Drake Seifert seifertd

Corwin Perren perrenc

Nick McComb mccombn <- Submitted under this account for team

# CS362 Final Project Part A

**Explain the testIsValid() Function of the UrlValidator test code.**

The original testIsValid() function begins by calling the function testIsValid(Object[] testObjects, Long options). This testIsValid() function takes a testURL array as a parameter and runs through all possible permutations of its combinations. It begins by creating a new UrlValidator object to begin testing on, and proceeds to use assert statements on various URLs, making sure proper URLs return true, and improper URLs return false (the URLs being tested can be found at the bottom of the UrlValidatorTest object). It continues by creating a StringBuffer object, and running through an array of tests. After setting a boolean variable equal to true, a bitwise AND operation is used against the boolean variable to verify the correct result of the URL. If the expected result of the URL occurs, a period will print out to the system menu, otherwise an X will be printed. Various status indicators are also printed along the way. This set of testing proceeds to jump through and test every testObject within the object list. Finally, the original testIsValid() function clears the testPartsIndex list by setting all elements to zero.

**How many URLs are being tested?**

The testIsValid() function tests all URLs in the following lists: testUrlScheme, testUrlAuthority, testUrlPort, testPath, and testUrlQuery. For each item in the list, the permutation of all combinations are taken through procedural generation.

A count in testIsValid determines that there are 1890 valid URLs that are being tested by default.

**Explain how the URLs are being built.**

The URLs are built using a combination of 4 different arrays that are included in the UrlValidatorTest.java. A url is built of arrays that represent the scheme, the authority, the path, and the query, where are combined procedurally using the following scheme:

```
<scheme>://<authority><path>?<query>
```

**Give an example of valid URL being tested and an invalid URL being tested by the testIsValid() method.**

The following code was added to the end of the testIsValid function to demonstrate the testing of a valid and invalid URL. Both asserts were not invoked.

This is an invalid URL that was tested by the testIsValid() method:

```
h3t://255.com:80/t123?action=edit&mode=up
```

This is a valid URL that was tested by the testIsValid() method:

```
http://www.google.com
```

**Do you think that a real world test (URL Validator's testIsValid() test in this case) is very different than the unit tests and card tests that we wrote (in terms of concepts & complexity)? Explain in few lines.**

In terms of complexity, this real world test is much more advanced. The function manages to take every combination of accurate and broken URLs, and covers an incredibly wide spectrum of possible URLs. However, seeing the source code for a thorough, real-world test definitely helps conceptualize how to create and utilize a successful test suite.