COM S 417: Software Testing
Exam 1

March 3rd, 2022

This exam will be **available from 2PM to 9PM**. You can write your answers on this exam and scan/return as a .pdf, or you can type your answers and return.

Answer all 6 questions.

Last Name: ________________________

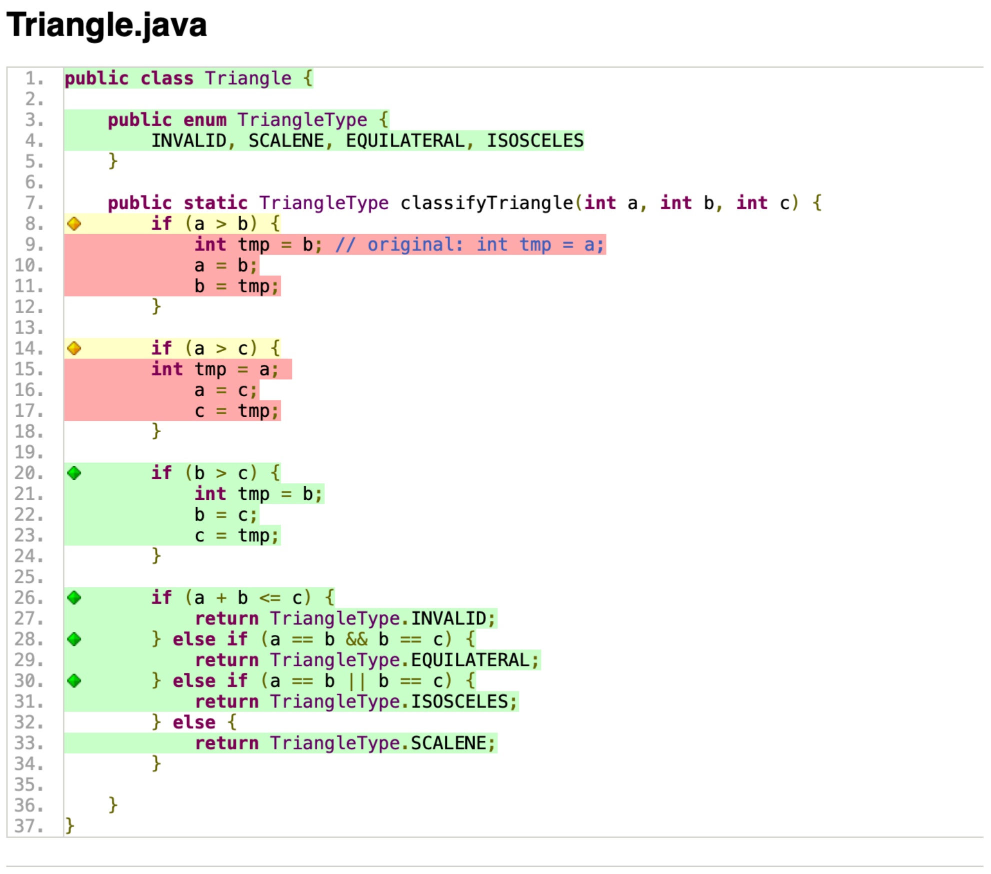
First Name: ________________________

Section: ________________________

Question	**Possible Points**	**Points**
1	30	
2	10	
3	20	
4	20	
5	10	
6	10	
Total	100	

Q1. (30 points)
Figure 1 shows a **Triangle program**. This performs the same functionality as the program we have used in class. It takes in three integers and returns the type of triangle (EQUILATERL, SCALENE, ISOCELES, or INVALID). It has a fault on line 9 (the original correct code is given as a comment). There are 8 JUnit test cases shown here in the form (a, b, c, oracle) that were run to create the report. Note: the last test simply calls the constructor to cover line 1. You can ignore it for this question. Please answer the questions below based on these test cases and the coverage report. You can use any tools we used in class to help solve this problem, however you need to provide clear explanations for your answers below.

1. (0, 0, 0, INVALID)
2. (4, 4, 10, INVALID)
3. (3, 3, 3, EQUILATERAL)
4. (-1, 8, -1, INVALID)
5. (6, 8, 8, ISOSCELES)
6. (4, 5, 6, SCALENE)
7. (7, 8, 7, ISOSCELES)
8. new Triangle() // this just calls the constructor



Triangle.java

```
 1. public class Triangle {
 2.
 3.     public enum TriangleType {
 4.         INVALID, SCALENE, EQUILATERAL, ISOSCELES
 5.     }
 6.
 7.     public static TriangleType classifyTriangle(int a, int b, int c) {
 8.         if (a > b) {
 9.             int tmp = b; // original: int tmp = a;
10.             a = b;
11.             b = tmp;
12.         }
13.
14.         if (a > c) {
15.         int tmp = a;
16.             a = c;
17.             c = tmp;
18.         }
19.
20.         if (b > c) {
21.             int tmp = b;
22.             b = c;
23.             c = tmp;
24.         }
25.
26.         if (a + b <= c) {
27.             return TriangleType.INVALID;
28.         } else if (a == b && b == c) {
29.             return TriangleType.EQUILATERAL;
30.         } else if (a == b || b == c) {
31.             return TriangleType.ISOSCELES;
32.         } else {
33.             return TriangleType.SCALENE;
34.         }
35.
36.     }
37. }
```

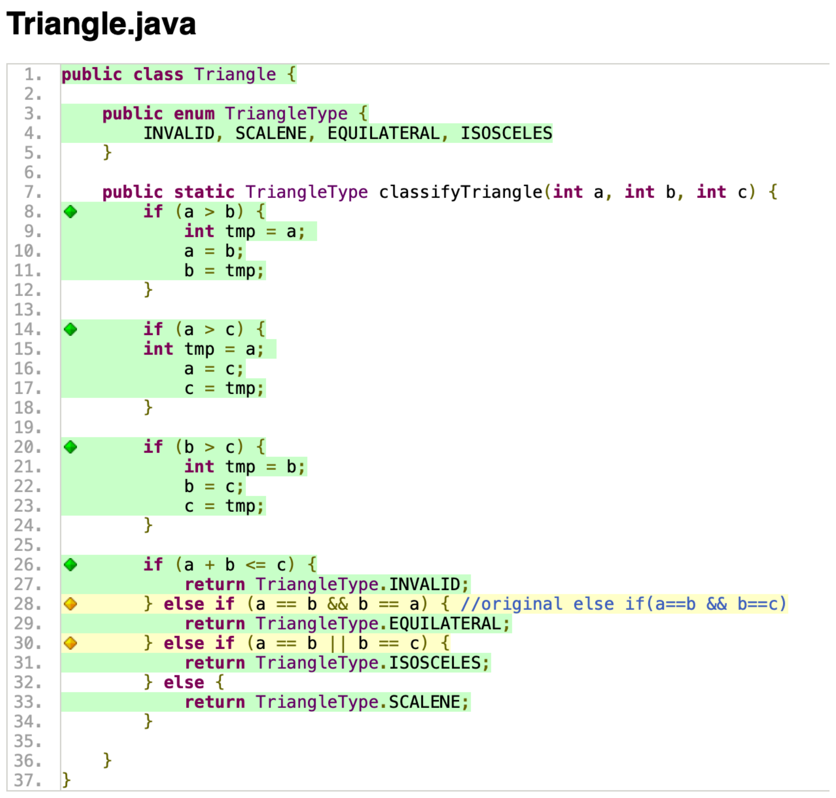
Figure 1. Jacoco Coverage for Triangle Class

(a) All of the *test cases pass when run with JUnit*. **Explain why this happens using the *RIPR* model**. *Be explicit*. Use test cases, line numbers and oracles to relate this to the RIPR model. **(10 points)**

(b) Add **one new test case** (in the same form with inputs/oracle) to cover the missing lines/branches (i.e. to reach 100 percent coverage), **that still does not find the fault**. Explain why this test is insufficient to expose the fault (use the code to explain). Note – you must use a correct oracle (i.e. you can't create a test case that has the wrong answer to make this work). **(5 points)**

(c) Add a test case that **reveals the fault. Explain. (5 points)**

(d) Suppose the fault is in a different location (see Figure 2 below). This has been run with all of the additional test cases which led to 100 percent branch and statement coverage before. One of the seven test cases from above failed. Which test case failed? (**5 points**)



Triangle.java

```
 1. public class Triangle {
 2.
 3.     public enum TriangleType {
 4.         INVALID, SCALENE, EQUILATERAL, ISOSCELES
 5.     }
 6.
 7.     public static TriangleType classifyTriangle(int a, int b, int c) {
 8.         if (a > b) {
 9.             int tmp = a;
10.             a = b;
11.             b = tmp;
12.         }
13.
14.         if (a > c) {
15.         int tmp = a;
16.             a = c;
17.             c = tmp;
18.         }
19.
20.         if (b > c) {
21.             int tmp = b;
22.             b = c;
23.             c = tmp;
24.         }
25.
26.         if (a + b <= c) {
27.             return TriangleType.INVALID;
28.         } else if (a == b && b == a) { //original else if(a==b && b==c)
29.             return TriangleType.EQUILATERAL;
30.         } else if (a == b || b == c) {
31.             return TriangleType.ISOSCELES;
32.         } else {
33.             return TriangleType.SCALENE;
34.         }
35.
36.     }
37. }
```

Figure 1. Alternative Fault in Triangle program (Line 28)

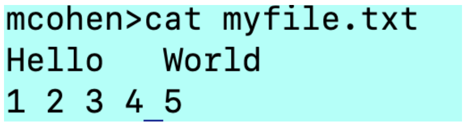
(e) Explain why 100 percent branch coverage is no longer possible. *Note that both line 28 and 30 are missing 1 of 4 branches*. **(5 points)**

Q2. (10 points) The **'cat'** program on Unix concatenates and writes files to standard output. It reads in one or more files or and then outputs the contents. Use the following information about this program to answer the questions below.

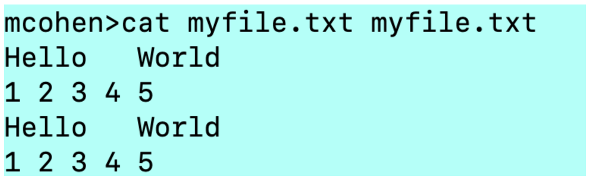
If myfile.txt has the contents:

```
Hello   World
1 2 3 4 5
```

Then the following shows the results of using cat with one and two files:



```
mcohen>cat myfile.txt
Hello   World
1 2 3 4_5
```



```
mcohen>cat myfile.txt myfile.txt
Hello   World
1 2 3 4 5
Hello   World
1 2 3 4 5
```

The **man page** for cat shows the following (partial) usage of the program:

cat [OPTION]... [FILE]...

-b, --number-nonblank
    number nonempty output lines, overrides -n

-E, --show-ends
    display $ at end of each line

-n, --number
    number all output lines

-s, --squeeze-blank
    suppress repeated empty output lines

-T, --show-tabs
    display TAB characters as ^I

List **4 possible characteristics** and **their blocks** you can use to partition the input domain of the program options using program functionality (i.e. Functionality based IDM). Label which are the characteristics and which are the blocks.
Note – there are more than four we can use here – just provide four. **(10 points)**

Q3. (20 points)
Below (Figure 3) is a GUI application and its event flow graph created by a testing tool such as GUITAR. The Dialog box (with event e5) is a modal window that opens only when event e3 is clicked. Answer the following questions about using this graph for testing the application.

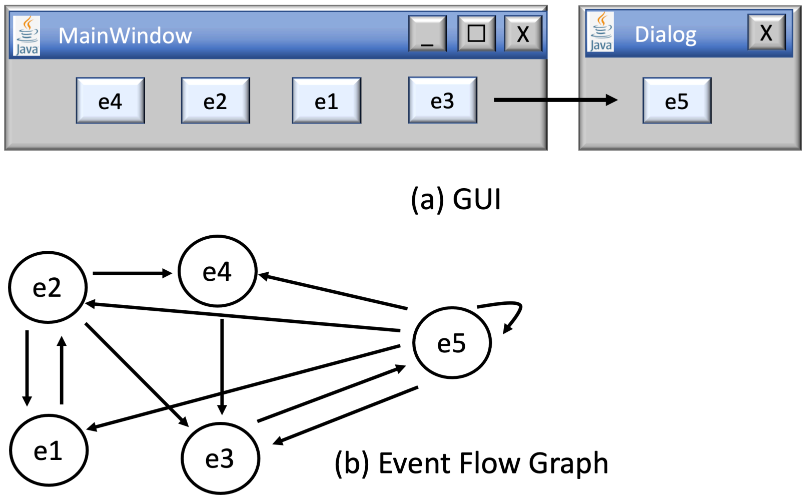


Figure 2. GUI and its EFG

(a) Suppose you want to generate test cases using this EFG. Use the two following coverage criterion and list all test cases from the EFG to satisfy that criterion. (note – length is the number of events) **(10 points)**

Length 1:

Length 2:

(b) Is there a **coverage subsumption relationship** between the length 1 and length 2 test cases? If so, state what that relationship is, and demonstrate why it holds. If there is none, state why. **(5 points)**

(c) List one test case (any length) that needs a **reaching event (a prefix-value)** and explain what this is and why we need it. **(3 points)**

(d) Use the **graph to explain why** we need to define a length for the coverage criterion and can't generate all possible test cases from this graph. (**2 points**)

Q4. (20 points)
Answer these questions about automated testing tools.

(a) In class we seen multiple automation tools such as **Selenium, Evosuite**, **AFL** and **GUITAR** to automate testing. All four automate the process of running tests once generated. However, not all generate tests or use models to guide coverage. Some have automated oracles while others require manual oracles. Some work from a code perspective while others work on different types of artifacts. **Pick two tools and differentiate them *based on these properties*** using what you have learned in class. Provide concrete examples. **(10 points)**

(b) Evosuite creates oracles based on existing code. Explain a problem with this approach that could lead to faults being missed. Provide **a small example that demonstrates** the issue**. (5 points)**

(c) Why might this problem with automated oracles based on code, be less of a problem in regression testing? **Describe a regression testing workflow where we can trust the automated oracles. (5 points)**

Q5. (10 points)
Suppose you are testing a program that has images which need to be tested. Each image has a both a shape and a pattern. The possible combinations are shown below. For instance, square images are always striped. But star images may be dotted or solid.

Shape	Pattern
square	striped
triangle	solid
star	dotted, solid
octagon	striped, solid
cloud	dotted

Now suppose we design two test criteria, C1 and C2 for this program. The first test criteria (C1) *covers all shapes of the images* and the second criteria (C2) *covers all of the patterns*.

There is a subsumption relationship between C1 and C2. State what the subsumption relationship is and **explain with an example**. As part of this show a test suite for each C1 and C2 that covers all of the test requirements.

Q6. Controllability and **Observability** both impact the *Testability* of a program. Use concrete examples ***to explain how both controllability and observability*** may be difficult when testing some programs. Be specific. You can use different types of programs for the two different properties. **(10 Points)**

--- Exam End---
Overflow Pages

Overflow Pages