

# Broken Links Documentation

Badoi Mircea Aurelian

April 29, 2020

## 1 Problem Statement

Design an application which takes an URL and displays all the links found on the page. Application could also shows a sorted list alphabetically of the links, a sorted list by time response and as well as filters the link by status DOWN(broken).

## 2 Problem Solving

In order to find the broken links, we have followed the next steps:

- collect the links found on the given page
- send request to the server for all links found and read the response code
- based on response from server, the links are categorized by DOWN or UP

For sorting we use sort method of ArrayList by writing a comparator which suits our needs.

Filtering it is just a traverse of the list of links. We keep just the links with status DOWN.

## **3 Application outline**

### **3.1 The high level architectural overview of the application**

Application is divided in 3 main folders.

-first one (src) contains the interfaces and classes for solving the requirements

-the second one (test) has the JUnit Tests for:

- FilterListOfLinksByDown class

- LinkTypeBrokenCheck class

- SortListOfLinksByName class

- SortListOfLinksByTimeResponse class.

-the third one (GUI) where we build the interface with the user using Java Swing. It also contains MainFrameController class which controls the app.

### **3.2 The specification of the input data format**

Input data is given by the user as a valid URL. In case of an invalid link, the application displays "ERROR...TRY AGAIN".

### **3.3 The specification of the output data format**

The output represent a list of outgoing links, the time response and the status of them.

### 3.4 The list of all the files in the application and their description

Classes and Interfaces:

-src folder:

-WebLinkType class it's a type of data. It is composed of an enum status (statusLink), a string (url) and an int(timeResponseServer)

-FilterListOfLinksByAction interface has one function filterBy

-FilterListOfLinksByDown implements the upper interface. Function filterBy takes a list of WebLinkType and returns an ArrayList with the DOWN links

-FindLinkAction interface has one function findAllLinks

-FindLinks implements the interface FindLinkAction. Function findAllLinks takes a String and returns an ArrayList of URLs found on a page(for this we use JSoup).

-LinkTypeAction interface has two functions, isLinkBroken and responseTimeServer

-LinkTypeBrokenCheck implements LinkTypeAction. Function isLinkBroken takes an URL type element and a WebLinkType, try to connect to the server, acknowledge the response and set the status of WebLinkType element.

-SortOnListOfLinksAction interface has one function sortBy.

-SortListOfLinksByName implements SortOnListOfLinksAction. Function sortBy takes a list and sort alphabetically it.

-SortListOfLinksByTimeResponse implements SortOnListOfLinksAction. Function sortBy takes a list and sort by time response it.

-test folder:

-FilterByDownStatusTest class

-LinkTypeBrokenCheckTest class

-SortListByNamesTest class

-SortListByTimeResponseTest class.

-GUI folder:

-Main class

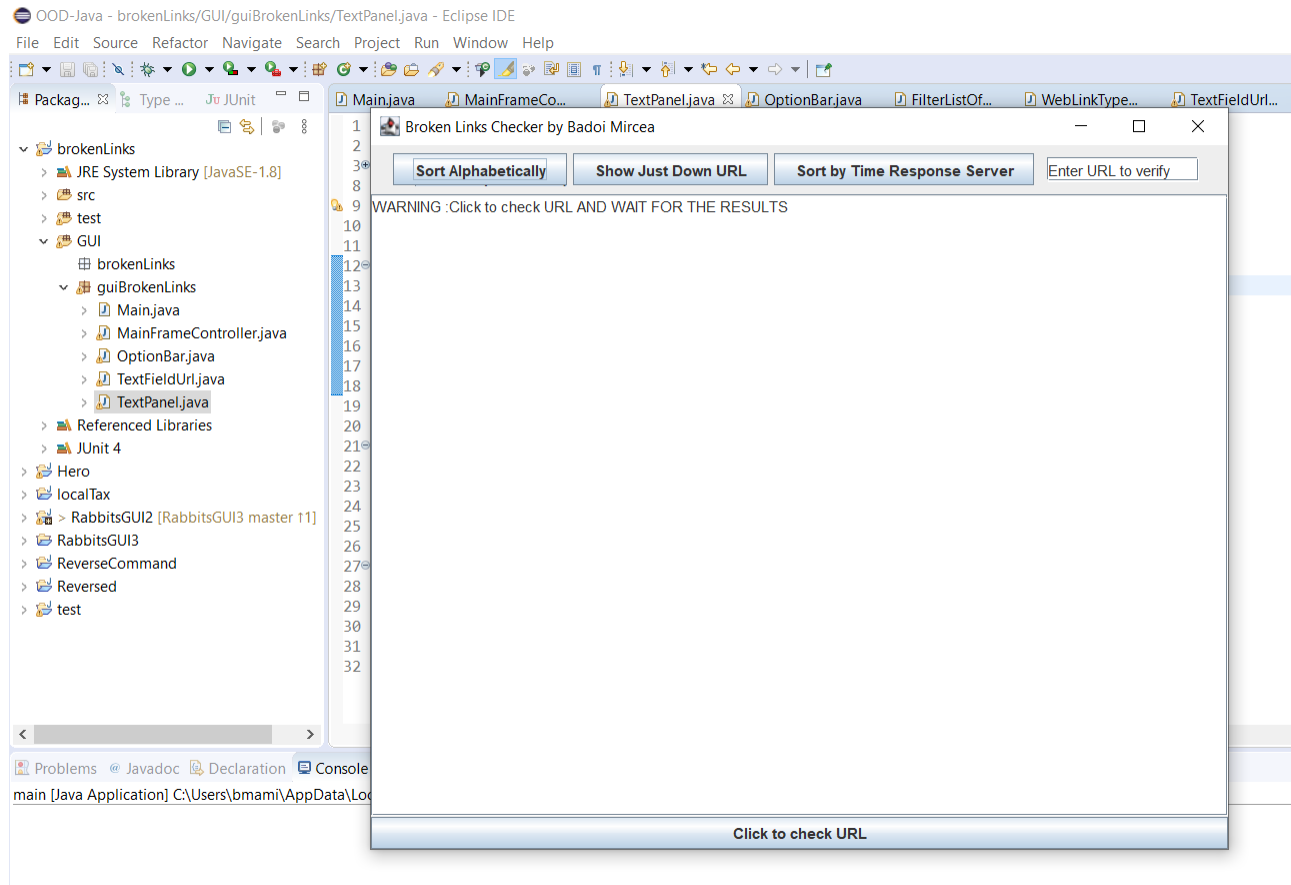
-MainFrameController class set the preconditions of the program and build the necessary components of app

-OptionBar class makes the tool bar with three buttons, two for sorting and one to filter. It contains also a text field where you introduced the URL

-TextFieldUrl class contains the text field of the tool bar

-TextPanel class represents the display area of the app

### 3.5 Application design



The user is suppose to enter the URL in the upper right text field and click the Click to check URL button from the bottom of the app.

## 4 Experimental data

The tests were made with JUnit for the important classes of the app.

The screenshot shows the Eclipse IDE with the following components:

- Top Menu Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard Eclipse development tools.
- Left Panel:**
  - Packages View:** Shows the package structure: brokenLinks.FilterByDownStatusTest [Run].
  - JUnit View:** Shows the test execution results: "Finished after 0.036 seconds", "Runs: 3/3", "Errors: 0", "Failures: 0".
  - Failure Trace View:** Currently empty.
- Main Editor:** Displays the source code of `FilterByDownStatusTest.java`.
 

```

1 package brokenLinks;
2
3 import static org.junit.Assert.*;
4
5 @RunWith(Parameterized.class)
6
7 public class FilterByDownStatusTest {
8     private ArrayList<WebLinkType> actualOutput;
9     private ArrayList<WebLinkType> functionReturnResultHelper;
10    private FilterListOfLinksByAction helper;
11    private ArrayList<status> actualStatusOutputConvertedToArrayList;
12    private WebLinkType temp;
13    ArrayList<status> expectedServerStatus = new ArrayList<status>();
14    ArrayList<status> inputServerStatus = new ArrayList<status>();
15
16    // constructor in order to use automate tests
17    public FilterByDownStatusTest(ArrayList<status> expectedServerStatus, ArrayList<status> inputServerStatus) {
18        super();
19        this.expectedServerStatus = expectedServerStatus;
20        this.inputServerStatus = inputServerStatus;
21    }
22
23    @Before
24    // initialise the needed variables before every test
25    public void init() {
26        actualOutput = new ArrayList<WebLinkType>();
27        functionReturnResultHelper = new ArrayList<WebLinkType>();
28        helper = new FilterListOfLinksByDown();
29        actualStatusOutputConvertedToArrayList = new ArrayList<status>();
30    }

```

OOD-Java - brokenLinks/test/brokenLinks/LinkTypeBrokenCheckTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

JUnit

Finished after 5.068 seconds

Runs: 2/2 Errors: 0 Failures: 0

brokenLinks.LinkTypeBrokenCheckTest [F

```

1 package brokenLinks;
2
3 import static org.junit.Assert.*;
4
5 @RunWith(Parameterized.class)
6
7 public class LinkTypeBrokenCheckTest {
8     private LinkTypeAction test;
9     private ArrayList<String> expectedResponsesFromServer = new ArrayList<String>();
10    private ArrayList<String> actualResponsesFromServer;
11    private ArrayList<String> inputLinksToCheck = new ArrayList<String>();
12
13    // constructor in order to use automate tests
14    public LinkTypeBrokenCheckTest(ArrayList<String> expectedResponsesFromServer, ArrayList<String> inputLinksToCheck)
15    {
16        super();
17        this.expectedResponsesFromServer = expectedResponsesFromServer;
18        this.inputLinksToCheck = inputLinksToCheck;
19    }
20
21    @Before
22    // initialise the needed variables before every test
23    public void init() {
24        test = new LinkTypeBrokenCheck();
25        actualResponsesFromServer = new ArrayList<String>();
26    }
27
28    @Parameters
29    // parameters(input and output)for automate tests
30    public static Collection<Object[]> data() {
31
32    }
33
34    }

```

Failure Trace

OOD-Java - brokenLinks/test/brokenLinks/SortListByNamesTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

JUnit

Finished after 0.022 seconds

Runs: 2/2 Errors: 0 Failures: 0

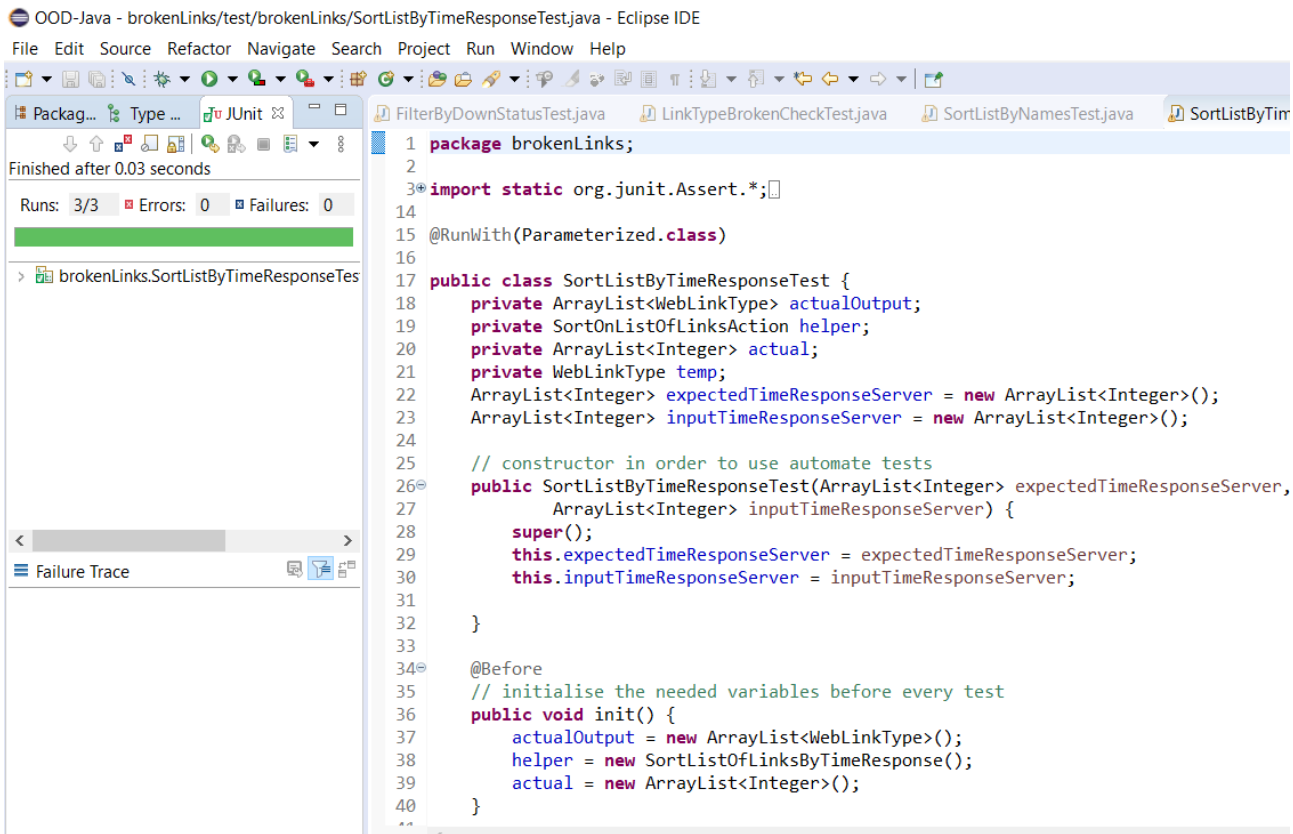
brokenLinks.SortListByNamesTest [Runne

```

1 package brokenLinks;
2
3 import static org.junit.Assert.*;
4
5 @RunWith(Parameterized.class)
6
7 public class SortListByNamesTest {
8     private ArrayList<WebLinkType> actualOutput;
9     private SortOnListOfLinksAction helper;
10    private ArrayList<String> actualConvertedToArrayList;
11    private WebLinkType temp;
12    ArrayList<String> expectedOrderOfNames = new ArrayList<String>();
13    ArrayList<String> inputOrderOfNames = new ArrayList<String>();
14
15    // constructor in order to use automate tests
16    public SortListByNamesTest(ArrayList<String> expectedOrderOfNames, ArrayList<String> inputOrderOfNames)
17    {
18        super();
19        this.expectedOrderOfNames = expectedOrderOfNames;
20        this.inputOrderOfNames = inputOrderOfNames;
21    }
22
23    @Before
24    // initialise the needed variables before every test
25    public void init() {
26        actualOutput = new ArrayList<WebLinkType>();
27        helper = new SortListOfLinksByName();
28        actualConvertedToArrayList = new ArrayList<String>();
29    }
30
31    }

```

Failure Trace



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons. The left sidebar contains a 'JUnit' tab showing the execution results of a test. It indicates 'Finished after 0.03 seconds', 'Runs: 3/3', 'Errors: 0', and 'Failures: 0'. Below this is a 'Failure Trace' section. The main editor window displays the source code of the file 'SortListByTimeResponseTest.java'. The code is as follows:

```
1 package brokenLinks;
2
3 import static org.junit.Assert.*;
14
15 @RunWith(Parameterized.class)
16
17 public class SortListByTimeResponseTest {
18     private ArrayList<WebLinkType> actualOutput;
19     private SortOnListOfLinksAction helper;
20     private ArrayList<Integer> actual;
21     private WebLinkType temp;
22     ArrayList<Integer> expectedTimeResponseServer = new ArrayList<Integer>();
23     ArrayList<Integer> inputTimeResponseServer = new ArrayList<Integer>();
24
25     // constructor in order to use automate tests
26 public SortListByTimeResponseTest(ArrayList<Integer> expectedTimeResponseServer,
27     ArrayList<Integer> inputTimeResponseServer) {
28     super();
29     this.expectedTimeResponseServer = expectedTimeResponseServer;
30     this.inputTimeResponseServer = inputTimeResponseServer;
31
32 }
33
34 @Before
35 // initialise the needed variables before every test
36 public void init() {
37     actualOutput = new ArrayList<WebLinkType>();
38     helper = new SortListOfLinksByTimeResponse();
39     actual = new ArrayList<Integer>();
40 }
```

## 5 Conclusion

Working at this application helped me gain more experience in JUnit, Java Swing and also it helped me understand how it works. I managed to make me study more on Java programming language and gain more coding ability experience.

From my point of view, the program I implemented, as well as, the tools I used for this app, are correct and both compute the right answer.

## References

- [1] <https://jsoup.org/?fbclid=IwAR1zWNpsbhz4iU2KzWY05UomM0LU0IT3cZE5TbiDZYeVK17W9XCnwG4uxS8>, accessed in April 2020.
- [2] [geeksforgeeks.org/?fbclid=IwAR0Y2jkrkTRgwGEuLZxznVgZHwJybaqvFaUG1KKBTbiKxMk\\_\\_\\_rH1z7xb4](https://www.geeksforgeeks.org/?fbclid=IwAR0Y2jkrkTRgwGEuLZxznVgZHwJybaqvFaUG1KKBTbiKxMk___rH1z7xb4), accessed in April 2020.
- [3] <https://www.javatpoint.com/java-swing?fbclid=IwAR2nqA8PeQDXvupzmEpJNEm6pWlwcQI8Q5stKF688Nu3GtvNz>, accessed in April 2020.