Tony Liang

Khiem Nguyen

12/6/15

CS 362 Project part B

*** Some of Khiem’s unit tests are random test foe extra credit consideration, they are: portRandomTest(); and ipv4RandomTest()

* Methodology *

For manual testing we used urls that we knew would be valid or invalid. For example, we used the default "http:://www.amazon.com" as one of the valid and "://www.amazon.com" as one of the invalid urls. We also tested IP addresses.

We partitioned the input very similarly to the official Apache test code. That is, we split the input domain to the URL scheme, authority, port, and query. We felt this was a logical and straightforward way to partition because we know that if all the other inputs were valid, we could just test a particular part of the URL that would induce a failed test case. We also know that the code base used for UrlValidator.java splits validation into such categories and it would be logical to align with that existing partitioning.

To find the bugs, we used Agan's rules. Though some of them are obvious, we used Rule #5, "Change One Thing at a Time" and Rule #6, "Keep an Audit Trail" frequently to help our bug-finding.

The result of our tests can be found in console_output.txt.

* Team Coordination *

Originally we were a group of three members. Then, apparently, James
dropped from the course.

Khiem provided the initial framework, manual tests, input partition and unit tests. He also found several bugs as a result of his testing.

Tony took Khiem's work to refine and edit. He also did a lot of code
cleanups and consolidation.

Tony wrote the report and did some more testing. He made sure the bug reports
are in accordance with the official Apache format.

* Bugs Found *

Title: Port numbers higher than 999 incorrectly invalidates URL
Type: Bug
Priority: Major
Affects Version: 1.4
Reporter: Khiem Nguyen
Description:

When a port number in the URL is at least 1000,
UrlValidator.isValid() incorrectly returns false
(unless it is 65536 or above).

For example, "http://www.google.com:1234" returns false
from isValid().

Using our testPortRandom() test in UrlValidatorTest.java, we were able to discern a clear pattern: port numbers with 4 or 5 digits invalidated. We subsequently confirmed it in testPort() by testing port 999 versus 1000 and other 4 or 5-digit numbers and found that the latter numbers failed.

Through code inspection, the offending code was found in line 158 in UrlValidator.java.

The PORT_REGEX expression is set to look at only three digits instead of proper 5 digits (since 65535 is the highest port number).

Incorrect code:

```
private static final String PORT_REGEX = "^:(\\d{1,3})$";
```

Correct code:

```
private static final String PORT_REGEX = "^:(\\d{1,5})$";
```

Title: Valid GET query strings incorrectly invalidates URL
Type: Bug
Priority: Major
Affects Version: 1.4
Reporter: Khiem Nguyen
Description:

As simple GET query string such as "?action=view" appended to an otherwise valid URL invalidates it in UrlValidator.isValid().

For example, a perfectly valid "http://www.google.com?action=view" is returns false from isValid().

Our testQuery() method in UrlValidatorText.java clearly shows this bug in the console.

Through code inspection, we looked at isValidQuery() in UrlValidator.java and found the bug in line 446. There is a negating "!" where there should be none. The line should return whether the query matches a regex expression defining what a correct query is. Instead, because of the "!", it returns the opposite result.

Incorrect code:

```
return !QUERY_PATTERN.matcher(query).matches();
```

Correct code:

```
return QUERY_PATTERN.matcher(query).matches();
```

Title: IP address segments greater than 255 can incorrectly be validated
Type: Bug

Priority: Normal

Affects Version: 1.4

Reporter: Khiem Nguyen

Description:

If an IP address contains one or more segments of 256 or above,
that incorrectly yields a validation from UrlValidator.isValid().

For, example, "1.1.256.1" incorrectly validates.

Our testManualTest() method in UrlValidatorTest.java, as part of general testing,
caught this error.

Through code inspection, we looked at InetAddressValidator.java and found the bug
in line 95. The code allows IP address segments of 256 or greater to be validated,
which causes isValidInet4Address() to incorrectly return true if such a segment
where in the IP address.

Note that the code only applies to IPv4 addresses.

Incorrect code:

```
if (iIpSegment > 255) {
        return true;}
```

Correct code:

```
if (iIpSegment > 255) {
```

```
        return false;}
```

Title: County Code TLDs alphabetically after "it" invalidates URL

Type: Bug

Priority: Major

Affects Version: 1.4

Reporter: Tony Liang

Description:

If a URL's authority were a hostname/domain ending with a country code top-level domain alphabetically greater than "it" for Italy, the URL is incorrectly invalidated in UrlValidator.isValid().

For example, "http://www.bbc.co.uk" and "http://www.parliament.uk" are incorrectly invalid.

Our testAuthority() method in UrlValidatorTest.java caught an instance of this error with regards to .uk.

To find the bug, we first inserted the following assert statement:

```
assertTrue("www.parliament.uk should validate",
        urlVal.isValid("http://www.parliament.uk"));
```

...at the end of testAuthority(). This predictably failed but did not produce any useful information, even from the trace.

So we used code inspection yet again. UrlValidator references the DomainValidator class to validate non-IP address authorities. So we looked at that, specifically the isValid() function in the .java file.

Eventually, with regards to country-code based TLDs, it refers to an array containing all country codes called COUNTRY_CODE_TLDS. It incorrectly stops at Italy. Specifically, the error is in line 358 of DomainValidator.java.

Incorrect code:

```
        private static final String[] COUNTRY_CODE_TLDS = new String[] {
    "ac",               // Ascension Island
    "ad",               // Andorra
    ...
    "is",               // Iceland
    "it",               // Italy

 };
```

Correct code:

```
        private static final String[] COUNTRY_CODE_TLDS = new String[] {
    "ac",               // Ascension Island
    "ad",               // Andorra
    ...
    "zm",               // Zambia
    "zw",               // Zimbabwe
```

```
};
```

The "..." in the above code is an alphabetical continuation of the

country codes up until the ending entries in the array.

Source: https://en.wikipedia.org/wiki/List_of_Internet_top-level_domains#Country_code_top-level_domains