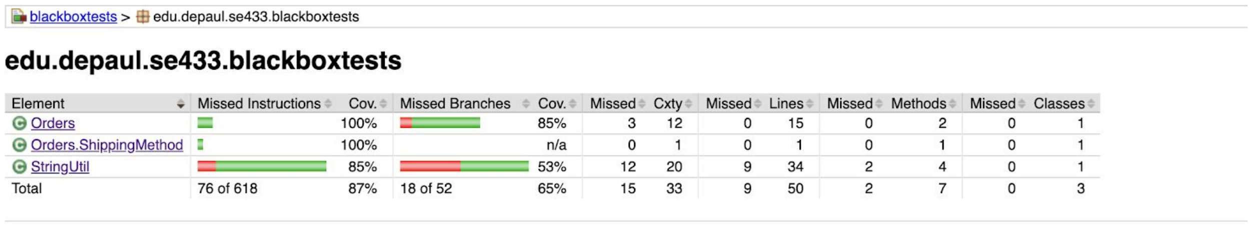
James Valles
SE 433/333
Assignment #8

Initial Findings

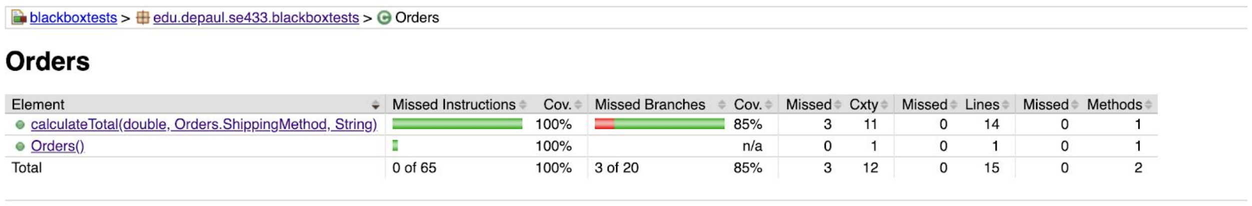

blackboxtests > edu.depaul.se433.blackboxtests

edu.depaul.se433.blackboxtests

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Orders		100%		85%	3	12	0	15	0	2	0	1
Orders.ShippingMethod		100%		n/a	0	1	0	1	0	1	0	1
StringUtil		85%		53%	12	20	9	34	2	4	0	1
Total	76 of 618	87%	18 of 52	65%	15	33	9	50	2	7	0	3

Overall Orders Class

Order class has 100% line coverage 0/15 lines missed. 2 methods Orders() and calculateTotal(), 0 missed. 0/1 classes missed. 3/12 Cyclomatic complexity missed (75% coverage). Branch coverage 85% (3/20 branches missed).


blackboxtests > edu.depaul.se433.blackboxtests > Orders

Orders

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
calculateTotal(double, Orders.ShippingMethod, String)		100%		85%	3	11	0	14	0	1
Orders()		100%		n/a	0	1	0	1	0	1
Total	0 of 65	100%	3 of 20	85%	3	12	0	15	0	2

Drill deeper to review the calculateTotal() method

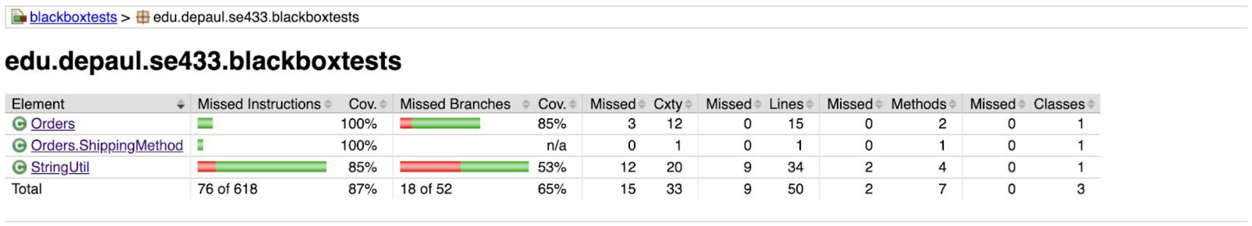
Order class has one method (jacoco considers constructors and static initializers as methods) called calculateTotal(), which initially has 100% line coverage, 0/14 lines missed.
Branch Coverage initially is 85%. It missed 3/20 branches.

This is a list of what branch is missed:

if ((destinationState.equals("California") || destinationState.equals("Illinios") || destinationState.equals("New York").
3/11 Cyclomatic complexity missed (72.7% coverage)

Drill deeper to review Orders() - jacoco considers constructors and static initializers as methods

0/1 lines missed 100% line coverage, **0/1** Cyclomatic complexity missed (100% coverage).
Branch coverage n/a

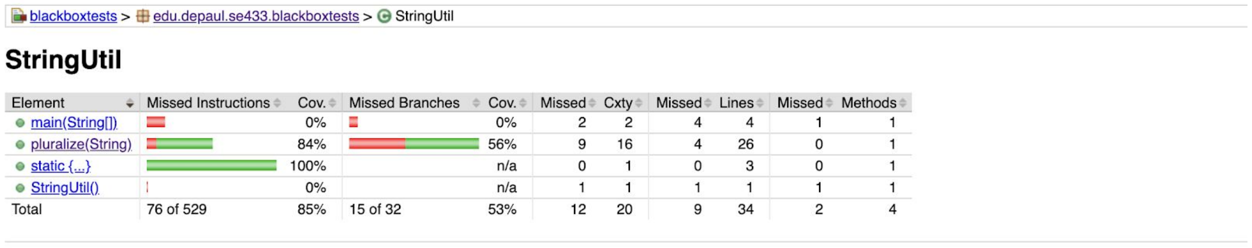

blackboxtests > edu.depaul.se433.blackboxtests

edu.depaul.se433.blackboxtests

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Orders		100%		85%	3	12	0	15	0	2	0	1
Orders.ShippingMethod		100%		n/a	0	1	0	1	0	1	0	1
StringUtil		85%		53%	12	20	9	34	2	4	0	1
Total	76 of 618	87%	18 of 52	65%	15	33	9	50	2	7	0	3

Overall StringUtil

StringUtil class has 73.5% line coverage 9/34 lines missed. 4 methods (jacoco counts constructors and static initializers as methods): main(), pluralize(), static(), StringUtil() , 2 missed. 0/1 classes missed. 12/20 Cyclomatic complexity missed (40% coverage). Branch coverage 53% (15 of 32 branches missed).


blackboxtests > edu.depaul.se433.blackboxtests > StringUtil

StringUtil

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
main(String[])		0%		0%	2	2	4	4	1	1
pluralize(String)		84%		56%	9	16	4	26	0	1
static {...}		100%		n/a	0	1	0	3	0	1
StringUtil()		0%		n/a	1	1	1	1	1	1
Total	76 of 529	85%	15 of 32	53%	12	20	9	34	2	4

Drill deeper to review the pluralize() method

StringUtil class has one real method (jacoco considers constructors and static initializers as methods) called pluralize(), which **initially has 84.6% line coverage, 4/26 lines missed. Branch Coverage initially is 56%. It missed 13/30 branches.**

This is a list of what branch is missed:

```
if (word != null), if (word.indexOf(punct.charAt(i)) >= 0), if
(word.equals(uncountable[i])), if (word.equals(s1)), if (c1 ==
'y' && (c2 != 'a' || c2 != 'e' || c2 == 'i' || c2 == 'o' || c2
== 'u')), for (int i = 0; i < words.length; i++)
```

9/16 Cyclomatic complexity missed (43.7% coverage)

Drill deeper to review main(), StringUtil(), - jacoco considers constructors and static initializers as methods

main(): 4/4 lines missed 0% line coverage, **2/2** Cyclomatic complexity missed (0% coverage). Branch coverage 0%

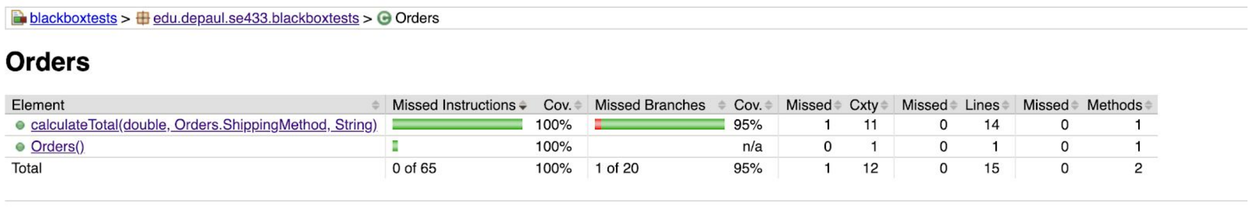
StringUtil(): (Constructor) 1/1 lines missed 0% line coverage, **1/1** Cyclomatic complexity missed (0% coverage). Branch coverage n/a

Static{...} : Missed Branches n/a: 0/1 Cyclomatic complexity, 0/3 liness missed (100% coverage)

Tests Added to cover more code
Orders

Will use “California,” “Illinois,” and “NewYork” instead of just CA, IL, NY.

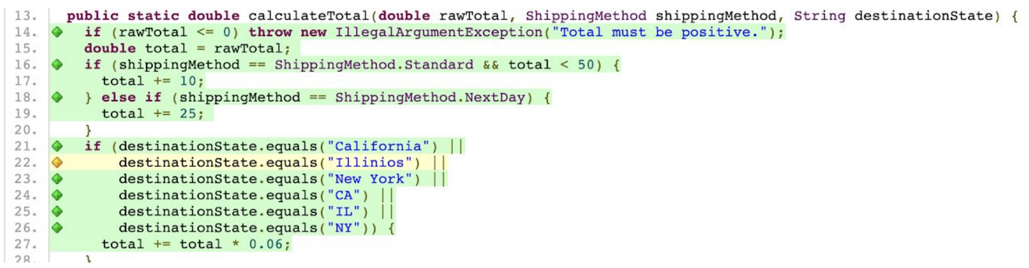
After adding these tests, I got 95% branch coverage. But, I noticed there is typo in the Orders class, “Illinois” is spelled incorrectly



blackboxtests > edu.depaul.se433.blackboxtests > Orders

Orders

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
calculateTotal(double, Orders.ShippingMethod, String)	🟩	100%	🟥🟩	95%	1	11	0	14	0	1
Orders()	🟩	100%		n/a	0	1	0	1	0	1
Total	0 of 65	100%	1 of 20	95%	1	12	0	15	0	2

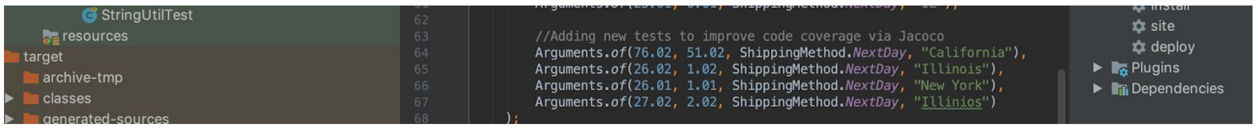


```
public static double calculateTotal(double rawTotal, ShippingMethod shippingMethod, String destinationState) {
  if (rawTotal <= 0) throw new IllegalArgumentException("Total must be positive.");
  double total = rawTotal;
  if (shippingMethod == ShippingMethod.Standard && total < 50) {
    total += 10;
  } else if (shippingMethod == ShippingMethod.NextDay) {
    total += 25;
  }
  if (destinationState.equals("California") ||
      destinationState.equals("Illinios") ||
      destinationState.equals("New York") ||
      destinationState.equals("CA") ||
      destinationState.equals("IL") ||
      destinationState.equals("NY")) {
    total += total * 0.06;
```

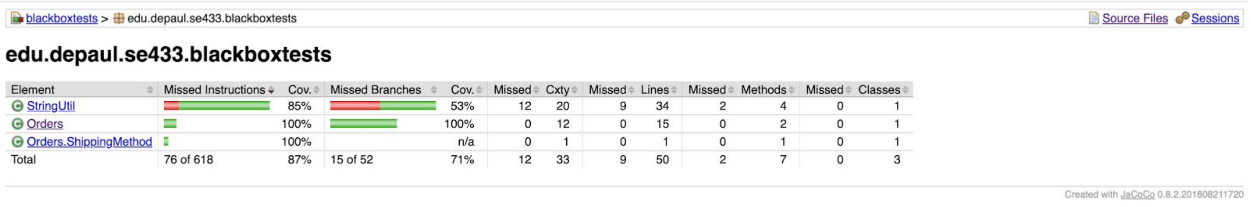
I added another test case, and I got 100% branch coverage. **<u>But under normal circumstances, it would be extremely difficult to reach 100% branch coverage, because “Illinois” is spelled wrong “Illinios” in the Orders() class.</u>**

After adding the following additional test cases. I was able to achieve 100% line coverage and 100% branch coverage:

Arguments.*of*(76.02, 51.02, ShippingMethod.*NextDay*, "California"),
Arguments.*of*(26.02, 1.02, ShippingMethod.*NextDay*, "Illinois"),
Arguments.*of*(26.01, 1.01, ShippingMethod.*NextDay*, "New York"),
Arguments.*of*(27.02, 2.02, ShippingMethod.*NextDay*, "Illinios")



```
    //Adding new tests to improve code coverage via Jacoco
    Arguments.of(76.02, 51.02, ShippingMethod.NextDay, "California"),
    Arguments.of(26.02, 1.02, ShippingMethod.NextDay, "Illinois"),
    Arguments.of(26.01, 1.01, ShippingMethod.NextDay, "New York"),
    Arguments.of(27.02, 2.02, ShippingMethod.NextDay, "Illinios")
);
```

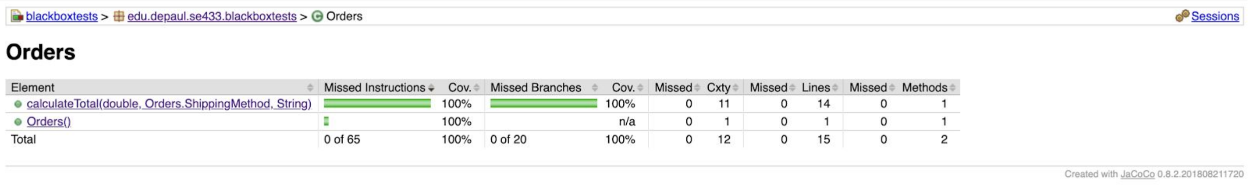


blackboxtests > edu.depaul.se433.blackboxtests   Source Files Sessions

edu.depaul.se433.blackboxtests

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
StringUtil	🟥🟩	85%	🟥🟩	53%	12	20	9	34	2	4	0	1
Orders	🟩	100%	🟩	100%	0	12	0	15	0	2	0	1
Orders.ShippingMethod	🟩	100%		n/a	0	1	0	1	0	1	0	1
Total	76 of 618	87%	15 of 52	71%	12	33	9	50	2	7	0	3

Created with JaCoCo 0.8.2.201808211720



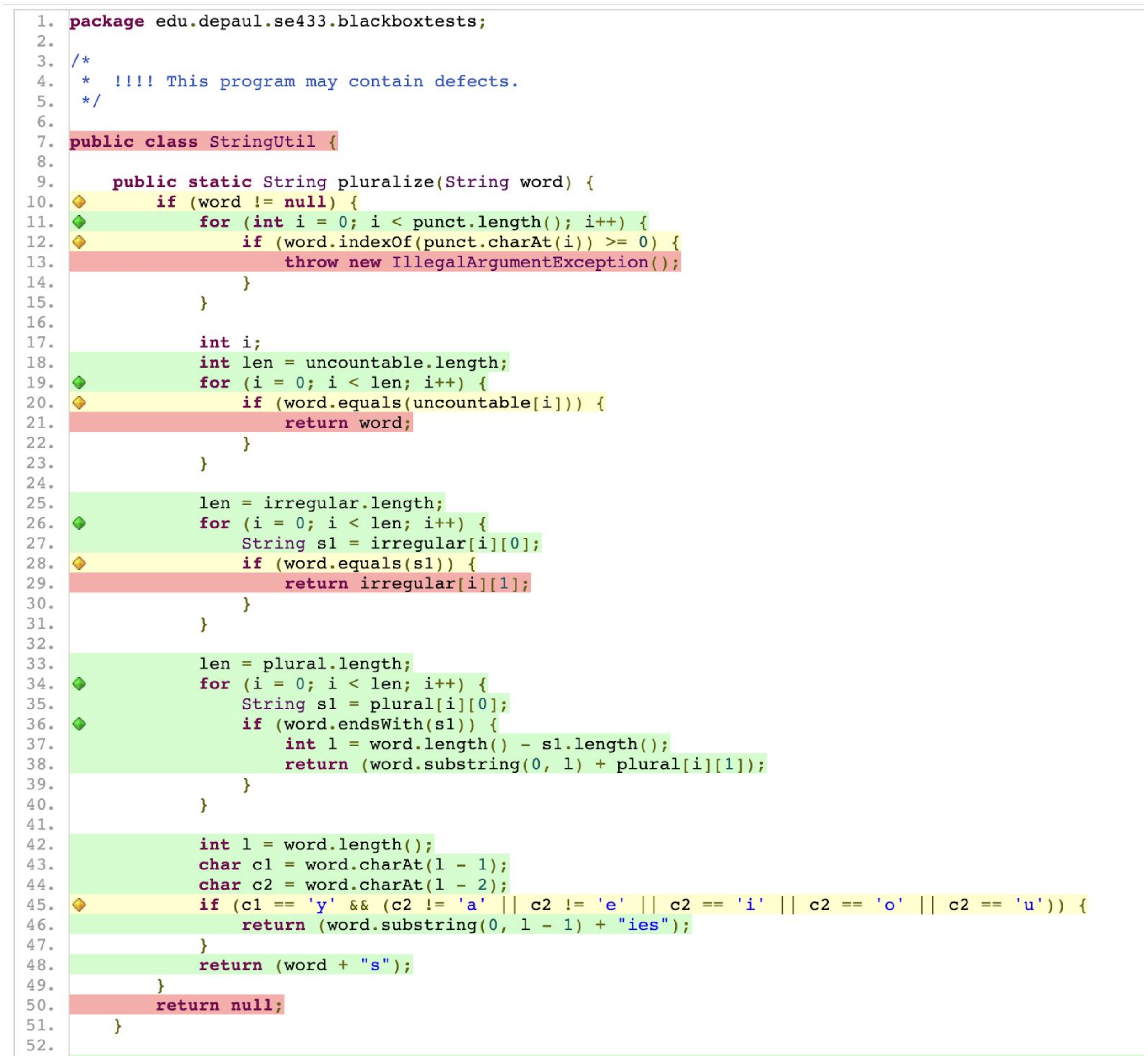
blackboxtests > edu.depaul.se433.blackboxtests > Orders   Sessions

Orders

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
calculateTotal(double, Orders.ShippingMethod, String)	🟩	100%	🟩	100%	0	11	0	14	0	1
Orders()	🟩	100%		n/a	0	1	0	1	0	1
Total	0 of 65	100%	0 of 20	100%	0	12	0	15	0	2

Created with JaCoCo 0.8.2.201808211720

<u>StringUtil</u>
<u>Tests Added to cover more code:</u>



```java
package edu.depaul.se433.blackboxtests;

/*
 *  !!!! This program may contain defects.
 */

public class StringUtil {

    public static String pluralize(String word) {
        if (word != null) {
            for (int i = 0; i < punct.length(); i++) {
                if (word.indexOf(punct.charAt(i)) >= 0) {
                    throw new IllegalArgumentException();
                }
            }

            int i;
            int len = uncountable.length;
            for (i = 0; i < len; i++) {
                if (word.equals(uncountable[i])) {
                    return word;
                }
            }

            len = irregular.length;
            for (i = 0; i < len; i++) {
                String s1 = irregular[i][0];
                if (word.equals(s1)) {
                    return irregular[i][1];
                }
            }

            len = plural.length;
            for (i = 0; i < len; i++) {
                String s1 = plural[i][0];
                if (word.endsWith(s1)) {
                    int l = word.length() - s1.length();
                    return (word.substring(0, l) + plural[i][1]);
                }
            }

            int l = word.length();
            char c1 = word.charAt(l - 1);
            char c2 = word.charAt(l - 2);
            if (c1 == 'y' && (c2 != 'a' || c2 != 'e' || c2 == 'i' || c2 == 'o' || c2 == 'u')) {
                return (word.substring(0, l - 1) + "ies");
            }
            return (word + "s");
        }
        return null;
    }

```

//Adding new tests to improve code coverage via Jacoco
Arguments.*of*("equipment", "equipment"),
Arguments.*of*("person", "person"),
Arguments.*of*(null, null),
Arguments.*of*(",", ","),
Arguments.*of*("array", "arrays"),
Arguments.*of*("chimney", "chimneys"),
Arguments.*of*("guy", "chimneys")

if (word != null),
Test to cover: Arguments.*of*(null, null)

****if (word.indexOf(punct.charAt(i)) >= 0)**
Test to cover: Arguments.*of*(“,”, “,”)

if (word.equals(uncountable[i])),
Test: Arguments.*of*("equipment", "equipment")

Added the following to cover the main() that had 2 branches and 4 lines that were not covered.

- Missed 4 lines in StringUtil class, and 2 branches missed.

Would need to run this code from test file to cover. Wouldn't make sense to run this as the test cases are hardcoded and the sout is irrelevant.

```
public static void main(String[] args) {
        String[] words = { "car", "woman", "house", "quality" };
        for (int i = 0; i < words.length; i++) {
            System.out.println("The plural of " + words[i] + " is " +
pluralize(words[i]));
        }
    }

```

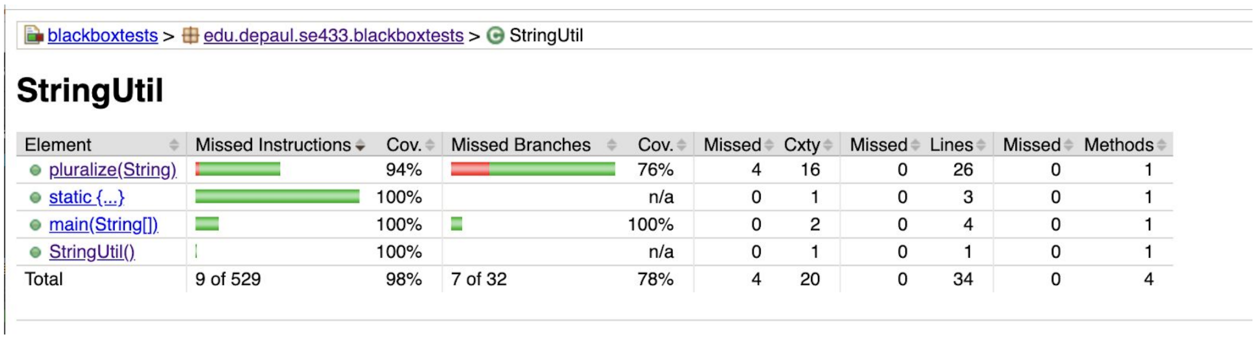
Went for 0% branch coverage to 100%. And went from 0% line coverage, to 100% branch coverage. See test and final output below.

```
/**Added this test to increase converage of Main() in StringUtil and StringUUtil constructor
* This test checks that main doesn't throw an illegalArgumentException */

@ParameterizedTest
@DisplayName("Exception thrown main method.")
@MethodSource("invalidMainExceptionTest")
void mainMethodCheck(Class expected, String[] args) {

 StringUtil stringUtil = new StringUtil();
 try{
  stringUtil.main(args);}
 catch (Exception e){
   fail("Should not throw an exception");
 }
}

private static Stream<Arguments> invalidMainExceptionTest() {
 return Stream.of(
     Arguments.of(Test.None.class, null)
 );
}
```


blackboxtests > edu.depaul.se433.blackboxtests > StringUtil

StringUtil

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
pluralize(String)	🟥🟩	94%	🟥🟩	76%	4	16	0	26	0	1
static {...}	🟩	100%		n/a	0	1	0	3	0	1
main(String[])	🟩	100%	🟩	100%	0	2	0	4	0	1
StringUtil()	🟩	100%		n/a	0	1	0	1	0	1
Total	9 of 529	98%	7 of 32	78%	4	20	0	34	0	4

Difficulties getting to 100%

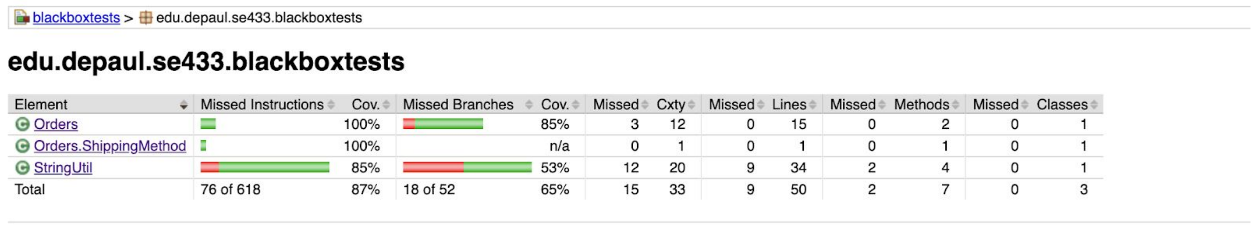
- For orders, it was difficult to get to 100% branch coverage, as “Illinois” is spelled wrong. Wouldn’t make sense to spell it wrong in a test in the “real” world. I did so to increase the coverage, but I know this is not best practice. Without doing this I wouldn’t be able to achieve 100% coverage. For this assignment, the goal was to increase coverage. I found myself writing some tests that made no sense (ie: for Orders, testing Illinios. This taught me to remember what was taught in class, the goal is not just to increase coverage. But to write tests that make sense.

- Getting full branch coverage here proved to be the most difficult. I was able to get it down from 9/12 miss branches to 7/12 branches . I can’t really get it down any further, because, having a word that ends with y && a, e, or not i, not o, not u (This part will always be true: `(c2 != 'a' || c2 != 'e' || c2 == 'i' || c2 == 'o' || c2 == 'u'))`). So it doesn’t really make sense.
  ```
      if (c1 == 'y' &&
  ```

- Overall I found this assignment to be very interesting. I not only had to make sense of the results, but look at the program in an entirely different ways. The coverage report help produce new test cases I hadn’t even thought of. Thank you!

Final Results After New Test Cases Screenshots

Before

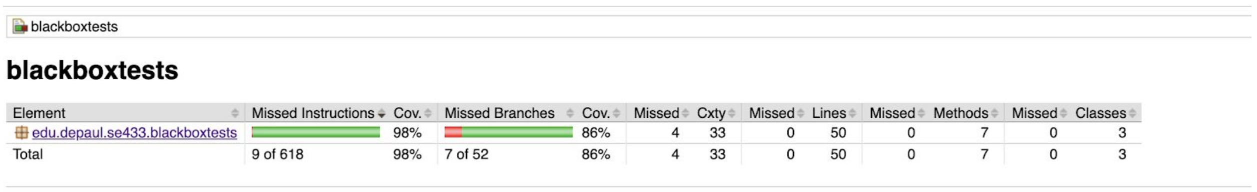


blackboxtests > edu.depaul.se433.blackboxtests

edu.depaul.se433.blackboxtests

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Orders	🟩	100%	🟥🟩	85%	3	12	0	15	0	2	0	1
Orders.ShippingMethod	🟩	100%		n/a	0	1	0	1	0	1	0	1
StringUtil	🟥🟩	85%	🟥🟩	53%	12	20	9	34	2	4	0	1
Total	76 of 618	87%	18 of 52	65%	15	33	9	50	2	7	0	3

After



blackboxtests

blackboxtests

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
edu.depaul.se433.blackboxtests	🟩	98%	🟥🟩	86%	4	33	0	50	0	7	0	3
Total	9 of 618	98%	7 of 52	86%	4	33	0	50	0	7	0	3

Overall, coverage went **from 9 missed lines to 100% line coverage**, and branch went from **65% branch coverage to 86% (18/52 missed to 7/52 missed).**

blackboxtests > edu.depaul.se433.blackboxtests > Orders

Orders

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
calculateTotal(double, Orders.ShippingMethod, String)		100%		100%	0	11	0	14	0	1
Orders()		100%		n/a	0	1	0	1	0	1
Total	0 of 65	100%	0 of 20	100%	0	12	0	15	0	2

blackboxtests > edu.depaul.se433.blackboxtests > StringUtil

StringUtil

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
pluralize(String)		94%		76%	4	16	0	26	0	1
static {...}		100%		n/a	0	1	0	3	0	1
main(String[])		100%		100%	0	2	0	4	0	1
StringUtil()		100%		n/a	0	1	0	1	0	1
Total	9 of 529	98%	7 of 32	78%	4	20	0	34	0	4