CS 362
Final Project
Ryan Kitchen

Methodology:

Manual Testing:

Created a simple loop that allows manual entry of URLs to be tested. These urls were tested on both the valid and buggy versions of URL validator. URL formats to be tested were determined by the protocol sections listed in the URI RFC.

URL	Valid	Buggy	Comments
http://www.google.com	pass	pass	
http://www.google.co.uk	pass	fail	check for international support
http://www.test.tech	fail	fail	Not updated to include new TLDs
www.google.com	fail	fail	Must include scheme
http://www.google.com?query=test	pass	fail	check reserved characters
http://www.google.com/?query=test	pass	fail	
http://localhost	fail	fail	Does not accept domains without TLD
http://10.0.0.1	pass	pass	
http://10.0.0	fail	fail	should fail, incomplete IP
HTTP://google.com	fail	fail	Should pass, RFC article 3.1 specifies scheme must not be case sensitive
http://google.com:80	pass	pass	
http://user@google.com	fail	fail	should pass, RFC includes user specification for URLs
http://www.google.com.	fail	fail	should pass, extra period specifies external TLD per RFC 3.3

Test Partitioning:

In order to fully test the isValid method, several different sections will be made to test different parts of the URL. These sections will be based on the URL components specified in the RFC:

(scheme) :// (authority)(path)?(query)

Scheme:

Scheme can be http/ftp/https and others, will test with http and ftp. RFC specifies scheme is not case sensitive.

Authority:

For URLs authority is the domain name, optionally with the port and user specified. Valid domain names can include alphanumeric characters, and dashes are allowed only in the middle of a name component (not on the ends). Subdomains can be separated by dots. A final terminating dot is allowed. The top level (rightmost) domain must be alphabet characters only.

Path:

HTTP requests specify the start of the path as the first slash following the host name. Path can include unreserved characters (listed in the RFC) as well as escaped reserved characters, and each segment may contain additional parameters separated by semicolon. Segments are separated by slashes

?Query:

Query can include a number of different operators specified by the interpreter. Includes reserved characters.

Test Cases:

Scheme:

Manual testing with different scheme specifiers. Seems to work ok

Authority:

Unit test: Address with all allowable characters for the authority portion

Unit test: Address with port number

Unit test: Address with user info specified

Unit test: Address with each invalid character in the authority section. All should return invalid

Path:

Unit test: Path with all allowable characters

Unit test: Path with multiple levels

Unit test: Path with disallowed characters, should fail

Unit test: Path with unescaped reserved characters, should fail

?Query:

Unit test: Query with multiple reserved query symbols

Random Testing:

Random tests performed will generate valid and invalid variants of the URL, and check them against isValid. The invalid URLs will be generated by randomly inserting one reserved/illegal character into the host field of the URL. This will enable detection of specific characters that might slip past the validator.

Results:

Scheme:

No bugs found. Tested http, https, ftp.

Authority:

Bug report 01:

Bug: URL Validator does not accept international TLDs

Method: Found during manual testing. Failed with inputs:

http://www.google.co.uk

http://www.google.ru

Tested against valid version. Valid version accepts these URLs

Cause: In the DomainValidator class, the second half of the country codes is missing in the country TLD array

Bug report 02:

Bug: URL Validator does not accept port numbers above 3 digits long

Method: Found with unit tests. Failed with input:

http://www.google.com:1234

Followed up with manual testing. Passes with input:

http://www.google.com:123

Cause: In URLValidator class, the regex for port numbers only allows between 1 and 3 digits, should be between 1 and 5

Bug report 03:

Bug: URL Validator does not accept user specification

Method: Found in unit testing. Failed with input:

http://user@test.com

This is a bug as the RFC specifies a user data option for URLs

Cause: isValidAuthority in URLValidator contains no method to check for a user data specification

Bug report 04:

Bug: URL Validator does not accept queries

Method: Found in manual testing. Failed with inputs:

http://www.google.com?query=test

http://www.google.com/?query=test

Tested against valid version, valid accepts both

Cause: Stray ! operator in isValidQuery, will always return false if there is a query as the regex for isValidQuery will always return true

Bug report 05:

Bug: URL Validator allows unescaped reserved characters

Method: Found with unit test. Passed with input:

http://www.google.com/;

Should have failed, ; is a reserved character that requires escaping, and is only allowed unescaped in the query section

Cause: Path regex allows for ; and ?, both of which are reserved and not allowable for path without being escaped.

Bug report 06:

Bug: URL Validator does not accept capitalized scheme

Method: Found with manual testing. Failed with input:

HTTP://www.google.com

Should have passed, RFC specifies that scheme is not case sensitive.

Cause: Default schemes include http,https,and ftp. isValidScheme does not convert input to lowercase before checking against the valid scheme list.

Bug report 07:

Bug: Authority validator will accept # as a viable character.

Method: Found in random testing. Passed with inputs:

http://AZFjAfLaTBbSzwJpIlkJG-LP7SUmtVs.com#053/IIk+/v!

http://wkTSO-t-Gg.com#756/O/+

http://PIvCOW0jz3646FIP-Z9wHt.co#:6/h_/IM/=A/6@

http://6D-5QtHPf4T.co#/N/mi

http://yoMw-FAKfc1cahmwctim.com#241/@7vu0

http://wb8k.co#/3h+/WZs/c

http://YxoMYr-Vk-iH.com:60#/)zm/'xTC/f

http://LJ-dpOG-dd.com:8#1/Y/9/I=

http://J.com#41/3/94xw/)

Cause: Authority regex seems to allow for any characters, when the RFC specifies only alphanumeric, - , and :portno.

Extra Credit:

I wrote a random tester for URL Validator. The tester generates URLS by creating the individual components and then combining them together.

The first phase is testing valid URLs. The URLs are randomly generated from only the valid character set for each portion.

The second phase is testing invalid URLs. Each URL will have exactly one error that should be caught by the isValid method.

Bugs caught:

First phase testing only caught the port bug. Used 1000 test cases. Altered the testing to exclude port bugs- caught no errors. The test data did not include queries because queries are not accepted by the buggy URLValidator

Second phase caught a new bug in the authority validator- authority validator will accept characters not allowed by the RFC. Used 1000 test cases. Invalid cases were created by inserting one illegal character randomly into the host field- since the path field can have almost any character, and the query part does not work at all, inserting characters into other fields will not provide useful results.