Isfandyar Rabbani, 932960451
CS362
Final Project – Part B

Methodology Testing

Manual Testing

I first tested the URL validator manually using several legal and illegal inputs. The logic I used for testing was to use various permutations of the same website to try to narrow down how to break the program. The function that I made for manual testing with command line input arguments looked like this:

```
public static void manualTesting(String[] args)
    {
        UrlValidator validator = new UrlValidator();
        System.out.println(args[0] + " valid?: " + validator.isValid(args[0]));
    }
```

Some results with the correctly working validator:

https://www.google.com – true
http://www.google.com – true
www.google.com – false
google.com – false
192.168.1.1 – false
1234 – fail
^%*&#$&^# - fail
www.com – fail

Here I found that at least one of the functions in the program is simply testing the format of the input and failing if it does not match that format. By my preliminary results, I can already see that the URL requires a format with some kind of :// prefix.

1234://www.google.com - false
http://123.google.com – true
httpss://www.google.com - false
https:///www.google.com - false

Now I have found that the program requires http:// or https:// as prefix to the website before validating it. So now I will try to break it with even including this prefix.

https://google.com – true
https://www..com – false
https://www.a.com – true
https://www.z.com – true
https://www./.com – false
https://192.168.1.1 – true
http://www.1.army - true
http://www.thelongestdomainnameintheworldandthensomeandthensomemoreandmore.com/ - true
https://oregonstate.instructure.com/courses/1683590/assignments/7302434#submit – true
http://www.cplusplus.com/files/tutorial.pdf - true

Input Partitioning

Now that I have an idea of what the expected results of the URL validator will be, I have to find a way to efficiently sift through inputs to test the buggy version with. In order to come up with a meaningful way to partition the inputs, I thought about how to best divide the domain so that you can test whether the program is able to function as expected. My brainstorming left me to divide the possible inputs into two different groups. One group is for URL inputs that the working validator deemed valid URLs. The second group of inputs will be URLs that the working validator deemed invalid URLs. Given the nature of URL makeup, this is the most logical way to divide the domain as URLS can be complex permutations or extremely short.

The following is the breakdown of the two input divisions I created:

First subdivision – 'Valid' URLs:
https://www.google.com
http://www.google.com
http://123.google.com
http://www.thelongestdomainnameintheworldandthensomeandthensomemoreandmore.com/
https://www.com
https://http.com
https://www.1.com
https://1.com
https://1.ly
https://1.xxx

https://oregonstate.instructure.com/courses/1683590/assignments/7302434#submit
http://www.cplusplus.com/files/tutorial.pdf
http://www.1.com
http://www.1.ba
http://www.1.bb
http://www.1.army
http://www.1.adult
http://www.1.wien
http://www.1.wtf
http://www.1.yokohama
http://www.1.wedding
https://www.bing.com/news/search?q=news+for+you&FORM=Z9LH3
https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/aa752574(v=vs.85)
https://www.blah.com/some/crazy/path.html;param1=foo;param2=bar

Second subdivision – 'Invalid' URLs:
https://1
https://1.1
http://www.1.bc
http://www.1.a
http://www.1.b
https:///www.google.com
httpss://www.google.com
192.168.1.1
1234
^%*&#$&^#
www.com
1234://www.google.com

Programming-Based Testing

Test Cases

```
import java.security.SecureRandom;
import java.util.Base64;
import java.util.Base64.Encoder;

public class FinalProjectPartB {

    // uses command line input for manual testing
    public static void manualTesting(String[] args)
    {
        UrlValidator validator = new UrlValidator();
        System.out.println(args[0] + " valid?: " + validator.isValid(args[0]));
    }

    // unit tests for valid url input based on the correct version
    public static void unitTestsForValidUrls()
    {
        // instantiate incorrect validator object
        UrlValidator validator = new UrlValidator();
```

```
        // create array of valid urls
        String[] testValidUrls = {
            "https://www.google.com",
            "http://www.google.com",
            "http://123.google.com",
            "https://192.168.1.1",
            "https://www.com",
            "https://http.com",
            "https://www.1.com",
            "https://1.com",
            "https://1.ly",
            "https://1.xxx",

"https://oregonstate.instructure.com/courses/1683590/assignments/7302434#submit",
            "http://www.cplusplus.com/files/tutorial.pdf",
            "http://www.1.ba",
            "http://www.1.bb",
            "http://www.1.army",
            "http://www.1.adult",
            "http://www.1.wien",
            "http://www.1.wtf",
            "http://www.1.yokohama",
            "http://www.1.wedding",

"http://www.thelongestdomainnameintheworldandthensomeandthensomemoreandmore.com/",

"http://iamtheproudowneroftheLongestlongestlongestdomainnameinthisworld.com/",
            "https://www.blah.com"
        };

        // assert valid urls
        for (int i = 0; i < testValidUrls.length; i++)
        {
            assert(validator.isValid(testValidUrls[i]) == true);
        }

        System.out.println("Passed all valid URL test.");
    }

    // unit tests for invalid url input based on the correct version
    public static void unitTestsForInvalidUrls()
    {
        // instantiate incorrect validator object
        UrlValidator validator = new UrlValidator();

        // create array of invalid urls
        String[] testInvalidUrls = {

"//http://iamtheproudowneroftheLongestlongestlongestdomainnameinthisworld.com/",
            "https://1",
            "https://1.1",
            "http://www.1.a",
            "http://www.1.b",
            "http://www.1.bc",
            "httpss://www.google.com",
            "https:///www.google.com",
            "https://www..com",
```

```
            "https://www./.com"
        };

        // assert invalid urls
        for (int i = 0; i < testInvalidUrls.length; i++)
        {
            assert(validator.isValid(testInvalidUrls[i]) == false);
        }

        System.out.println("Passed all invalid URL test.");
    }

    // main method, runs the unit tests
    public static void main(String[] args)
    {
        manualTesting(args);
        unitTestsForValidUrls();
        unitTestsForInvalidUrls();
    }
}
```

Bug Report

Bug #1 – Showstopper bug – All valid and invalid URL's alike are returning as invalid URLs.

Every single URL that I asserted should be either invalid or valid in my test design is returning invalid from the program. In fact, I could not get a single URL at all to return true from the URL validator. This was detected by all of my tests, as all literally every single one of my tests for valid URL validation failed with dying colors.

After debugging using Eclipse, I found that the malfunctioning code is the third function call in the validator code called isValidScheme. Something about the regular expression or the way that it is handled in it's dependencies is causing the program to function erratically.

Bug #2 – '?' character for queries in URL causes 'No match' error. The functionality of validating URL's with queries in it is specified in the source code and documentation.

The source of this error appears to be from the passing of the regular expression for the isValid method. Something about the syntax does not allow '?' in the input URL anywhere at all, even where it is legal.

Bug #3 – Legal and valid parentheses causes 'Badly placed ()'s' error.

The source of this error appears to be from the passing of the regular expression for the isValid method. Something about the syntax does not allow for parentheses to be included in the characters after the final slash.

Bug #4 – Legal and valid semicolon in URL causes URL path shortening.

The source of this error appears to be from the passing of the regular expression for the isValid method. Something about the syntax does not allow ';' in the input URL anywhere at all, even where it is legal.

Debugging

Bug #1 - *After debugging using Eclipse, I found that the malfunctioning code is the third function call in the validator code called isValidScheme. Something about the regular expression or the way that it is handled in it's dependencies is causing the program to function erratically.*

Bug #2 - *The source of this error appears to be from the passing of the regular expression for the isValid method. Something about the syntax does not allow '?' in the input URL anywhere at all, even where it is legal.*

```
Input: https://www.bing.com/news/search?q=news+for+you&FORM=Z9LH3
[1] 1455
java: No match.
FORM=Z9LH3: Command not found.
[1]  + Exit 1                  java FinalProjectPartB https://www.bing.com/news/search?q=news+for+you

Input: https://www.bing.com/news/search?q=news+for+you
java: No match.

Input: https://www.bing.com/news/search?
```

java: No match.

Input: https://www.bing.com/news/search?q=news
java: No match.

Input: https://www.bing.com/news/search?q=
java: No match.

Input: https://www.bing.com/news/search?q
java: No match.

Input: https://www.bing.com/?
java: No match.

Bug #3: *The source of this error appears to be from the passing of the regular expression for the isValid method. Something about the syntax does not allow for parentheses to be included in the characters after the final slash.*

Input: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/aa752574(v=vs.85)
Badly placed ()'s.

Input: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/aa752574(v=vs.85)
Badly placed ()'s.

Input: https://docs.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/platform-apis/aa752574(v=vs.85)
Badly placed ()'s.

Bug #4: *The source of this error appears to be from the passing of the regular expression for the isValid method. Something about the syntax does not allow ';' in the input URL anywhere at all, even where it is legal.*

Input: https://www.blah.com/some/crazy/path.html;param1=foo;param2=bar
https://www.blah.com/some/crazy/path.html valid?: true
param1=foo: Command not found.
param2=bar: Command not found.

Team Work

For me, this was the easiest part of the project. My work schedule does not coincide with anyone elses. Therefore, I did the project by myself, so I did not have to deal with the tapdance of coordinating with people's schedules.

This turned out to be a blessing, as I experienced every aspect of the manual testing, test design, and debugging on my own rather than sharing the responsibility with others. I feel that I received a very comprehensive lesson in the entire process.