val.add(newArray);
244       status=S_IN_ARRAY;
245       statusStack.addFirst(new Integer(status));
246       valueStack.addFirst(newArray);
247       break;
248     default:
249       status=S_IN_ERROR;
250   }
251 } //inner switch
252 case S_IN_ERROR:
253   throw new ParseException(getPosition(), ParseException.ERROR_UNEXPECTED_TOKEN, token);
254 } //switch
255 if(status==S_IN_ERROR){
256   throw new ParseException(getPosition(), ParseException.ERROR_UNEXPECTED_TOKEN, token);
257 }
258 } //while(token.type!=Ytoken.TYPE_EOF);
259 }
260 catch(IOException ie){
261   throw ie;
262 }
263 }
264 }
265 throw new ParseException(getPosition(), ParseException.ERROR_UNEXPECTED_TOKEN, token);
266 }
267 private void nextToken() throws ParseException, IOException{
268   token = lexer.yylex();
269   if(token == null)
270     token = new Ytoken(Ytoken.TYPE_EOF, null);
271 }
272 }
273 private Map createObjectContainer(ContainerFactory containerFactory){
274   if(containerFactory == null)
```

حال، با افزودن تست‌های جدید coverage هر کلاس را افزایش می‌دهیم:

در ادامه با اضافه کردن تست‌های جدید، coverage هر کلاس را رو به بالا بهبود می‌دهیم.

کلاس JSONArray

با اضافه کردن تست زیر در کلاس TestJSON خواهیم داشت:

```
json-simple - TestJson.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Project json-simple - /University/software/LAB/SE_LAB/SE_LAB/SE_LAB/src
  > .idea
  > .settings
  > doc
  > main
    > java
      > org.json.simple
        > parser
          > JSONArray
          > JSONAware
          > JSONObject
          > JSONStreamAware
          > JSONValue
    > test
      > java
        > org.json.simple
          > JSONException
          > TestJson
  > target
  > .classpath
  > .project
  > AUTHORS.txt
  > build.xml
  > ChangeLog.txt
  > LICENSE.txt
Run: < Run > Tests passed: 7 of 7 tests - 21 ms
  > TestJson (org.json.simple) 21 ms /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/opt/idea-IU-222.3345.118/lib/idea_rt.jar=40259:/opt/idea-IU-222.3345.118/bin
    > testJSONArrayWithMixedTypes 8 ms =====decode=====
    > testJSONObjectWithNestedObjects 1 ms =====the 2nd element of array=====
    > testJSONObjectWithNullValues 1 ms 10
    > testJSONArrayCollection 0 ms
    > testJSONArrayBooleanToString 0 ms
    > testJSONArrayToStringWithNullValues 1 ms
    > testDecode 10 ms
  Process finished with exit code 0
Tests passed: 7 (moments ago)
```

```
json-simple - TestJson.java
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Project json-simple - /University/software/LAB/SE_LAB/SE_LAB/SE_LAB/src
  > .idea
  > .settings
  > doc
  > main
    > java
      > org.json.simple
        > parser
          > JSONArray
          > JSONAware
          > JSONObject
          > JSONStreamAware
          > JSONValue
    > test
      > java
        > org.json.simple
          > JSONException
          > TestJson
  > target
  > .classpath
  > .project
  > AUTHORS.txt
  > build.xml
  > ChangeLog.txt
  > LICENSE.txt
Run: < Run > Tests passed: 7 of 7 tests - 21 ms
  > TestJson (org.json.simple) 21 ms /usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/opt/idea-IU-222.3345.118/lib/idea_rt.jar=40259:/opt/idea-IU-222.3345.118/bin
    > testJSONArrayWithMixedTypes 8 ms =====decode=====
    > testJSONObjectWithNestedObjects 1 ms =====the 2nd element of array=====
    > testJSONObjectWithNullValues 1 ms 10
    > testJSONArrayCollection 0 ms
    > testJSONArrayBooleanToString 0 ms
    > testJSONArrayToStringWithNullValues 1 ms
    > testDecode 10 ms
  Process finished with exit code 0
Tests passed: 7 (moments ago)
```

با افزودن این تست متده `toJSONString` که ورودی آن از نوع آرایه‌ای به صورت first item و second item می‌باشد فراخوانی می‌شود و این تست بررسی می‌کند که آیا این آرایه به درستی به فرمت JSON از نوع String تبدیل می‌شود یا خیر.

کلاس‌های Yylex, JSONParser, JSONObject و JSONArray با اضافه کردن زیر خواهیم داشت:

The image shows two side-by-side Java IDE windows, likely IntelliJ IDEA, displaying code related to JSON processing and testing.

Top Window (json-simple - TestJson.java):

```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Project json-simple pom.xml (json-simple) TestJson.java JSONValue.java JSONStreamAware.java JSONObject.java JSONArray.java
src
  main
    java
      org.json.simple
        parser
          JSONArray 28% methods, 18% lines covered
          JSONException 28% methods, 18% lines covered
          JSONObject 66% methods, 63% lines covered
          JSONStreamAware 62% methods, 44% lines covered
    test
      java
        org.json.simple
          JSONException
  target
  .classpath
  .project
  AUTHORS.txt
  build.xml
  ChangeLog.txt
  LICENSE.txt
  pom.xml
  README.txt
  test.xml
  VERSION.txt
External Libraries
Scratches and Consoles

```

```

public void testJSONArrayCollection() {
    final ArrayList<String> testList = new ArrayList<>();
    testList.add("First item");
    testList.add("Second item");
    final JSONArray jsonArray = new JSONArray(testList);

    assertEquals(expected: "[\"First item\", \"Second item\"]", jsonArray.toJSONString());
}

new +
@Test
public void testJSONArrayBooleanToString() {
    final boolean[] testArray = new boolean[]{true, false, true};
    final JSONArray jsonArray = new JSONArray(Collections.singletonList(testArray));
    assertEquals(expected: "[true, false, true]", jsonArray.toJSONString());
}

new +
@Test
public void testJSONObjectWithNestedObjects() {
    final JSONObject innerObject1 = new JSONObject();
    innerObject1.put("key1", "value1");
    innerObject1.put("key2", "value2");

    final JSONObject innerObject2 = new JSONObject();
    innerObject2.put("key3", "value3");
    innerObject2.put("key4", "value4");

    final JSONObject outerObject = new JSONObject();
    outerObject.put("innerObject1", innerObject1);
    outerObject.put("innerObject2", innerObject2);

    assertEquals(expected: "{\"innerObject1\":{\"key1\":\"value1\", \"key2\":\"value2\"}, \"innerObject2\":{\"key3\":\"value3\", \"key4\":\"value4\"}}", outerObject.toJSONString());
}

new +
@Test
public void testJSONArrayToStringWithNullValues() {
    final JSONArray jsonArray = new JSONArray();
    jsonArray.add(null);
    jsonArray.add("value");
    jsonArray.add(null);
    assertEquals(expected: "[null, \"value\", null]", jsonArray.toJSONString());
}

new +
@Test
public void testJSONObjectWithNullValues() {
    final JSONObject jsonObject = new JSONObject();
    jsonObject.put("key1", null);
    jsonObject.put("key2", "value2");
    jsonObject.put("key3", null);
    assertEquals(expected: "{\"key1\":null, \"key2\":\"value2\", \"key3\":null}", jsonObject.toJSONString());
}

new +
@Test
public void testJSONArrayWithMixedTypes() {
    final JSONArray jsonArray = new JSONArray();
    jsonArray.add("string");
    jsonArray.add(10);
    jsonArray.add(true);
    jsonArray.add(2.5);
    assertEquals(expected: "[\"string\", 10, true, 2.5]", jsonArray.toJSONString());
}

```

Bottom Window (json-simple - TestJson.java):

```

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Project json-simple pom.xml (json-simple) TestJson.java JSONValue.java JSONStreamAware.java JSONObject.java JSONArray.java
src
  main
    java
      org.json.simple
        parser
          JSONArray 28% methods, 18% lines covered
          JSONException 28% methods, 18% lines covered
          JSONObject 66% methods, 63% lines covered
          JSONStreamAware 62% methods, 44% lines covered
    test
      java
        org.json.simple
          JSONException
  target
  .classpath
  .project
  AUTHORS.txt
  build.xml
  ChangeLog.txt
  LICENSE.txt
  pom.xml
  README.txt
  test.xml
  VERSION.txt
External Libraries
Scratches and Consoles

```

```

public void testJSONArrayCollection() {
    final ArrayList<String> testList = new ArrayList<>();
    testList.add("First item");
    testList.add("Second item");
    final JSONArray jsonArray = new JSONArray(testList);

    assertEquals(expected: "[\"First item\", \"Second item\"]", jsonArray.toJSONString());
}

new +
@Test
public void testJSONArrayBooleanToString() {
    final boolean[] testArray = new boolean[]{true, false, true};
    final JSONArray jsonArray = new JSONArray(Collections.singletonList(testArray));
    assertEquals(expected: "[true, false, true]", jsonArray.toJSONString());
}

new +
@Test
public void testJSONObjectWithNestedObjects() {
    final JSONObject innerObject1 = new JSONObject();
    innerObject1.put("key1", "value1");
    innerObject1.put("key2", "value2");

    final JSONObject innerObject2 = new JSONObject();
    innerObject2.put("key3", "value3");
    innerObject2.put("key4", "value4");

    final JSONObject outerObject = new JSONObject();
    outerObject.put("innerObject1", innerObject1);
    outerObject.put("innerObject2", innerObject2);

    assertEquals(expected: "{\"innerObject1\":{\"key1\":\"value1\", \"key2\":\"value2\"}, \"innerObject2\":{\"key3\":\"value3\", \"key4\":\"value4\"}}", outerObject.toJSONString());
}

new +
@Test
public void testJSONArrayToStringWithNullValues() {
    final JSONArray jsonArray = new JSONArray();
    jsonArray.add(null);
    jsonArray.add("value");
    jsonArray.add(null);
    assertEquals(expected: "[null, \"value\", null]", jsonArray.toJSONString());
}

new +
@Test
public void testJSONObjectWithNullValues() {
    final JSONObject jsonObject = new JSONObject();
    jsonObject.put("key1", null);
    jsonObject.put("key2", "value2");
    jsonObject.put("key3", null);
    assertEquals(expected: "{\"key1\":null, \"key2\":\"value2\", \"key3\":null}", jsonObject.toJSONString());
}

new +
@Test
public void testJSONArrayWithMixedTypes() {
    final JSONArray jsonArray = new JSONArray();
    jsonArray.add("string");
    jsonArray.add(10);
    jsonArray.add(true);
    jsonArray.add(2.5);
    assertEquals(expected: "[\"string\", 10, true, 2.5]", jsonArray.toJSONString());
}

```

The code in both windows is identical, demonstrating various JSON manipulation and testing scenarios using the org.json.simple library.

در این با فراخوانی متدها parse و JSONValue از کلاس JSONParser متد parse از کلاس JSONValue فراخوانی می‌شود و چون رشته‌ی ما شامل کاراکترهای { و } می‌باشد، در نتیجه توکن‌های جدیدی در متدهای pushback و yypushback ساخته می‌شود و کیس‌های جدیدی در متدهای parse در JSONParse اجرا می‌شوند و خطهایی اجرا می‌شود که در تست‌های قبلی اجرا نشده بود.

همچنین با توجه به اینکه رشته‌ای که در تست تعريف شده (S) یک آبجکت JSON می‌باشد، هنگام پars کردن یک آبجکت از نوع JSONParse ساخته می‌شود.

با اجرای این تست با توجه به اینکه مقادیر عددی یک آبجکت باید به رشته تبدیل شوند، خطوط جدیدی از متدهای writeJSONString و writeJSONArray در کلاس JSONValue پیاده‌سازی می‌شوند که تصاویر

همگی آنها و اثر coverage در کد کلاس‌ها و گزارش فایل آنها در ادامه آمده‌اند:

برای اولین تصاویر ns-1 را مشاهده می‌کنیم که در واقع coverage معلوم است:

```
39 /**
40  * Encode a list into JSON text and write it to out.
41  * If this list is also a JSONStreamAware or a JSONAware, JSONStreamAware and JSONAware specific behaviours will be ignored at this top level.
42  *
43  * @see org.json.simple.JSONValue#writeJSONString(Object, Writer)
44  */
45 /**
46  * @param collection
47  * @param out
48  */
49 public static void writeJSONString(Collection collection, Writer out) throws IOException{
50     if(collection == null){
51         out.write("null");
52         return;
53     }
54     boolean first = true;
55     Iterator iter=collection.iterator();
56
57     out.write("[");
58     while(iter.hasNext()){
59         if(first)
60             first = false;
61         else
62             out.write(",");
63
64         Object value=iter.next();
65         if(value == null){
66             out.write("null");
67             continue;
68         }
69         JSONValue.writeJSONString(value, out);
70     }
71     out.write("]");
72 }
73
74 public void writeJSONString(Writer out) throws IOException{
75     writeJSONString(this, out);
76 }
77
78 /**
79  * Convert a list to JSON text. The result is a JSON array.
80  * If this list is also a JSONAware, JSONAware specific behaviours will be omitted at this top level.
81  *
82  * @see org.json.simple.JSONValue#toJSONString(Object)
83  */
84 /**
85  * @param collection
86  * @return JSON text, or "null" if list is null.
87  */
88 public static String toJSONString(Collection collection){
89     final StringWriter writer = new StringWriter();
90
91     try {
92         writeJSONString(collection, writer);
93         return writer.toString();
94     } catch(IOException e) {
95         // This should never happen for a StringWriter
96         throw new RuntimeException(e);
97     }
98 }
99
100 public static void writeJSONString(byte[] array, Writer out) throws IOException{
101    if(array == null){
102        out.write("null");
103    } else if(array.length == 0) {
104        out.write("[]");
105    } else {
106        out.write("[");
107        out.write(String.valueOf(array[0]));
108
109        for(int i = 1; i < array.length; i++){
110            out.write(",");
111        }
112    }
113 }
```

```
translate.go0 x Google Keep x Default Mapper x vim.rtorr.com x @ cw.sharif.edu x 14012-404041 x Software_Eng x Software_Eng x Coverage Repo x + - _
```

File /home/mmbehnasr/University/software/LAB_SE_LAB/SE_LAB/final_coverage/ns-1/sources/source-3.html

```
110  
111 * @param value  
112 * @param writer  
113  
114 public static void writeJSONObject(Object value, Writer out) throws IOException {  
115     if(value == null){  
116         out.write("null");  
117         return;  
118     }  
119  
120     if(value instanceof String){  
121         out.write((String)value);  
122     }  
123     else if(value instanceof Double){  
124         if(((Double)value).isInfinite() || ((Double)value).isNaN())  
125             out.write("null");  
126         else  
127             out.write(value.toString());  
128     }  
129     else if(value instanceof Float){  
130         if(((Float)value).isInfinite() || ((Float)value).isNaN())  
131             out.write("null");  
132         else  
133             out.write(value.toString());  
134     }  
135     else if(value instanceof Number){  
136         out.write(value.toString());  
137     }  
138     else if(value instanceof Boolean){  
139         out.write(value.toString());  
140     }  
141     else if((value instanceof JSONStreamAware)){  
142         ((JSONStreamAware)value).writeJSONObject(out);  
143     }  
144     else if((value instanceof JSONAware)){  
145         out.write(((JSONAware)value).toJSONString());  
146     }  
147     else if(value instanceof Map){  
148         JSONObject.writeJSONObject((Map)value, out);  
149     }  
150     else if(value instanceof Collection){  
151         JSONArray.writeJSONArray((Collection)value, out);  
152     }  
153     else if(value instanceof byte[][]){  
154         JSONArray.writeJSONArray((byte[][])value, out);  
155     }  
156     else if(value instanceof short[][]){  
157         JSONArray.writeJSONArray((short[][])value, out);  
158     }  
159 }
```

در تصاویر بعدی ns-2 را مشاهده می کنیم که در واقع coverage معلوم نیست:

```
translate.goog  x | Google Keep  x | Default Mappi...  x | vim:rtorr.com  x | cw.sharif.edu  x | :14012-404041 x | Software_Engi... x | Software_Engi... x | XML Schema P... x | Coverage Repo... x + - _ 
translate.goog  x | Google Keep  x | Default Mappi...  x | vim:rtorr.com  x | cw.sharif.edu  x | :14012-404041 x | Software_Engi... x | Software_Engi... x | XML Schema P... x | Coverage Repo... x + - _ 
← C ⌂ File | /home/mmbhehnasr/University/software/LAB/SE_LAB/final_coverage/ns-2/sources/source-3.html
481     They will be read again by their next call of the scanning method
482
483     * @param number  the number of characters to be read again
484     *          This number must not be greater than yylength()!
485   */
486  public void yypushback(int number) {
487      if ( number > yylength() )
488          zzScanError(ZZ_PUSHBACK_2BIG);
489
490      zzMarkedPos -= number;
491  }
492
493 /**
494  * Assumes scanning until the next regular expression is matched,
495  * the end of input is encountered or an I/O-Error occurs.
496  *
497  * @return      the next token
498  * @exception  java.io.IOException  if any I/O-Error occurs
499  */
500
501 public Yytoken yylex() throws java.io.IOException, ParseException {
502     int zzInput;
503     int zzAction;
504
505     // cached fields:
506     int zzCurrentPosL;
507     int zzMarkedPosL;
508     int zzEndReadL = zzEndRead;
509     char [] zzBufferL = zzBuffer;
510     char [] zzCMapL = ZZ_CMAP;
511
512     int [] zzTransL = ZZ_TRANS;
513     int [] zzRowMapL = ZZ_ROWMAP;
514     int [] zzAttrL = ZZ_ATTRIBUTE;
515
516     while (true) {
517         zzMarkedPosL = zzMarkedPos;
518
519         yychar+= zzMarkedPosL-zzStartRead;
520
521         zzAction = -1;
522
523         zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;
524
525         zzState = ZZ_LEXSTATE[zzLexicalState];
526
527         zzForAction: {
528             while (true) {
529                 if (zzCurrentPosL < zzEndReadL)
530                     zzInput = zzBufferL[zzCurrentPosL++];
531                 else if (zzAtEOF) {
532                     zzInput = YYEOF;
533                     break zzForAction;
534                 }
535                 else {
536                     // store back cached positions
537                     zzCurrentPos = zzCurrentPosL;
538                     zzMarkedPos = zzMarkedPosL;
539                     boolean eof = zzRefill();
540                     // get translated positions and possibly new buffer
541                     zzCurrentPosL = zzCurrentPos;
542                     zzMarkedPosL = zzMarkedPos;
543                     zzBufferL = zzBuffer;
544                     zzEndReadL = zzEndRead;
545                     if (eof) {
546                         zzInput = YYEOF;
547                         break zzForAction;
548                     }
549                     else {
550                         zzInput = zzBufferL[zzCurrentPosL++];
551                     }
552                 }
553             }
554         }
555     }
556 }
```

```
translate.goog  x | Google Keep  x | Default Mappi...  x | vim:rtorr.com  x | cw.sharif.edu  x | :14012-404041 x | Software_Engi... x | Software_Engi... x | XML Schema P... x | Coverage Repo... x + - _ 
translate.goog  x | Google Keep  x | Default Mappi...  x | vim:rtorr.com  x | cw.sharif.edu  x | :14012-404041 x | Software_Engi... x | Software_Engi... x | XML Schema P... x | Coverage Repo... x + - _ 
← C ⌂ File | /home/mmbhehnasr/University/software/LAB/SE_LAB/final_coverage/ns-2/sources/source-3.html
187     "Error: could not match input";
188     "Error: pushback value was too large"
189 };
190
191 /**
192  * ZZ_ATTRIBUTE[aState] contains the attributes of state <code>aState</code>
193 */
194 private static final int [] ZZ_ATTRIBUTE = zzUnpackAttribute();
195
196 private static final String ZZ_ATTRIBUTE_PACKED_0 =
197     "\u2010\u1d13\u1d11\u1d10\u1d18\u1d11\u2113\u1d11";
198     "\u2010\u1d11\u1d10\u1d11\u1d10\u1d11\u0111\u0110\u2113";
199     "\u2010\u1d11";
200
201 private static int [] zzUnpackAttribute() {
202     int [] result = new int[45];
203     int offset = 0;
204     offset = zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
205     return result;
206 }
207
208 private static int zzUnpackAttribute(String packed, int offset, int [] result) {
209     int l = offset; // index in packed String '
210     int i = offset; // index in unpacked array '
211     int l = packed.length();
212     while (l < i) {
213         int value = packed.charAt(i++);
214         int value = packed.charAt(i++);
215         do result[i++] = value; while (--count > 0);
216     }
217     return j;
218 }
219
220 /** the input device */
221 private java.io.Reader zzReader;
222
223 /** the current state of the DFA */
224 private int zzState;
225
226 /** the current lexical state */
227 private int zzLexicalState = YYINITIAL;
228
229 /** this buffer contains the current text to be matched and is
230  * the source of the yytext() string */
231 private char zzBuffer[] = new char[ZZ_BUFFERSIZE];
232
233 /** the textposition at the last accepting state */
234 private int zzMarkedPos;
235
236 /** the current text position in the buffer */
237 private int zzCurrentPos;
238
239 /** startRead marks the beginning of the yytext() string in the buffer */
240 private int zzStartRead;
241
242 /** endRead marks the last character in the buffer, that has been read
243  * from input */
244 private int zzEndRead;
245
246 /** number of newlines encountered up to the start of the matched text */
247 private int yyline;
248
249 /** the number of characters up to the start of the matched text */
250 private int yychar;
251
252 /**
253  * the number of characters from the last newline up to the start of the
254  * matched text
255  */
256 private int yccolumn;
257
258 /**
```

درصدهای coverage نهایی نیز در تصویر زیر آمده که نشان می‌دهد با اضافه کردن تست‌های نوشته شده coverage بهبود یافته:

The screenshot shows a browser window with multiple tabs open. The active tab is titled 'Coverage Repo' and contains a coverage report for Java code. The report includes an 'Overall Coverage Summary' table and a 'Coverage Breakdown' table. The 'Overall Coverage Summary' table shows a Class, % of 85.7% (6/7), a Method, % of 41.9% (39/93), and a Line, % of 33% (260/787) for 'all classes'. The 'Coverage Breakdown' table details coverage for two packages: 'org.json.simple' with 100% (3/3) Class, 41.9% (18/43) Method, and 32.3% (94/291) Line; and 'org.json.simple.parser' with 75% (3/4) Class, 42% (21/50) Method, and 33.5% (166/496) Line. The bottom right corner of the report page indicates it was generated on 2023-04-14 23:45.

Package	Class, %	Method, %	Line, %
all classes	85.7% (6/7)	41.9% (39/93)	33% (260/787)

Package	Class, %	Method, %	Line, %
org.json.simple	100% (3/3)	41.9% (18/43)	32.3% (94/291)
org.json.simple.parser	75% (3/4)	42% (21/50)	33.5% (166/496)