CS 362 Software Engineering II
Winter 2016
Matt Boal
Gerald Gale
Scott Williams

Manual Testing:

Our manual testing was similar to our implementation of partitions. We wanted to test each part of a URL with slightly varying differences to test whether or not the UrlValidator was able to effectively validate each section. I chose to use google as the control for as many tests as I could.

1. Protocol:
    a. http://www.google.com
    b. https://www.google.com
    c. ftp://www.google.com
    d. htt://www.google.com
    e. http1://www.google.com
    f. http:/www.google.com
2. Hosts:
    a. http://www.google.com
    b. http://www.yahoo.com
    c. http://www.cnn.com
    d. http://www.12345.com
    e. http://www.two_words.com
    f. http://www.one-hyphen.com
    g. http://www.ask?.com
3. Domains:
    a. http://www.google.com
    b. http://www.google.net
    c. http://www.google.edu
    d. http://www.google.gov
    e. http://www.google.io
    f. http://www.google.111
    g. http://www.google.co.uk
    h. http://www.google.coms
4. Ports:
    a. http://www.google.com:80
    b. http://www.google.com:7000
    c. http://www.google.com:5555
    d. http://www.google.com:-1

    e. http://www.google.com:100a
5. Queries:
    a. http://www.google.com?key=this
    b. http://www.google.com?twenty=20&UnitID=439
6. Paths:
    a. http://www.google.com/mail
    b. http://www.google.com/$2
    c. http://www.google.com/

Partitioning:

We decided to partition the inputs for the URL validator into five categories. These are schemes, authorities, ports, paths, and queries. We show the inputs we used for partitioning and an explanation beneath each one for why we chose the inputs we did.

```
private ResultPair[] testUrlSchemes = {
    new ResultPair("", true),
    new ResultPair("http://", true),
    new ResultPair("https://", true),
    new ResultPair("ftp://", true),
    new ResultPair("htt://", false),
    new ResultPair("http:", false),
    new ResultPair("http:/", false),
    new ResultPair("https/", false),
    new ResultPair("///", false)
    };
```

For Schemes, we probably included more options than is strictly necessary to fully isolate different possible inputs. We wanted to first include the most common schemes (http, ftp, and http), and an empty scheme. After that, we misformed the http scheme in a few ways. We figured it was unnecessary to misform ftp and https, since this would probably cover those possible situations. Here we found that it returned as false when the scheme was missing and returned true when it was set to “htt://”. Otherwise, It worked.

```
private ResultPair[] testAuthorities = {
      new ResultPair("www.google.com", true),
      new ResultPair("amazon.com", true),
      new ResultPair("uk.yahoo.com", true),
      new ResultPair("www.yahoo.co.uk", true),
      new ResultPair("yahoo.co.uk", true),
      new ResultPair("0.0.0.0", true),
      new ResultPair("192.168.0.1", true),
      new ResultPair("192.174.456.1", false),
      new ResultPair("223.com", true),
      new ResultPair("www.google.ninja", true),
```

```
        new ResultPair("www.google.net", true),
        new ResultPair("www.google.gov", true),
        new ResultPair("www.google.edu", true),
        new ResultPair("www.google.io", true),
        new ResultPair("docs.google.com", true),
        new ResultPair("www.google.444", false),
        new ResultPair("192.145.3.2.", false),
        new ResultPair("192.168.1", false),
        new ResultPair(".4.3.4.5", false),
        new ResultPair("", false)};
```

There is quite a bit of partitioning required for the authority part of the URL. This is because it has 3 somewhat independent parts, and it can also be an IP address. We started by removing or malforming the first third and the TLD(last third). There is no reason to do that with the middle, since it can be almost anything and be valid. We also wanted to test various TLDs, since there are many available now, but there are still values that can be incorrect. Lastly, we tested a few IP adddresses. Some malformed and some not. In our testing we found that it doesn't have the newest TLD's available, or at least it doesn't have ".ninja". Also, it returned an invalid IP address "192.174.456.1" as a valid possible Authority. Also, it didn't recoginze the TLD ".co.uk". Otherwise, it did a pretty good job.

```
 private ResultPair[] testPorts = {
        new ResultPair(":80", true),
        new ResultPair(":0", true),
        new ResultPair("", true),
        new ResultPair(":-1", false),
        new ResultPair(":65636", true),
        new ResultPair(":65a", false)
        };
```

In testing the Ports we figured the most important inputs to test were, a valid port, a missing port, an invalid port, and an unusual port. We found that the only one that cause issues was having an unusual port ("65636"). As far as we know, this should return as a valid URL and still work.

```
 private ResultPair[] testPaths = {
        new ResultPair("/mail/u/2", true),
        new ResultPair("/mail", true),
        new ResultPair("/$37", true),
        new ResultPair("/mail/", true),
        new ResultPair("/..", false),
        new ResultPair("/../", false),
        new ResultPair("", true),
        new ResultPair("/", true),
        new ResultPair("/..//mail", false),
        new ResultPair("/inbox//mail", false)
        };
```

For the path input partitioning we wanted to test having it missing or just an empty slash, one value, multiple values, and malforming it in the way you might want to do if attempting to traverse a linux file

system, or just straight up incorrect malformed values. All of the examples we used above passed validation.

```
private ResultPair[] testQueries = {
      new ResultPair("", true),
      new ResultPair("?twenty=20&UnitID=439", true),
      new ResultPair("?stuff=true", true)
      };
```

Queries was the simplest to partition. As far as we know, there are really only three main partitions. An empty query, a query with one value and a query with multiple values. The query logic is very broken, and seems to return false every time it's anything but empty.

Unit Tests:

All of our test functions are located in the file UrlValidatorTest.java:

public void loopOverSchemes()
public void loopOverAuthorities(string testURL, Boolean expectedResult)
public void loopOverPorts(string testURL, Boolean expectedResult)
public void loopOverPaths(string testURL, Boolean expectedResult)
public void loopOverQueries(string testURL, Boolean expectedResult)
public void validate(string testURL, Boolean expectedResult)
public void printResults(string testURL, Boolean expectedResult, Boolean result)

Working as a group:

We had no issues working as a group together. We were able to efficiently communicate via email and Google Hangouts. Each member was conscientious of each other's schedule and workload and were able to accommodate accordingly. We decided to split up the work as evenly as possible based on the per-point value assigned in the final project.

- Scott Williams did the manual testing and wrote most of the report and bug reports.
- Matt Boal wrote all of the unit tests and found the initial bugs. He also came up with our input partitioning and wrote that section of the report.
- Gerald Gale performed all of the debugging and found the locations of each bug.

Overall, we had no issues working as a group and were able to work together easily and efficiently.

Bugs found:

1. **URL Query is returning false when it should be true. Example URL inputs:**
   http://www.google.com?twenty=20&UnitID=439
   http://www.google.com?stuff=true
   Both returned false from the isValid() method at line 314 in UrlValidator.java:
   if (!isValidQuery(urlMatcher.group(Parse_URL_QUERY))) {

   A minor bug that is reproducible every time. We cannot tell why this bug is happening because the logic is imported in "java.util.regex.Matcher".

2. **A missing or "NULL" top-level domain returns false when it should be true. Example URL input:**
   www.google.com
   Returns false from the isValid() method at line 361 in UrlValidator.java:
   If (authority == null) {
   Return false;
   }

   A minor bug that is reproducible every time. Very simple fix would be to change to "return true;" as a URL without a top-level-domain is still a valid URL.

3. **Some valid top-level-domains are not found. Example URL input:**
   http://www.yahoo.co.uk
   http://www.google.ninja
   Returns false from the isValid() method at line 385 in UrlValidator.java:
   If (!inetAddressValidator.isValid(hostLocation)) {

   This is a minor bug that is reproducible every time. More than likely, foreign URLs are not in the imported java library and should be added as well as some newer TLDs.

4. **Invalid schemes are not returning false. Example URL inputs:**
   htt://www.google.com
   htt://www.yahoo.com
   These schemes should be returning false at line 340 in UrlValidator.java:
   If (!SCHEME_PATTERN.matcher(scheme).matches()) {

   This is a minor bug that is reproducible every time.