CS 362 Software Engineering II

Final Project

Ian Kronquist

1 Finding Bugs

1. Manual Testing. For manual testing I tried a variety of URIs which used different protocols than http and https. I copied and built upon preexisting code in the URLValidatorTest file. I did not find any bugs initially.

```
UrlValidator urlValidator = new UrlValidator();
assertTrue("osuosl.org should validate",
        validator.isValid("http://osuosl.org/test/index.html?foo=bar"));
assertFalse("Git URIs are not URLs",
        validator.isValid("git://osuosl.org/test/index.html?foo=bar"));
assertFalse("Gopher URIs are not URLS",
        validator.isValid("gopher://osuosl.org/test/index.html?foo=bar"));
assertFalse("IRC URIs are not URLs",
        validator.isValid("irc://chat.freenode.net"));
assertFalse("osuosl.org should still validate",
        validator.isValid("irc://chat.freenode.net"));
assertTrue("Freenode's URL is valid",
        validator.isValid("http://chat.freenode.net"));
assertFalse("Crazy nonsense should not be valid",
        validator.isValid("http://chat--.freenode.net"));
assertFalse("URLs with emoji should not be valid",
        validator.isValid("http://."));
assertTrue("test.invalidtld should validate because tld's aren't validated",
        validator.isValid("http://test.invalidtld/"));
```

2. Input partitioning. By following Agan's first rule of debugging and inspecting and understanding the source code I determined that the tests partition the input into several categories: URL scheme, authority, port, path, and query. The default schemes are "http", "https", and "ftp". The schemes, authorities, ports, paths, and queries are all validated by different regexes.

3. I wrote a simple unit test, contained in the file
`/projects/kronquii/URLValidator/URLValidator/test/MyURLTest.java`,
which combined various valid and invalid parts of input together. I was able to generate 237 test failures pretty quickly. By following Agan's rules of debugging and inspecting the URLs which failed, I was able to determine that the bug likely lay in the query section. Something also seemed off about the port section as well. I decided to start by focusing on the queries. After I determined there was a bug in the `isValidQuery` method, I realized that the `PORT_REGEX` was incorrect.

4. Teamwork: I reached out to my team, Jesse Poor and James Guerra, several times via their onid email accounts and they never replied. I tried to be accommodating with my schedule and offered them a broad swath of my time to meet. I hope I am not marked down because they were not communicative. It is frustrating to do an entire group project yourself. I have not heard from James at all, and I have not heard from Jesse since the 8th of November.

Here are a couple of the most recent emails I sent them:

(a) Dec 1, 2015
Software Engineering II Final Project
Hello,
Are you guys interested in working together on the final project? Do you want to meet sometime to work on it? I am free most mornings before noon and most evenings after 6:00pm. I am in Corvallis on Pacific Time.

Sincerely,
Ian Kronquist

(b) Dec 5, 2015
Re: Software Engineering II Final Project
Hey guys,
The final project expects us to work together. Are you interested in working together at all? If so, please let me know. I am available pretty much all day Sunday and in the evening this Saturday.

Sincerely,
Ian Kronquist

2 Bug Reports

1. URL query parameters are incorrectly marked as invalid or valid
Type: Bug
Date Created: 12/05/2015
Status: Closed
Priority: Normal
Components: URLValidator
Reporter: Ian Kronquist

Found by: Automatic testing using input partitioning. I investigated the failure by setting breakpoints in the `isValidQuery` method and stepping through it.
Cause of failure: The return value of the `isValidQuery` method was inverted.
Minimal test case:

```
UrlValidator urlValidator = new UrlValidator();
assertTrue("Should be a valid URL to search google",
        validator.isValid("https://www.google.com/search?q=test"));
assertTrue("Should not be a valid URL",
```

```
        validator.isValid("https://www.google.com/search/q=noquestionmark"));
```

2. URLs with ports greater than 999 are marked as invalid
   Type: Bug
   Date Created: 12/05/2015
   Status: Closed
   Priority: Normal
   Components: URLValidator
   Reporter: Ian Kronquist

   Found by: Automatic testing using input partitioning. I investigated the failure by setting breakpoints in the `isValidAuthority` method (which matches on the `PORT_PATTERN` regex) and stepping through it. To confirm the exact nature of the bug I tested a variety of port values manually as well.
   Cause of failure: The value of the `PORT_PATTERN` regex is incorrect.
   Minimal test case:

```
UrlValidator urlValidator = new UrlValidator();
assertTrue("2^16-1 should be a valid port",
        validator.isValid("https://example.com:65535"));

assertTrue("1000 should be a valid port",
        validator.isValid("https://example.com:1000"));
```

3. Once #3 is fixed, 4 digit ports greater than 65535 are marked as correct
   Type: Bug
   Date Created: 12/05/2015
   Status: Closed
   Priority: Normal
   Components: URLValidator
   Reporter: Ian Kronquist

   Found by: Playing a hunch after fixing #3.
   Cause of failure: The value of the port is only validated with the `PORT_PATTERN` regex. The port value is not checked.
   Minimal test case:

```
UrlValidator urlValidator = new UrlValidator();
assertTrue("2^16-1 should be a valid port",
        validator.isValid("https://example.com:65536"));

assertTrue("1000 should be a valid port",
        validator.isValid("https://example.com:99999"));
```