Question 1

Coverage screenshot for MoneyBagTest before modification:

Problems | Javadoc | Declaration | **Console** | Coverage Explorer | Test Run Explorer | Coverage

MoneyBagTest (Feb 8, 2025 3:24:51 PM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
lab2	94.3 %	920	56	976
test	92.7 %	507	40	547
(default package)	92.7 %	507	40	547
MoneyTest.java	0.0 %	0	23	23
MoneyBagTest.java	96.8 %	507	17	524
main	96.3 %	413	16	429
(default package)	96.3 %	413	16	429
MoneyBag.java	96.9 %	286	9	295
Money.java	94.8 %	127	7	134

Coverage screenshot for MoneyBagTest after modification
3 methods commented out (TestMoneyBagEquals(), TestMoneyEquals(), and TestMoneyHash()):

Problems | Javadoc | Declaration | **Console** | Coverage Explorer | Test Run Explorer | Clover Dashboard | Coverage

MoneyBagTest (Feb 10, 2025 10:20:29 PM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
lab2	74.9 %	801	269	1,070
test	71.1 %	396	161	557
q3	0.0 %	0	99	99
q2	0.0 %	0	36	36
(default package)	93.8 %	396	26	422
MoneyTest.java	0.0 %	0	23	23
MoneyBagTest.java	99.2 %	396	3	399
main	78.9 %	405	108	513
q3	0.0 %	0	61	61
(default package)	94.4 %	405	24	429
MoneyBag.java	95.6 %	282	13	295
Money.java	91.8 %	123	11	134
q2	0.0 %	0	23	23

Coverage screenshot for MoneyTest before modification:

Problems | Javadoc | Declaration | **Console** | Coverage Explorer | Test Run Explorer | Coverage

MoneyTest (Feb 8, 2025 3:07:52 PM)

Element	Coverage	Covered Instructions	Missed Instruct	Total Instructions
Money	3.9 %	38	938	976
test	4.2 %	23	524	547
(default package)	4.2 %	23	524	547
MoneyBagTest.java	0.0 %	0	524	524
MoneyTest.java	100.0 %	23	0	23
main	3.5 %	15	414	429
(default package)	3.5 %	15	414	429
MoneyBag.java	0.0 %	0	295	295
Money.java	11.2 %	15	119	134

This question covers the Clover coverage software which visually shows how effective the test cases are at covering all the possible scenarios for the given main classes.

Question 2

Statement Coverage
Scenario 1: (a = 2, b = 3)
10 total statements
Line 1, 2, 3, 4, 10 coverage
5/10 = 50% statement coverage

Scenario 2: (a = 3, b = 2)
10 total statements
Line 1, 2, 5, 6, 7, 10 coverage
6/10 = 60% statement coverage

Scenarios 1 and 2:
10 total statements
Line 1, 2, 3, 4, 5, 6, 7, 10 coverage
8/10 = 80% statement coverage

Given Scenarios 1 and 2 we see that the overall coverage doesn't cover every statement so statement coverage is not 100%. To make it 100% we will have to introduce new inputs where a and b are equal to execute all the statements

Scenario 3:
10 total statements
Line 1, 2, 5, 8, 9, 10
6/10 = 60% statement coverage

Scenario 1, 2, and 3:
10 total statements
Line 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
10/10 = 100% statement coverage

Branch Coverage
4 total branches
if (b>a) = true
if (b>a) = false
else-if (a>b) = true
else-if (a>b) = false

Scenario 1: (a = 2, b = 3)
Yes for if statement
Others are skipped over
1/4 = 25% branch coverage

Scenario 2: (a = 3, b = 2)
No for if statement
Yes for else-if statement
2/4 = 50% branch coverage

To cover all statements and branches, we need a case where a == b, so that the else case (lines 8-9) executes.

Scenario 3: (a = 2, b = 2)
No for if statement
No for else-if statement
2/4 = 50% branch coverage

Now, all statements and branches are covered!

Statement Coverage

Line	a = 2, b = 3	a = 3, b = 2	a = 2, b = 2	Covered
1	✅	✅	✅	✅
2	✅	✅	✅	✅
3	✅	❌	❌	✅
4	✅	❌	❌	✅
5	❌	✅	✅	✅
6	❌	✅	❌	✅
7	❌	✅	❌	✅
8	❌	❌	✅	✅
9	❌	❌	✅	✅
10	✅	✅	✅	✅

Branch Coverage

Branch Condition	a = 2, b = 3	a = 3, b = 2	a = 2, b = 2	Covered
if (b > a) = True	✅	❌	❌	✅
if (b > a) = False	❌	✅	✅	✅
else-if (b < a) = True	❌	✅	❌	✅
else-if (b < a) = False	❌	❌	✅	✅

Screenshot coverage for Q2:

Problems | Javadoc | Declaration | Coverage Explorer | Test Run Explorer | Coverage

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
lab2	6.1 %	59	914	973
test	6.9 %	36	485	521
(default package)	0.0 %	0	485	485
q2	100.0 %	36	0	36
FunctionTest.java	100.0 %	36	0	36
FunctionTest	100.0 %	36	0	36
main	5.1 %	23	429	452
(default package)	0.0 %	0	429	429
q2	100.0 %	23	0	23
Function.java	100.0 %	23	0	23
Function	100.0 %	23	0	23

Question 3

Statement coverage for Q3:

Problems | Javadoc | Declaration | Console | Coverage Explorer | Test Run Explorer | Coverage

TriclassTest (1) (Feb 10, 2025 12:31:21 AM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
lab2	13.9 %	157	976	1,133
test	16.0 %	99	521	620
(default package)	0.0 %	0	485	485
q2	0.0 %	0	36	36
q3	100.0 %	99	0	99
TriclassTest.java	100.0 %	99	0	99
main	11.3 %	58	455	513
(default package)	0.0 %	0	429	429
q2	0.0 %	0	23	23
q3	95.1 %	58	3	61
Triclass.java	95.1 %	58	3	61

System print messages:

```
Testing started
Test 4 started
Test 4 finished
Test 2 started
Test 2 finished
Test 3 started
Test 3 finished
Test 1 started
Test 1 finished
Testing is finished
```

Coverage for statements in main class:

```java
    public static String classify(int x, int y, int z) {
        // Check if the input values are within the valid range (1 to 10)
        if (x < 1 || x > 10 || y < 1 || y > 10 || z < 1 || z > 10) {
            return "Invalid";
        }

        // Check triangle inequality theorem
        if (x + y <= z || y + z <= x || z + x <= y) {
            return "Invalid";
        }

        // Classify the triangle
        if (x == y && y == z) {
            return "Equilateral";
        } else if (x == y || y == z || z == x) {
            return "Isosceles";
        } else {
            return "Scalene";
        }
    }
}
```

This question demonstrated how coverage helps developers visually identify which statements are properly tested and how code can be optimized to eliminate redundancy. The image above displays green and yellow highlights. Green indicates that a statement is fully covered and frequently executed, while yellow signifies that a statement is only executed in certain test cases. Yellow highlights typically appear in conditional statements, as not all test cases pass through them—some are rejected if they fail the condition. Meanwhile, return statements following conditionals are highlighted in green because every test case that reaches them will always execute the return statement.