CS362 Final Report
Andy Garcia
Kendra Swafford
Anna Murphy

I. Methodology

For manual testing, the tested valid URLs include “http://google.com”, “http://google.com/test1/file”, and “http://google.com:65636”. The tested invalid URLs include “”httpgoogle”, “http://google.com:-1”, and “file:///c:\\”. These tests were designed around the latest URL specification from W3 (as further discussed below in the input partitioning discussion section).

For partitioning, the partitions are based on the different components of a URL, where the whole input set is defined as the set of all strings, and an equivalence class is defined on valid and invalid substrings. For example, a generic URL has the form:

“scheme://authority:port/path?query”

So the first partition contains those strings composed of an invalid scheme and valid authority, port, path, and query. The second partition contains strings with a valid scheme, invalid authority, and valid port, path, and query. The third partition contains strings with valid scheme and authority, invalid port, and valid path and query. The fourth partition contains strings with valid scheme, authority, and port, invalid path, and valid query. The fifth partitions contains strings with valid scheme, authority, port, and path, and an invalid query. The sixth partition contains only strings which are *required* for a URL (in this case, a valid authority + empty strings).

The specific substrings were designed using the latest URL specification from W3 (https://url.spec.whatwg.org/): a valid scheme must either be the empty string or begin with one ASCII alpha character, followed by zero or more alphanumeric characters, and separated from the authority with “://”. A valid authority must be a registered name or IP address. A valid port must be zero or more ASCII digits separated from the authority with “:”. A valid path must begin with “/” and be followed by a path-relative URL, which must be a path segment (composed of zero or more ASCII alphanumeric URL units) *not* beginning with “/”. A valid query must begin with “?” and contain zero or more URL units. By partitioning this way, each instruction of the URLValidator algorithm should be covered by the equivalence classes.

The names of the unit tests we devised are permTest() (which constructs permutations of different URL component combinations), testIsValidScheme(), testIsValidAuthority(), testIsValidPath(), testIsValidPort(), testIsValidQuery(), and isValidUnitTest1() (which allows for user-provided test cases at runtime).

II. Collaboration

The team used email and Google Hangouts to communicate with each other. Our first meeting was via video conference where we discussed our general approach to the project and divided up work based on previous experience and current availability. Because Anna had two years experience with Java, she volunteered to be the main contributor to the programming based test cases. Since Kendra works mainly from a Chromebook, Andy offered to take responsibility for running the tests and writing bug reports. Kendra elected to write the manual and input partitioning tests and compile the report. The group continued to touch base through email and Google Hangouts with biweekly progress meetings.

To be confident that the team kept consistent code, it was decided that Anna's project folder in the class repo would be the base of operations and that each member would pull and push using Git any time changes were made to the URLValidator. The team also used shared files on Google Drive to make sure that everyone had the same understanding of tasks and action items.

Regarding coding, the team worked largely independently, though each member looked over the code for errors and bugs in the tester, which were communicated to one another via email.

III. Bug Reports

Each bug was found by running the tests and the setting a breakpoint in each of tests that were erroring. From there, we stepped through each line of code until the line of code that triggered the error was found. This is the line number that is provided.

We followed Agan's rules pretty close by only changing one thing at a time. It was very helpful to use these principles when developing our test so that we could take a solid guess at where the noted bug was. This was one of the principles that mattered the most to us during development so that we can see exactly what breaks things.

Another principle that we implemented heavily was "Quit thinking and look." We decided to use the Eclipse debugging tool to step through each process in the errored function and see exactly what the values in the variables held, which eventually brought us to find the exact line of code that caused a fault. In one of the tests, we actually narrowed it down to the regex by easily seeing what the defined regex pattern was in the debugger and testing that against an online regex tool.

A problem that occured with our debugging (and semi-fixing along the way) was Agan principle number 9: "If it you didn't fix it, it ain't fixed." There was a bug that was triggered

before another bug could be triggered, which was the port bug and the scheme bug. We fixed the port bug, but then it still failed because we forgot to fix the scheme bug.

Finally, we followed his first principle, “Understand the System”, by trying to accrue some subject knowledge before writing our tester. This included not only getting familiar with the original URLValidator code, but also researching the current W3 standards for what a valid URL actually is. By doing a little homework, we were more confident that our testing approaches made sense.

Manual Test Fails when no scheme given

Created: 13/Mar/16 Updated: 13/Mar/16

Status:	Open
Project:	Commons Validator CS 362 Final Project

Type:	Bug	**Priority:**	Major
Reporter:	Andy Garcia, Kendra Swafford, Anna Murphy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	easyfix		
Remaining Estimate:	Not Specified		
File: Line	UrlValidator.java : 296,336		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		

Description

In UrlValidatorTest.java, this is the code snippet that generates the error.
URLValidator/src/UrlValidatorTest.java

```
if (urlVal.isValid(url.item) == true && url.valid == true)
        {
           System.out.println("PASS: " + url.valid + " -- " + url.item);
        }
else if (urlVal.isValid(url.item) == false && url.valid == false)
        {
           System.out.println("PASS: " + url.valid + " -- " + url.item);
        }
else {
           System.out.println("-> FAIL: " + url.valid + " -- " + url.item);
```

```
        }
```

In this case, both manual URLs given are known to be valid URLs, but both are resolved as invalid when tested with the isValid method.

Since the scheme is null, isValidScheme returns false upon checking for this. This does not into account that a scheme is not necessary.

Example to generate error
url.item == www.google.com, google.com, and google.com// and url.valid == true

Scheme failed with valid Scheme

Created: 13/Mar/16 Updated: 13/Mar/16

Status:	Open
Project:	Commons Validator CS 362 Final Project

Type:	Bug	**Priority:**	Major
Reporter:	Andy Garcia, Kendra Swafford, Anna Murphy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	easyfix		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
File: line	UrlValidator.java : 340		

Description

In UrlValidatorTest.java, this is the code snippet that generates the error.
URLValidator/src/UrlValidatorTest.java

```
// Test a valid scheme
      UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
      System.out.println("Testing schemes:\n");

      String validScheme = "http://";
      if (!urlVal.isValidScheme(validScheme)) {
         System.out.println("ERROR: " + validScheme + " failed with valid scheme.\n");
      }
      else {
         System.out.println(validScheme + " passed with valid scheme.\n");
```

```
    }
```

The code should return as passing with an a valid scheme, but instead fails with an invalid scheme. We believe it is due to the regex.

Query failed with a valid query

Created: 13/Mar/16 Updated: 13/Mar/16

Status:	Open
Project:	Commons Validator CS 362 Final Project

Type:	Bug	**Priority:**	Major
Reporter:	Andy Garcia, Kendra Swafford, Anna Murphy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	easyfix		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
File: line	UrlValidator.java : 446		

Description

In UrlValidatorTest.java, this is the code snippet that generates the error.
URLValidator/src/UrlValidatorTest.java

```
 UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
       System.out.println("Testing queries:\n");

       // Test a valid query
       String validQuery = "?action=view";
       if (!urlVal.isValidQuery(validQuery)) {
          System.out.println("ERROR: " + validQuery + " failed with valid query.\n");
       }
       else {
          System.out.println(validQuery + " passed with valid query.\n");
```

```
    }
```

The code should return as passing with an a valid scheme, but instead fails with an invalid query.

Partition Testing (6): failed with valid authority

Created: 13/Mar/16 Updated: 13/Mar/16

Status:	Open
Project:	Commons Validator CS 362 Final Project

Type:	Bug	**Priority:**	Major
Reporter:	Andy Garcia, Kendra Swafford, Anna Murphy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	easyfix		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
File: line	UrlValidator.java : 393		

Description

In UrlValidatorTest.java, this is the code snippet that generates the error.
URLValidator/src/UrlValidatorTest.java

```
UrlValidator urlVal = new UrlValidator(null, null, UrlValidator.ALLOW_ALL_SCHEMES);
      System.out.println("Testing sixth partition:\n");

      String url = "www.google.com";
      boolean valid = urlVal.isValid(url);
      if (valid)
      {
         System.out.println(url + " passed with valid authority and no optional components.\n");
      }
      else
```

```
      {
          System.out.println("ERROR: " + url + " failed with valid authority and no optional
components.\n");
      }
```

The code should return as passing, but returns as failing.
NOTE: This coincides with the bug about scheme being null.

isValidAuthority passing with invalid authority

Created: 13/Mar/16 Updated: 13/Mar/16

Status:	Open
Project:	Commons Validator CS 362 Final Project

Type:	Bug	**Priority:**	Major
Reporter:	Andy Garcia, Kendra Swafford, Anna Murphy	**Assignee:**	Unassigned
Resolution:	Unresolved	**Votes:**	0
Labels:	easyfix		
Remaining Estimate:	Not Specified		
Time Spent:	Not Specified		
Original Estimate:	Not Specified		
File: line	UrlValidator.java : 403 UrlValidatorTest.java : 406		

Description

In UrlValidatorTest.java, this is the code snippet that generates the error.
URLValidator/src/UrlValidatorTest.java

```
String invalidAuthority = "256.256.256.256";
      if (urlVal.isValidAuthority(invalidAuthority)) {
          System.out.println("ERROR: " + invalidAuthority + " passed with invalid authority.\n");
      }
```

The code should return as failing with an invalid authority, but instead passes.
The code is not checking for this type of authority. The maximum allowed should be 255.255.255.255