CS 362 Software Engineering II
Group 11
Hyunwook Shin - shinhyu
Marissa Kwon - kwonma
Zachary Williams - williaz3

Project Part A

1. Explain in detail the testIsValid() function from the UrlValidatorTest.java code in a section called **Unit Test Description.**
2. Describe in detail the total number of URLs that are tested and how it builds the URLs in a section called **Unit Test Data.**
3. Give 5 examples of 5 valid URLs and 5 invalid URLs being tested by the testIsValid() method in a section called **Unit Test Data Examples**.
4. The UrlValidator code is a direct copy of the apache commons URL validator code. The test file/code is also a direct copy of the apache commons test code. Explain how your unit tests that you wrote in Assignment 3 differ from a real-world test in URL Validator's testIsValid() and describe the differences in terms of concepts & complexity. Put this in a section called **Unit Test Techniques**.

- **Unit Test Description**
    - There is a testIsValid method that calls on a testIsValid function.The method that calls on the function will call on the testIsValid() function and sends testUrlParts and UrlValidator.ALLOW_ALL_SCHEMES as parameters. The testIsValid function that is called on accepts an array of objects called testObjects and also accepts a long integer value called options. The testUrlParts is an array with objects that are defined as testUrlScheme, testUrlAuthority, testUrlPort, testPath and testUrlQuery which is defined towards the end of the file. These array then creates a URL to be tested. These individual objects are combined in different permutations and based on the combination of the objects they either create a valid or invalid URL which is what the function is testing for.
    - The testIsValid(testObject[], long) method initializes a UrlValidator object to do the actual url checking. The code starts by testing two known versions of a valid url. Then a do/while loop builds and tests each combination of the testObjects fields. It rebuilds the urls and prints the result expected and the UrlValidator given result. There is a check to make sure that the expected result matches the UrlValidator result. If the flag to print is set, then the verdict ('.' or 'X') is printed to the serial monitor.

- **Unit Test Data**
    - The total number of URLs is 2520, which was calculated by taking all the true values of the ResultPair options and multiplying all of the parts that make up the URL. To explain further, the URLs are a permutation of the 5 objects mentioned in the Unit Test Description: testUrlScheme, testUrlAuthority, testUrlPort, testPath

and testUrlQuery. Each one of these objects produces a part of the url. Each of these parts has a corresponding ResultPair, which is listed towards the end of the file. By multiplying all of the true values of the ResultPair the total number of URLs that are testable are 2520.

- **Unit Test Data Examples**
  - Valid URL:
    - http://www.google.com
    - http://l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l.l .l.l.l.l.l.l.l.l.l.l.l.l.l.l.org
    - http://tech.yahoo.com/rc/desktops/102;_ylt=Ao8yevQHlZ4On0O3ZJGXLEQFLZA5
    - "http://somewhere.com/pathxyz/file(1).html
    - http://xn--d1abbgf6aiiy.xn--p1ai
  - Invalid URL:
    - http://broke.my-test/test/index.html
    - http://first.my-testing/test/index.html
    - http://broke.hostname/test/index.html
    - http://hostname
    - http://localhost/test/index.html

- **Unit Test Techniques**
  - The tests seen in the URL validator code or real world test have the same underlying principles as seen in the test written in assignment 3. When writing the unit tests and card test for assignment 3 a set of requirements were written so that the expected outcomes were written down before writing the tests. Similarly when the real world tests were written, a similar procedure was followed based on the examined code.
  - One difference that the group noticed between the real world tests and the test written in assignment 3 was the random generation of URLs which is the random generation of inputs indicative of random testing which the class did not cover until assignment 4. The 2520 urls generated represent coverage of all possible permutations for a subset of possible inputs that exist in the real world. The tester's methods of testing all possible url combinations generates unplanned sequences of valid and invalid urls, that can be interpreted as a form of random testing. There are probably more than just the 10 test paths provided that urls can be generated from, and these url inputs are not covered in our testing because they are not added to the testPath section. In this way all the possible inputs were tested, but it is not extensively covering all urls possible. Another difference between random generation for our assignments, and they way the urls are tested is that the way they are generated is the same each time the code is executed and running this test multiple times provides no new information..

- **Team Contribution**
    - Hyunwook Shin - Communication with group, Started Final Project Part A branch on repo, contribution to Unit Test Description, Unit Test Data and Unit Test Techniques.
    - Marissa Kwon - Communication with group, editing and contribution to Unit Test Description, Unit Test Techniques
    - Zachary Williams - Communication with group, URL Test Data Examples