RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	891
Course Title	Software Testing and Quality Assurance
Semester/Year	W2023

Instructor	Dr. Reza Samavi

Lab No.	3

Lab Title	Coverage-based Test Design Code Coverage

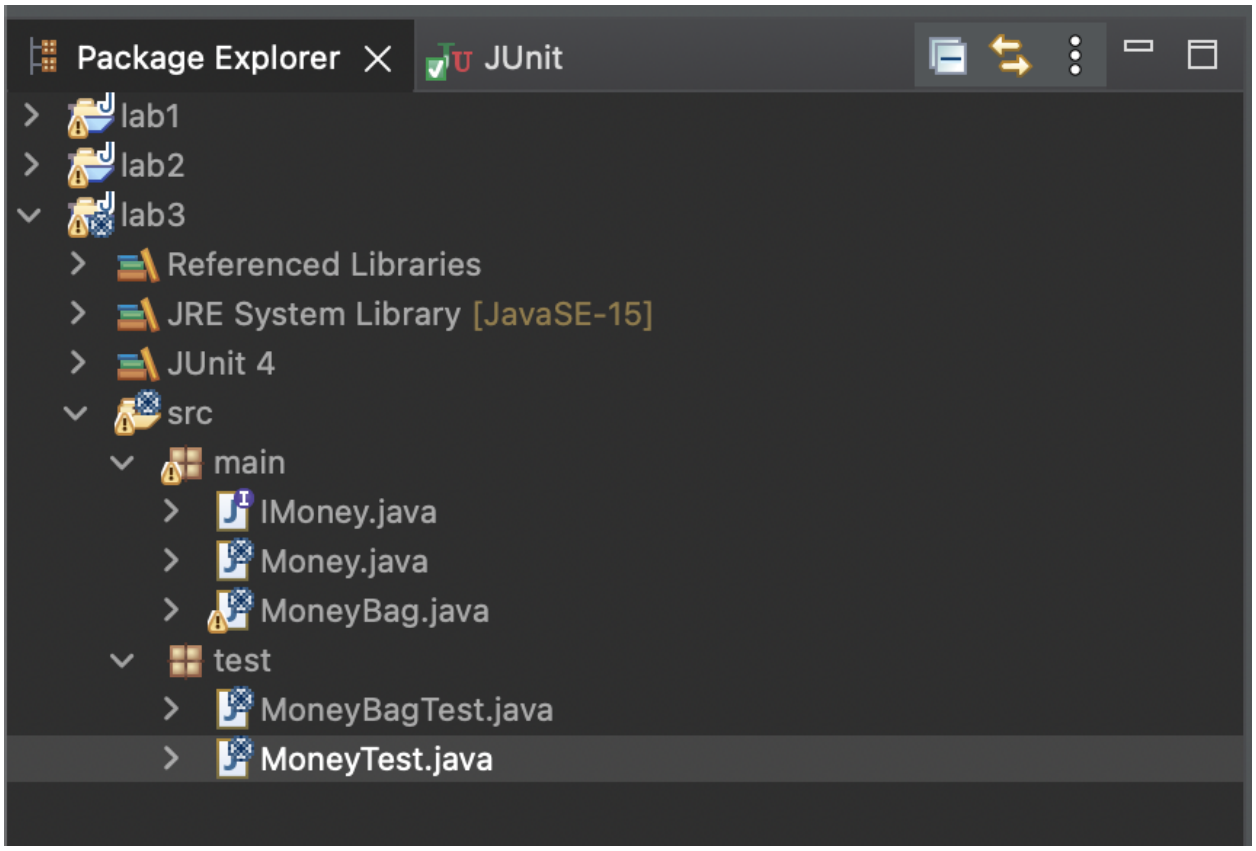
Submission Date	February 13th, 2023
Due Date	February 13th, 2023

Student Name	Student ID	Signature*
Abdulrehman Khan	500968727	A.K.

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work.*

Q1:

Packages, classes, and test classes:

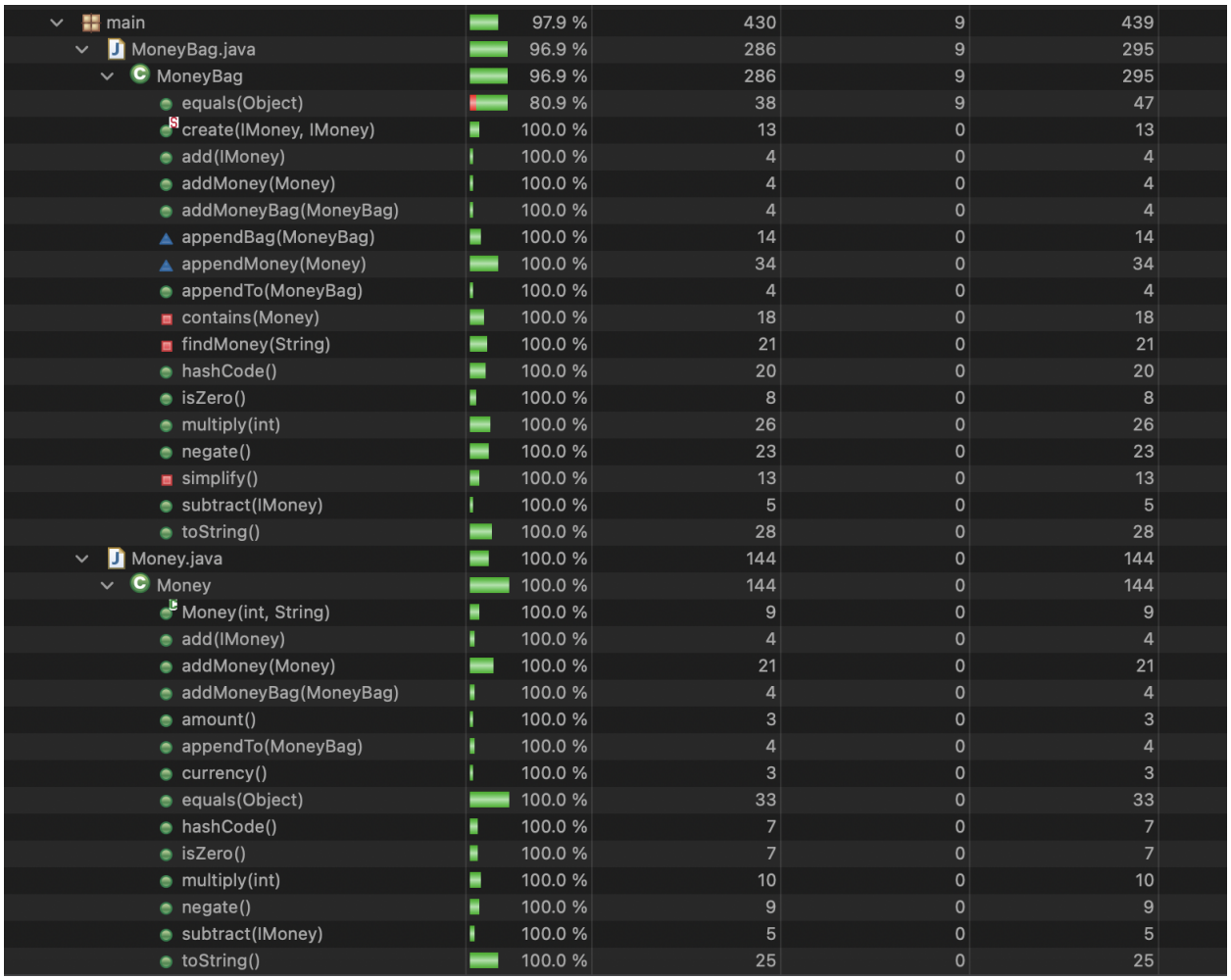


Overview of test coverage results:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
lab3	97.4 %	993	26	1,019
src	97.4 %	993	26	1,019
test	97.1 %	563	17	580
MoneyBagTest.java	96.9 %	523	17	540
MoneyTest.java	100.0 %	40	0	40
main	97.9 %	430	9	439
MoneyBag.java	96.9 %	286	9	295
Money.java	100.0 %	144	0	144

Detailed view of test coverage results:

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
lab3	97.4 %	993	26	1,019
src	97.4 %	993	26	1,019
test	97.1 %	563	17	580
MoneyBagTest.java	96.9 %	523	17	540
MoneyBagTest	96.9 %	523	17	540
testMoneyBagEquals()	89.7 %	70	8	78
testMoneyEquals()	90.2 %	37	4	41
main(String[])	0.0 %	0	3	3
testMoneyHash()	90.9 %	20	2	22
setUp()	100.0 %	43	0	43
tearDown()	100.0 %	5	0	5
testBagMultiply()	100.0 %	32	0	32
testBagNegate()	100.0 %	18	0	18
testBagNotEquals()	100.0 %	18	0	18
testBagSimpleAdd()	100.0 %	20	0	20
testBagSubtract()	100.0 %	20	0	20
testBagSumAdd()	100.0 %	20	0	20
testIsZero()	100.0 %	21	0	21
testMixedSimpleAdd()	100.0 %	14	0	14
testMoneyBagHash()	100.0 %	19	0	19
testNormalize2()	100.0 %	14	0	14
testNormalize3()	100.0 %	25	0	25
testNormalize4()	100.0 %	25	0	25
testPrint()	100.0 %	6	0	6
testSimpleAdd()	100.0 %	14	0	14
testSimpleBagAdd()	100.0 %	20	0	20
testSimpleMultiply()	100.0 %	13	0	13
testSimpleNegate()	100.0 %	12	0	12
testSimpleSubtract()	100.0 %	14	0	14
testSimplify()	100.0 %	20	0	20
MoneyTest.java	100.0 %	40	0	40
MoneyTest	100.0 %	40	0	40
testAdd()	100.0 %	20	0	20
testEquals()	100.0 %	17	0	17
main	97.9 %	430	9	439
MoneyBag.java	96.9 %	286	9	295
MoneyBag	96.9 %	286	9	295
equals(Object)	80.9 %	38	9	47
create(IMoney, IMoney)	100.0 %	13	0	13



main	97.9 %	430	9	439
MoneyBag.java	96.9 %	286	9	295
MoneyBag	96.9 %	286	9	295
equals(Object)	80.9 %	38	9	47
create(IMoney, IMoney)	100.0 %	13	0	13
add(IMoney)	100.0 %	4	0	4
addMoney(Money)	100.0 %	4	0	4
addMoneyBag(MoneyBag)	100.0 %	4	0	4
appendBag(MoneyBag)	100.0 %	14	0	14
appendMoney(Money)	100.0 %	34	0	34
appendTo(MoneyBag)	100.0 %	4	0	4
contains(Money)	100.0 %	18	0	18
findMoney(String)	100.0 %	21	0	21
hashCode()	100.0 %	20	0	20
isZero()	100.0 %	8	0	8
multiply(int)	100.0 %	26	0	26
negate()	100.0 %	23	0	23
simplify()	100.0 %	13	0	13
subtract(IMoney)	100.0 %	5	0	5
toString()	100.0 %	28	0	28
Money.java	100.0 %	144	0	144
Money	100.0 %	144	0	144
Money(int, String)	100.0 %	9	0	9
add(IMoney)	100.0 %	4	0	4
addMoney(Money)	100.0 %	21	0	21
addMoneyBag(MoneyBag)	100.0 %	4	0	4
amount()	100.0 %	3	0	3
appendTo(MoneyBag)	100.0 %	4	0	4
currency()	100.0 %	3	0	3
equals(Object)	100.0 %	33	0	33
hashCode()	100.0 %	7	0	7
isZero()	100.0 %	7	0	7
multiply(int)	100.0 %	10	0	10
negate()	100.0 %	9	0	9
subtract(IMoney)	100.0 %	5	0	5
toString()	100.0 %	25	0	25

As we can observe from both the overview and the detailed view of the clover test coverage results, the overall coverage is at 96.7% which is sufficient (3.3% away from 100%) as we need it to be close to 100% as possible.

Q2:

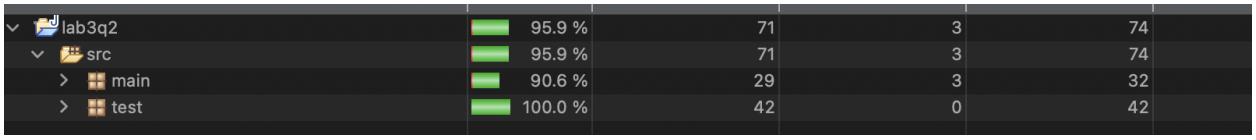
Math.java:

```java
package main;

public class math {

  public static int func(int a, int b) {
    if (b > a) {
      b = b - a;
      System.out.println(b);
    } else if (a > b) {
      b = a - b;
      System.out.println(b);
    } else {
      b = 0;
      System.out.println(0);
    }
    return b;
  }

}
```

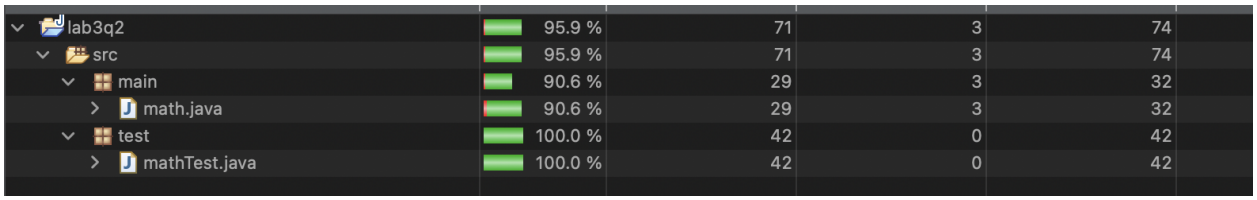
MathTest.java:

```java
package test;
import main.math;

public class mathTest {



    @Test
    // b > a
    public void testMath() {
        int a1 = 2;
        int b1 = 3;
        int actual = math.func(a1,b1);
        assertEquals (1, actual);
    }


    @Test
    // a > b
    public void testMath2() {
        int a2 = 3;
        int b2 = 2;
        int actual = math.func(a2, b2);
        assertEquals (1, actual);
    }

    @Test
    // a = b and other cases
    public void testMath3() {
        int a3 = 1;
        int b3 = 1;
        int actual = math.func(a3, b3);
        assertEquals (0, actual);

    }
}
```

Overview of test coverage results:



Detailed view of test coverage results:



Statement Coverage:

Total number of statements: 10
Executed statements overall: 8
Statement coverage overall: 8/10 = 80%

Total number of statements: 10
Executed statements if b > a : 5
Statement coverage: 5/10 = 50%

Total number of statements: 10
Executed statements if a > b : 7
Statement coverage: 7/10 = 70%

Total number of statements: 10
Executed statements if a = b : 5
Statement coverage: 4/10 = 40%

Branch Coverage:

Test case	Value of b	Value of a	output	Decision coverage	Branch coverage
1	3	2	1	33.33%	33.33%
2	2	3	1	33.33%	67%

3	1	1	0	33.33%	100%