Team: 404-Error
Members: Ricky Ramos, Matthew Schwarz, Matthew Kay
November 21, 2017

Deliverable #5

Project: OpenMRS

Task: Design and inject 5 faults into the code you are testing that will cause at least 5 tests to fail, but hopefully not all tests to fail. Exercise your framework and analyze the results.

Progress/Experience: A single fault has been inserted into each of our five methods. All faults essentially flip a boolean value (true to false or false to true). The majority of our test cases still passed. Our first method, compareNaturalAscii(String s, String t), was the only method in which our test case results remained the same even after the fault was injected.

Method 1: compareNaturalAscii(String s, String t)
Test Cases: 1 - 5
Description of Fault: The third value in compareNatural was flipped from true to false which means that case sensitivity is not taken into account.
Test Case Results:

Original: 5 Passed / 0 Failed

After Fault: 5 Passed / 0 Failed

Analysis: All of our tests were already using strings that were all lower case, so the change in case sensitivity had no effect on these specific test cases.

```
public static int compareNaturalAscii(String s, String t) {
  return compareNatural(s, t, true, null);
}
```

```
public static int compareNaturalAscii(String s, String t) {
  //Correct: return compareNatural(s, t, true, null);

  //***FAULT***
  return compareNatural(s, t, false, null);
}
```

Method 2: isStringInArray(String str, String[] arr)
Test Cases: 6 - 10
Description of Fault: The variable retVal has been initialized to true instead of false. This method will always return true.
Test Case Results:

Original: 5 Passed / 0 Failed

After Fault: 3 Passed (6,7,8) / 2 Failed (9,10)

Analysis: Because of the fault, the method will always return true meaning that the input string will always be inside the input string array. In our first three test cases, the input string is actually in the input string array, so those three still passed. In the last two tests, the string was not in the array, so those should have failed.

```java
public static boolean isStringInArray(String str, String[] arr) {
  boolean retVal = false;

  if (str != null && arr != null) {
    for (int i = 0; i < arr.length; i++) {
      if (str.equals(arr[i])) {
        retVal = true;
      }
    }
  }
  return retVal;
}
```

```java
public static boolean isStringInArray(String str, String[] arr) {
  //Correct: boolean retVal = false;

  //***FAULT***//
  boolean retVal = true;

  if (str != null && arr != null) {
    for (int i = 0; i < arr.length; i++) {
      if (str.equals(arr[i])) {
        retVal = true;
      }
    }
  }
  return retVal;
}
```

Method 3: containsOnlyDigits(String test)
Test Cases: 11 - 15
Description of Fault: The method now returns true instead of false for a non-empty string if the string contains a letter.
Test Case Results:

Original: 5 Passed / 0 Failed

After Fault: 3 Passed (11,12,13) / 2 Failed (14,15)

Analysis: Our first 3 test cases for this method still passed because they use input strings that aren't null or empty. The last two test cases failed because the first letter the method would see would make the method return true, which contradicted the expected output.

```java
public static boolean containsOnlyDigits(String test) {
 if (test != null) {
   for (char c : test.toCharArray()) {
     if (!Character.isDigit(c)) {
       return false;
     }
   }
 }
 if (test != null && !test.isEmpty()){
   return true;
 }else{
   return false;
 }
}
```

```java
public static boolean containsOnlyDigits(String test) {
 if (test != null) {
   for (char c : test.toCharArray()) {
     if (!Character.isDigit(c)) {
       //Correct: return false

       //***FAULT***
       return true;
     }
   }
 }
 if (test != null && !test.isEmpty()){
   return true;
 }else{
   return false;
 }
}
```

Method 4: containsDigit(String test)
Test Cases: 16 - 20
Description of Fault: The method now returns false if a digit appears in a string.
Test Case Results:

Original: 5 Passed / 0 Failed

After Fault: 2 Passed (19,20) / 3 Failed (16,17,18)

Analysis: Our first three test cases analyze input strings with at least one digit in them which results in the method returning false.

```
public static boolean containsDigit(String test) {
 if (test != null) {
   for (char c : test.toCharArray()) {
     if (Character.isDigit(c)) {
       return true;
     }
   }
 }
 return false;
}
```

```
public static boolean containsDigit(String test) {
 if (test != null) {
   for (char c : test.toCharArray()) {
     if (Character.isDigit(c)) {
       //Correct: return true;

       //***FAULT***
       return false;
     }
   }
 }
 return false;
}
```

Method 5: compareNaturalIgnoreCaseAscii(String s, String t)
Test Cases: 21 - 25
Description of Fault: The third value in compareNatural was flipped from false to true which means that case sensitivity is now taken into account.
Test Case Results:

Original: 5 Passed / 0 Failed

After Fault: 4 Passed (21, 23, 24, 25) / 1 Failed (22)

Analysis: Because of the fault, case sensitivity is now being taken into account. The only test case to fail because of the fault is 22 because capital 'T' is now being compared to lower case 'w'.

```
public static int compareNaturalIgnoreCaseAscii(String s, String t) {
  return compareNatural(s, t, false, null);
}
```

```
public static int compareNaturalIgnoreCaseAscii(String s, String t) {
  //Correct: return compareNatural(s, t, false, null);

  //***FAULT***
  return compareNatural(s, t, true, null);
}
```