November 21, 2017

Deliverable #5 Fault Injection

1. EXPERIENCES

A. The fault injections were fairly straightforward in that we had to make a small somewhat unnoticed change in the source code that would cause that method to fail a previously passed test. We injected five faults into the source code. The first fault caused all tests for truncate (Number n) to fail. This method is hard coded to truncate to two decimal values, we changed this "2" to a "3" and now it truncates to three decimal values while the expected is still two.

B. Our second fault was to truncateDouble (final Double value) and this fault also caused all tests to fail. This fault is a bit more serious considering that it will not output a number at all now. This method first checks to see if the input value is null, and if so it output null (because you cannot truncate a null value). It uses: if (value == null), which we have changed to: if (value != null), which causes it to now attempt to truncate a null value, which will throw an error and also will output null if the input is actually a valid number.

C. For our third, fourth, and fifth fault we changed the method truncate (Number n, int decimals). The third fault changed a logical operator in an if statement that checked to see if the decimal value was positive or negative. We changed: if (decimals < 0) to if (decimals > 0) which now causes all positive decimal values to trip this if statement and throw an illegal arguments exception while a negative decimal value will then proceed to be truncated. For our fourth fault, [INSERT SENTENCE HERE]. The fifth and final fault we simulate a simple typographical error, changing a "1" to a "2" in an integer value. This change will throw off the expected output by 1.

2. INJECTED FAULTS

A. NumberUtils.java -> truncate (Number n)
   - i. Tests #001-005
   - ii. Line 49, Source code: return truncate (n, 2);
   - iii. Line 50, Fault injection code: return truncate (n, 3);
   - iv. Expects a double truncated to 2 decimal places.
   - v. Tests #1, #2, #3, #5: FAIL
   - vi. Test #4: PASS

B. NumberUtils.java -> truncateDouble (final Double value)

i. Tests #015-020
ii. Line 62, Source code: if (value == null)
iii. Line 63, Fault injection code: if (value != null)
iv. Looks for a null input parameter. Now truncates null, and returns null on valid input.
v. Tests #16, #17, #18, #19: FAIL
vi. Tests #15, #20: PASS

C. NumberUtils.java -> truncate (Number n, int decimals)
  i. Tests #021-025
  ii. Line 85, Source code: if (decimals < 0)
  iii. Line 86, Fault injection code: if (decimals > 0)
  iv. Expects to throw error if decimal value is negative, and proceed if positive. This fault flips that and now throws error for positive and proceeds if negative.
  v. Tests #21, #23, #24, #25: FAIL
  vi. Test #22: PASS

D. NumberUtils.java -> truncate (Number n, int decimals)
  i. Tests #021-025
  ii. Line 100, Source code: if (index == -1)
  iii. Line 101, Fault injection code: if (index == 1)
  iv. Checks to see if the double has a -1 index value, then returns a truncated double as a string. The fault causes this to occur when the index value is 1.
  v. Tests #21, #23, #24, #25: FAIL
  vi. Test #22: PASS

E. NumberUtils.java -> truncate (Number n, int decimals)
  i. Tests #021-025
  ii. Line 112, Source code: final int last = index + 1 + decimals;
  iii. Line 113, Fault injection code: final int last = index + 2 + decimals;
  iv. DESCRIPTION HERE
  v. Tests #21, #23, #24, #25: FAIL
  vi. Test #22: PASS