

Black-Box Testing of Csv2Html.java

Software Report Document

Pratik Shekhar

SE 320

Software Verification and Validation

Instructor: Colin S. Gordon

October 31, 2016

Table of Contents

1. Introduction	i
1.1 Purpose	i
1.2 Scope/Functional description of the program	i-ii
1.2.1 Output Format	ii
1.2.2 Access to Csv2Html.java	ii
1.2.3 Software System Attributes	ii
1.2.3.1 Availability	ii
1.2.3.2 Security	ii
1.3 Definitions, Acronyms, and Abbreviations	iii
1.4 References	iii
1.5 Overview	iii
 2. Test Cases	 iv-x
 3. Evaluation	 x

1. Introduction

1.1 Purpose

The purpose of this document is to specify the requirements from testing perspectives of a command line utility and its reusable Java module convert CSV files into HTML tables. These requirements describe the areas pertaining to Availability, and Security.

This document is intended for any individual user, developer, tester, project manager or documentation writer that needs to understand the basic system architecture and its specifications in order to test using black box testing techniques. Here are the potential uses for each one of the reader types:

- • **Developer:** The developer who wants to read, change, modify or add new requirements into the existing program, must firstly consult this document and update the requirements with appropriate manner so as to not destroy the actual meaning of them and pass the information correctly to the next phases of the development process.
- • **User:** The user of this program reviews the specifications presented in this document and determines if the software has all the suitable requirements and if the software developer has implemented all of them.
- • **Tester:** The tester needs this document to validate that the initial requirements of this programs actually corresponds to the executable program correctly.

1.2 Scope/Functional Description of the Program

Csv2Html.java is an application/command line Java program for converting CSV (comma separate value) files to HTML tables. In addition to the basic use of commas to separate fields of the table, this version also supports escaping commas with a \ or use of double quotes. By default it takes a file name on the command line (a .csv file) and turns it into a .html file of the same base name. If two filenames are given, it reads the first as a CSV file and writes the output HTML to the second. An output file is chosen if only the input is given on the command line.

- The default CSV format is agnostic about things like escaping characters of the input.
- This implementation:
 - Escapes various HTML entities as appropriate (< > & ; \ ")
 - Uses backslash escapes to permit commas inside cells
 - Inside double-quotes, commas are interpreted as normal characters, while other special
- characters may be escaped with backslashes.
- It takes two command line arguments. First, a mandatory input filename. Second, an optional.
- output filename: If no output file is specified, a .html file with the same base name as the input file is used (e.g., foo.csv is converted into a table in foo.html).

Black-Box Testing of Csv2Html.java

- You can use your favorite browser to view the input files (most have an "Open File" option, which directs the browser to a local file using the file:// protocol).

1.2.1 Output format:

After reading the CSV file and using the Csv2Html.java, the file will output the contents of the CSV file data in a Html file for the user to view.

1.2.2 Access to the Csv2Html.java:

Pre-conditions:

The application must be launched and compiled. The user must compile it correctly. In order to compile the program, the user must follow the instructions below:

type : javac Csv2Html.java and hit enter
type : java Csv2Html <“.csv” file name >

Base course of action:

As soon as user calls Test data with Csv2Html.java, the program will be executed.

Alternate actions:

If the user calls any random Test data not present in the directory with Csv2Html.java, the program will flag an error.

Post-conditions:

The program will be executed if it compiles successfully.

1.2.3 Software System Attributes

1.2.3.1 Availability

Checking that the system always has something to function and always pop up error messages in case of component failure. In that case the error messages appear when something goes wrong so to prevail availability problems.

1.2.3.2 Security

This program uses object oriented mechanisms to protect its data passed using methods There is currently no security schema of this program.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:

Resources: The resource is a test application property defines the file to read in “.csv” format and uses Csv2Html.java

Black Box Testing: Testing without having an insight into the details of the underlying code. Sometimes referred to as a functional or behavioral testing. It requires an executable program.

Test Data: Inputs which have been devised to test the system.

Test Cases: Inputs to test the system and the predicted outcomes from these inputs if the system operates according to its specification.

The category-Partition Method: The main idea is to systematically partition the input domain of function being tested, and then select data for each class of the partition

Parameters: Explicit inputs to a functional unit supplied by the user or another program. (Also sometimes *qualities* of those parameters.)

Environment Conditions: Characteristics of the system’s state at the time of executing a functional unit.

Category: A major property or characteristic of a parameter or environment condition. For example, the “pattern size”

Choice: A set of similar values that can be assumed by the type of information in the category. They are the partition classes. For example, the pattern is empty or the pattern consists of multiple characters.

1.4 References

- “Class CsvReader”. Web. 30 Jan. 2016. <http://javacsv.sourceforge.net>

1.5 Overview

The rest of the document describes the each test case used to test program, along with the output generated by the program and an evaluation of testing activity.

2. Test Cases

Test Case 1: Passing single test data - “.csv” file :

This is a type of test case with an intent to pass.

I used a sample test data named demo2.csv to test the program.

Syntax: `java Csv2Html demo2.csv`

It generated a demo2.html file in the directory. When you click on the demo.html file, it opens up in the default browser, (safari in my case) with the generated output in a tabular format.

```
1  A,B
2  X,Y,Z
3
```

X,Y	Z
-----	---

Figure 1 and 2 : demo2.csv file and the generated output in tabular format.

Test Case 2: Passing two test data i.e, two “.csv” file :

This is a type of test case with an intent to pass.

Used two test data, namely demo.csv and demo2.csv to test the program.

Syntax: `java Csv2Html demo.csv demo2.csv`

It doesn't generate any Html file. Instead, it reads the first as a CSV file and writes the output HTML to the second.

Test Case 3: Passing multiple test data

case 1: All the three file with same content

Used 3 test data, each with the same content in it.

Syntax: `Csv2Html demo.csv demo3.csv demo4.csv`

```
Title,Author
Developer Testing, Tarlinder, Alexander
Software Testing, "Patton, Ron"
```

Black-Box Testing of Csv2Html.java

It doesn't generate any Html file. Instead, it reads the first csv file, namely demo.csv and writes the output in Html format to demo3.csv and doesn't read demo4.csv at all. Thus, there is no change in demo4.csv file.

```
Title,Author
Developer Testing, Tarlinder, Alexander
Software Testing, "Patton, Ron"
<html>
<body>
<table border="1">
  <tr>
    <td>Titles</td>
    <td>Authors</td>
  </tr>
  <tr>
    <td>Developer Testing</td>
    <td>Tarlinder, Alexander</td>
  </tr>
  <tr>
    <td>Software Testing</td>
    <td>Patton, Ron</td>
  </tr>
</table>
</body>
</html>
```

```
Title,Author
Developer Testing, Tarlinder, Alexander
Software Testing, "Patton, Ron"
```

Figures: 3,4, and 5: demo.csv, demo3.csv, demo4.csv

case 2: All the three file with different content

Used three different test data, each with different content in csv format in it.

Syntax: Csv2Html demo.csv demo3.csv demo4.csv

It doesn't generate any Html file. Instead, it reads the first csv file, namely demo.csv and writes the output in Html format to demo3.csv and doesn't read demo4.csv at all. Thus, there is no change in demo4.csv file.

To conclude, if multiple files are passed, we can say that the program is only capable to read the first csv file and writes the output to the second csv file in Html format . It negates/ doesn't even do anything to the test data passed after the second csv file. Incase the first csv file has already been executed at least once, the program if executed again with the first csv file, it will read the first csv file, write the output in Html format to the second file below the table generated previously when it was last executed.

Test Case 4: Passing output file with the test data to generate a desired Html file :

Used a single test data, namely demo2.csv file and passed an output file desired.html to see if the program generates the correct output file in the Html format with the given name.

Syntax: Csv2Html demo2.csv desired.html

It generated the desired Html output file by the name provided.

Black-Box Testing of Csv2Html.java

Note: If the outfield file is not passed with “.html” extension, it just generates a file with the name provided and html code in it. For example:

Syntax: Csv2Html demo2.csv desired

It will generate a file named desired in the directory which will contain the html syntax of demo2.csv file.

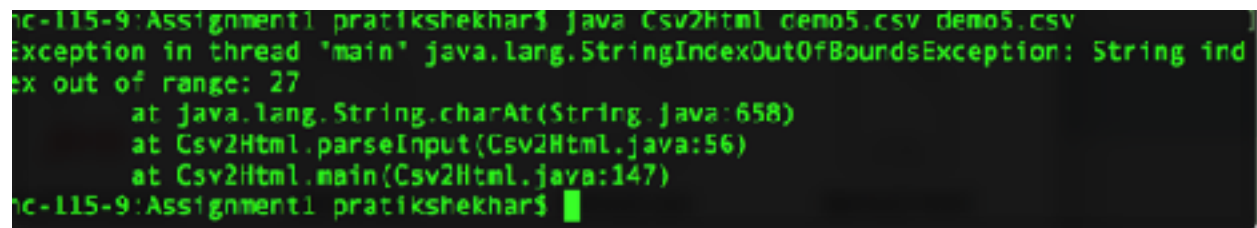
Test Case 5: Passing two same test data :

This is a type of test case with an intent to fail.

Used two same test data in order to test the application.

Syntax: Csv2Html demo5.csv demo5.csv

The program generated an error as expected.



```
nc-115-9:Assignment1 pratikshekhar$ java Csv2Html demo5.csv demo5.csv
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String ind
ex out of range: 27
    at java.lang.String.charAt(String.java:658)
    at Csv2Html.parseInput(Csv2Html.java:56)
    at Csv2Html.main(Csv2Html.java:147)
nc-115-9:Assignment1 pratikshekhar$
```

Figure 6: Shows an error with two same csv file is passed side by side.

Test Case 6: Multiple test data with the same name :

Unfortunately, the directly doesn't save two file with same name.

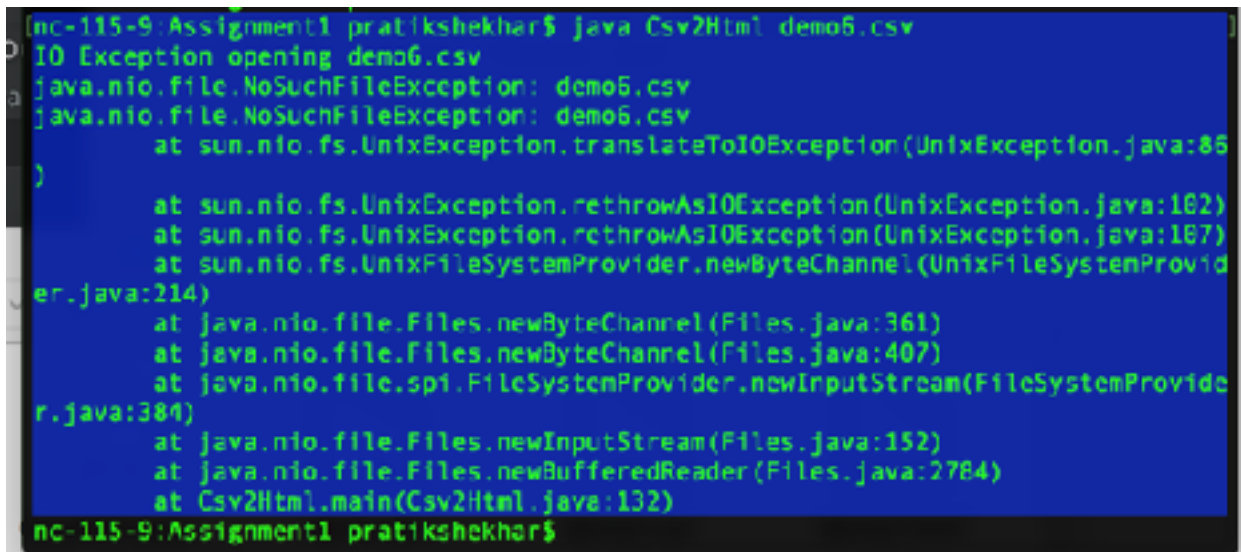
Test Case 7: Given test data not present in the directory / not found :

This is a type of test case with an intent to fail.

Syntax: Csv2Html demo6.csv

The test data demo6.csv file doesn't exist in the directory. Therefore, if you try to run the program, it should flag an error. The program generated an error as expected.

Black-Box Testing of Csv2Html.java



```
nc-115-9:Assignment1 pratikshekhar$ java Csv2Html demo6.csv
IO Exception opening demo6.csv
java.nio.file.NoSuchFileException: demo6.csv
java.nio.file.NoSuchFileException: demo6.csv
    at sun.nio.fs.UnixException.translateToIOException(UnixException.java:85)
    at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:102)
    at sun.nio.fs.UnixException.rethrowAsIOException(UnixException.java:107)
    at sun.nio.fs.UnixFileSystemProvider.newByteChannel(UnixFileSystemProvider.java:214)
    at java.nio.file.Files.newByteChannel(Files.java:361)
    at java.nio.file.Files.newByteChannel(Files.java:407)
    at java.nio.file.spi.FileSystemProvider.newInputStream(FileSystemProvider.java:384)
    at java.nio.file.Files.newInputStream(Files.java:152)
    at java.nio.file.Files.newBufferedReader(Files.java:2784)
    at Csv2Html.main(Csv2Html.java:132)
nc-115-9:Assignment1 pratikshekhar$
```

Figure 7: The program generates an error if the file doesn't exist

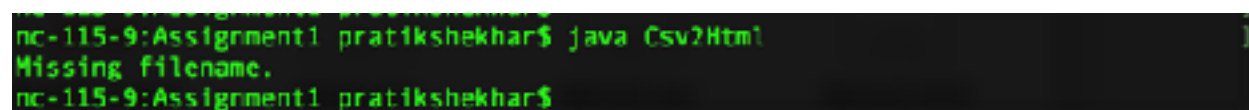
Test Case 8: Test data not passed :

This is a type of test case with an intent to fail.

This is a type of test that should fail if no test data is passed.

Syntax: java Csv2Html

The program displays a message of missing filename as expected because no test data was passed.



```
nc-115-9:Assignment1 pratikshekhar$ java Csv2Html
Missing filename.
nc-115-9:Assignment1 pratikshekhar$
```

Figure 8: A missing filename message is displayed if no test data is passed to the program.

Test Case 9: Test data of different format other than csv :

This is a type of test case with an intent to fail.

Used a test data of different format (".rtf" in this case) other than csv.

Syntax: Csv2Html demo6.rtf

Black-Box Testing of Csv2Html.java

The program generates an error as expected.

```
nc-115-9:Assignment1 pratikshekhar$ java Csv2Html demo6.rtf
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: 27
    at java.lang.String.charAt(String.java:658)
    at Csv2Html.parseInput(Csv2Html.java:56)
    at Csv2Html.main(Csv2Html.java:147)
nc-115-9:Assignment1 pratikshekhar$
```

Figure 9: Test case fails as the test data is of different format.

Test Case 10: Test data empty :

Used empty test data as in without any content in it.

Syntax: *Csv2Html demo.csv*

It generated an empty html file without any tables as there weren't any data in the csv file.

Test Case 11: Test data with embedded commas :

Field must be delimited with double-quotes

Syntax: *Csv2Html demo.csv*

It generated the Html file with the desired output.

Title	Author
Developer Testing	Tarlinder, Alexander
Software Testing	Patton, Ron

```
Title,Author
Developer Testing, Tarlinder\, Alexander
Software Testing, "Patton, Ron"
```

Figure 10 and 11: Test case with embedded comma

Test case 12: Test data with embedded double quotes :

Embedded double quote characters may then be represented by a pair of consecutive double quotes, or by prefixing an escape character such as a backslash.

Syntax: *Csv2Html demo.csv*

It generated the Html file with the desired output.

Title	Author
Developer Testing	Tarlinder, Alexander
Software Testing	"Patton, Ron"

Black-Box Testing of Csv2Html.java

```
. Title,Author
: Developer Testing, Tarlinder\, Alexander
: Software Testing, "\"Patton, Ron\""
|
```

Figure 12 and 13: Test case with embedded double quotes

Test Case 13: Test data with special characters :

Used test data which had a special character instead of comma.

Syntax: Csv2Html demo.csv

The program generated the html file but the special character was also included. It didn't flag any error or separated the values.

```
Title,Author
Developer Testing, Tarlinder @ Alexander
Software Testing, "Patton, Ron"
```

Title	Author
Developer Testing	Tarlinder @ Alexander
Software Testing	Patton, Ron

Figure 15 and 16: Test data with special character

Test Case 14: Test data with leading and trailing whitespace :

Used test data with trailing white spaces in it.

Syntax: Csv2Html demo.csv

The program ignored the whitespace. However, Unless the field is delimited with double-quotes in that case the whitespace is preserved.

```
Title,Author
Developer Testing, Tarlinder      \,      Alexander
Software Testing, "Patton, Ron"
|
```

Title	Author
Developer Testing	Tarlinder , Alexander
Software Testing	Patton, Ron

Figure 17 and 18: Test data with whitespace

Test Case 15: Test data with no write permission :

Used test data where the user has no write permission, and tried to run the program.

Syntax: Csv2Html demo.csv

It generates an Html file.

3. Evaluation

Did your testing this time reveal any new bugs?

If I understand this question correctly, I realized there are situations and test cases which is simply not enough to test the program using black box testing or be certain that a particular feature of the program is being tested the way it is supposed to and not being confused as a bug. All the test cases that I have behaved the way I expected, so my perspective is that If we execute the program with an intent that all the test cases should pass and if any one of them fails, we have a bug or with an intent to fail and it passes, then we have a bug. I'm not certain that certain things that I encountered while testing were features of the program. The reason I say i'm not certain because it didn't give me any errors. So, i'm confused if that was a feature or actually a bug that I'm not aware of as I didn't write this program.

Do you consider some of the test cases to be more valuable / important than others? If yes, which ones are more important and why?

I believe that all the test cases are valuable/important as each test cases specifies and test a unique features of the program and make sure it works correctly with as little bug as possible. However, I do believe that test case 5 and test case 7 are important than others. The reason I say this because, if same test data with same names are passed in the program or given test data not present in the directory, the program should be able to handle the problem, otherwise it will result in reading incorrect files or generating a random Html file which doesn't exist and producing incorrect outputs if no validation check is done.

Do you believe you have adequately tested your program? How do you know?

Software testing is a risk based activity. Even though we can test the few of the important components of Csv2Html.java application using black box testing techniques but we can not be certain that we have tested it adequately. What we can get from risk-based testing is to carry out the testing with best practices in risk management to achieve a project outcome that balances risks with quality, features, budget and schedule. Therefore, I don't think that I have tested my program adequately.