

Assignment 02

Question 1

Answer the following questions: (1 point each)

a) Identify the fault.

```
if (x[i] >= 0) should be if (x[i] > 0)
```

This line will cause the function to count all non-negative numbers rather than just positive numbers.

b) Identify a test case that does NOT execute the fault.

```
test: x = [] Expected = 0
```

c) Identify a test case that executes the fault, but does not result in an error state.

```
test: x=[-4, 2, 0, 2] Expected = 2
```

d) Identify a test case that results in an error, but not a failure. (Hint: Error results in failure if not handled by the program)

```
test: x=null
```

If x is null, then there will be a thrown error

e) For the given test case in the textbox, identify the first error state during runtime. Describe the complete state including the values of each of the variables, expected and actual results.

```
Input: x=[-4,2,0,2]
```

```
Error State: i = 2; x[i] = 0; count = 2; x.length = 4;
```

```
Expected Result: count++ line ignored, count left as 1.
```

```
Actual Result: count++ line executes, counted incremented to 2.
```

Question 2

a) (7 points) Add main method and additional methods (as appropriate) to test the attached java files. Driver will read test inputs from a txt file (user.txt). The user.txt will contain test inputs in the following format:

```
3224400
3224410
3224420
```

Expected console output from the driver:

```
3224400 is a local prepaid user
3224410 is a local prepaid user
3224420 is a local prepaid user
```

b) Answer the following questions based on Q2a: (3 Points)

i. Identify any debugging strategy you used.

Brute force.

ii. Explain why did you use/choose that debugging strategy.

Because,
This is a small program and
It reports major issues immediately.

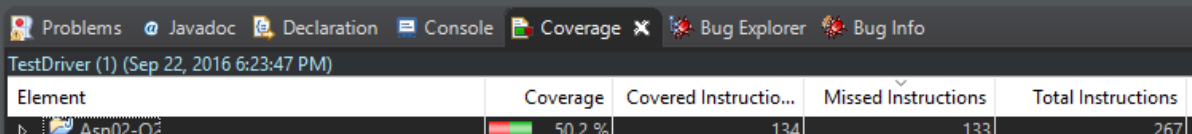
iii. Which other debugging strategy could be used?

Backtracking.

Question 3

a) After solving Q2a, run the driver for one sample input. What is the initial code coverage for PolicyUser, LocalUser, and RoamingUser classes using EclEmma code coverage tool?

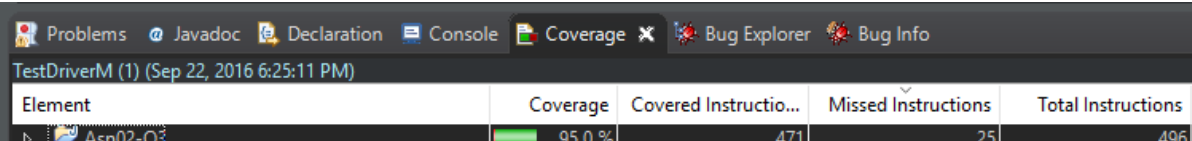
Initial code coverage was 50.2%.



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Asn02-Q2	50.2 %	134	133	267

b) Add additional codes in the driver to increase code coverage for the attached java classes to ~100%.

Final code coverage was 95%. The remaining 5% are try/catch exceptions that don't occur during normal operation, and variable declarations without assignment.



Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Asn02-Q2	95.0 %	471	25	496

These codes are provided under the `csc455.fa16.g08.asn02.jh`, and `csc455.fa16.g08.asn02.jhM` packages in the repo.

Question 4

See the JAVA code on the next few pages. Use this code in a new JAVA project (name the Project/File as FindBugsGroupNumber.java), and perform a static code analysis using the FindBugs tool.

- Identify the reported bugs from the tool, and give a brief overview of these bugs.
- Find a solution to fix each of these bugs, and submit the modified (working version) code/project.
- Use a table to answer a) and b)

Bug Line	Overview	Edit
82: final Long value = (Long) doubleValue;	Tries to assign a Double value as a Long.	Final Double value = (Double) doubleValue;
94: final SecurityException value = (SecurityException) exception;	Tries to assign RuntimeException as SecurityException	final RuntimeException value = (RuntimeException) exception;
106: System.out.println(" - " + (value instanceof Double));	Will always return false, since Double can't be Long.	System.out.println(" - " + (value instanceof Double));
118: final String[] stringArray = (String[]) stringVector.toArray();	Can't downcast toArray() to String[]	System.out.println(" - " + stringArray.length);
131: final BigDecimal bigDecimal = new BigDecimal(3.1);	Big Decimal value created from a double, will give different long value.	Final BigDecimal bigDecimal = new BigDecimal.valueOf(3.1)
145: System.out.println(" - " + (string1 == string2));	Comparison of strings using == operation. May result in false fail.	System.out.println(" - " + (string1.equals(string2)));
157: System.out.println(String.format(" - %s > %s", "10", "9"));	Illegal string formatting syntax..	System.out.println(String.format(" - %s > %s", "10", "9"));
166: bigDecimal.add(BigDecimal.ONE);	Return value of bigDecimal.add ignored.	Final BigDecimal bigDecimal = bigDecimal.add(BigDecimal.ONE);
178: if (null != value & value.length() > 2) {	Bitwise comparing values	if (null != value && value.length() > 2)
196: if (value = true) {	Assigns boolean value, rather than testing it.	If (value == true) {