CS362-004 Final Project Part A
May 5, 2015

Jessica Huang - huangjes
Steven Quagliata - quaglias
Jacqueline Francik - francikj

Review of existing test case(s) For Part-A, you will be provided a correctly working version of URL Validator.

1. Explain testIsValid Function of UrlValidator test code. (Available under URLValidatorCorrect folder)
    a. There are two parts for using “testIsValid” and they use more complex sections of the code to provide the desired result as follows:
        i. The function calls “testIsValid(testUrlParts, UrlValidator.ALLOW_ALL_SCHEMES)”
        This in turn has the following effects:
            1. Creates a new UrlValidator object
            2. Assert the validity of url “http://www.google.com”
            3. Assert the validity of url “http://www.google.com/”
            4. Runs a “do while” loop
                a. Loop creates a stringbuffer
                b. Merges url elements until one combined element is left
                c. Calls “isValid” with the new element
                d. Stores pass/fail value
                e. Uses “if/else” comparison between new element and expected outcome and display to terminal a pass/fail
        ii. The function calls “setup()”
            1. Resets the index to 0

2. Give how many total number of the URLs it is testing.
    - testUrlScheme = 9
    - testUrlAuthority = 19
    - testUrlPort = 7
    - testPath = 10
    - testUrlPathOptions = 15 → not used in testIsValid
    - testUrlQuery = 3

    9 * 19 * 7 * 10 * 3 =
    35,910 total number of URLs are being tested.

3. Explain how it is building all the URLs

- The 5 parts of the URL: scheme, authority, port, path, and query, are passed into the isTestValid function as an array of Objects. In a loop, url elements are merged until a complete url is created. The loop continues to create urls until it reaches the final index which is also the last url that can possibly be created.

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

    - Valid URL: "http://www.google.com:80/test1?action=view"
    - Invalid URL: "http/go.com:80/t123?action=edit&mode=up"

5. UrlValidator code is a direct copy paste of apache commons URL validator code. The test file/code is also direct copy paste of apache commons test code. Do you think that a real world test (URL Validator's testIsValid() test in this case) is very different than the unit tests that we wrote (in terms of concepts & complexity)? Explain in few lines.

The tests run under the dominion assignments follow a similar method of testing as the ones provided with the URL Validator's "testIsValid()" function. The premise behind the URL validator is to create objects (gamestate in dominion vs url in URL Validator) and run tests against those new iterations using a set of known input types and comparing that to the value of the actual new object.

6. Submit a file called ProjectPartA.pdf in Canvas with your write-up