# CS 362 Software Engineer 2
# Final Project
# Part A

**Aidan Grimshaw (grimshaa), Khuong Luu (luukh)**

**1. Explain testIsValid Function of UrlValidator test code. It is available under URLValidatorCorrect folder.**

The *testIsValid* function of UrlValidatorTest creates an instance of UrlValidator and starts by running basic validation tests, asserting http://www.google.com and http://www.google.com/ are valid domains, failing if not.

It then runs through a combination of url components from arrays at the bottom of the document. These are matched up in a mad libs fashion. The array indexes are also paired with a boolean value that indicates whether the value is valid or not. When matching up the component pieces of the url, the boolean values are also "added" where if 1 false value is added the expression becomes false.

When this process is complete, it returns an expected value that *urlVal.isValid()* should match. The function then asserts that the result of urlVal.isValid and the expected value match. If the results are valid, it prints a "." next to the url being tested. If not, it prints an "X" next to the url being tested, spacing these results out with a newline every *statusPerLine* amount.

**2. Give how many total number of the URLs it is testing. Also, explain how it is building all the URLs.**

It appears that the total number of URLs tested appears to be 1296 tests run by combining different URL components and 2 tests asserting http://www.google.com and http://www.google.com/ are valid domains for a total of 1268 URLS tested.

The combined urls are built by iterating through the matrix of *types of url parts* and *individual url parts of a type*. These are concatenated onto a master string that is run through the *urlVal.isValid()* function.

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

**3.1. Example of a valid URL being tested:**

```
https://go.com:80/test1?action=view
```

**3.2. Example of an <u>invalid</u> URL being tested:**

```
h3t://go.a:-1/../
```

**4. 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.**

To my observation, we do think the the real world test (URL Validator's testIsValid() test in this case) is similar to our unit test in our class in term of concept and somewhat different (more complex) in term of complexity.

**4.1. Similar in term of concepts**

The real-world URL Validator's testIsValid function (hereby we called "real-world test" for convenience) test isValid function unit by unit. In particular, since the isValid function has some smaller scope functions such as isValidScheme, isValidAuthority, isValidPath, etc. the testIsValid function mock those unit by separating parts of the test data. In those test data, the real-world test function, like our tests in class, also cover many different input cases, including edge cases.

The real-world test function also does integration test. It makes 2 assert statement at the beginning of the function to test if the target function is integrated correctly.

**4.2. Different In term of complexity**

The way they implement the concepts are more complex than our way.

In our class, we wrote different test function for different cases of test we would like to test, while in this real-world URLValidatorTest class, all the test data/test cases are put into a loop and the only test function will tested them one by one by iterating each one of them in an array.

There are also a whole function just for incrementing the index in the array that contains the test data: incrementTestPartsIndex; and this function is written masterful and very thorough with all cases considered.

Also, there is a whole bunch of code to control the output printing of the test class so that this test class and other test classes can print output consistently and systematically. Compared to our test class in this class, we don't do this kind of output control.

## 5. Contributions

### 5.1 Khuong Luu

Both of us investigated and together tried to explain how the testIsValid function works. Then, I wrote up answers for question 3 and question 4 (the 2 later questions). I also verify the number of test cases that Aidan got match my result. I also reviewed Aidan's part and he reviewed mine.

We communicate via Google Hangout and collaborate via Google Docs.

### 5.2. Aidan Grimshaw

We collaborated virtually using google hangouts voice chat to communicate and collaborate on the project. I wrote up the explanation of how the testIsValid function works and the explanation of the total number of URLs being tested and how the testing URLs are built. Khuoung and I checked over each other's work once we were both done with our individual sections.