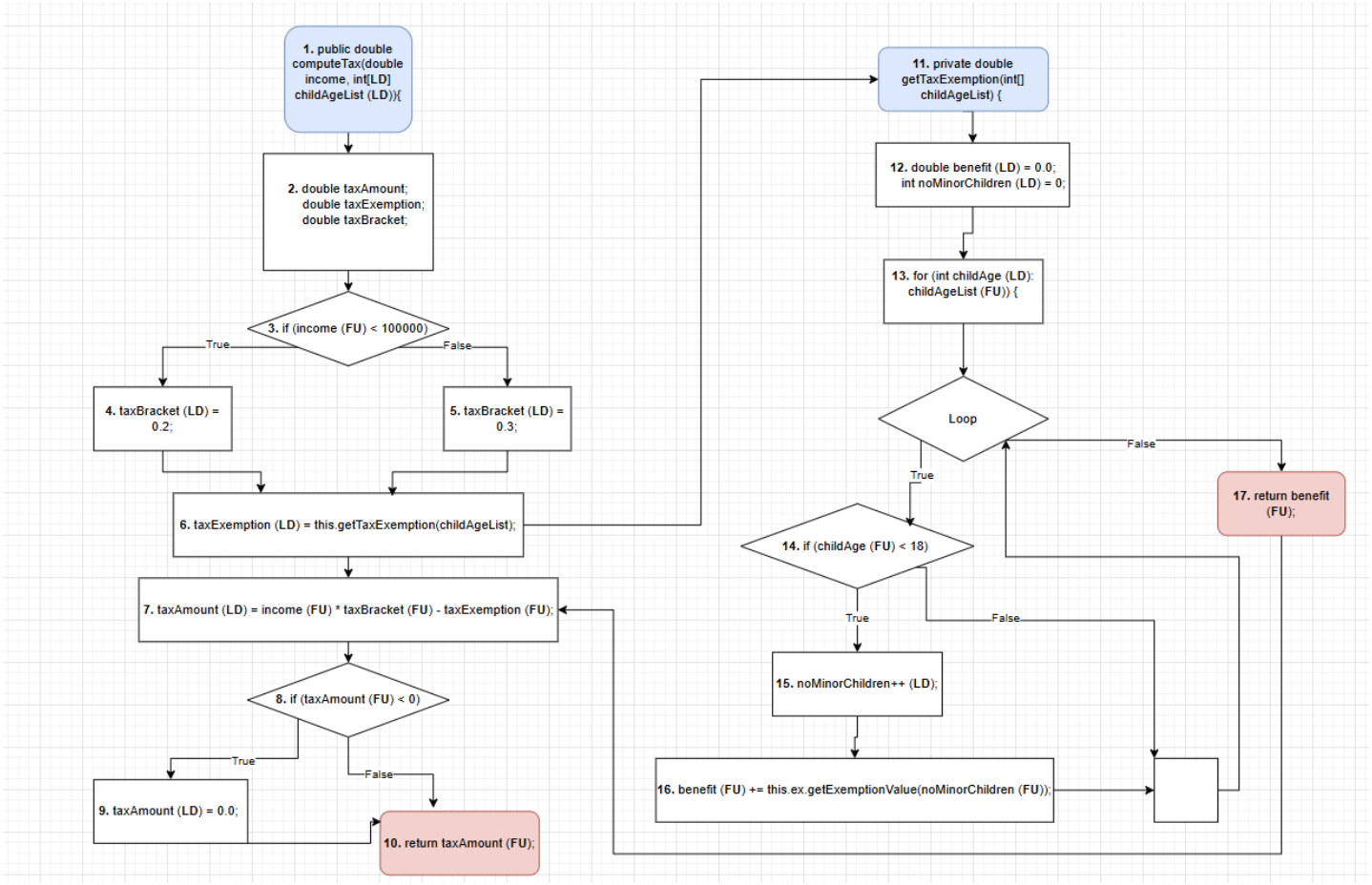
Assignment 2

Oskar Sundfors

Task 1

1.



LD = Last Def

FU = First Use

Red = Final nodes

Blue = First nodes

2.

childAgeList:

1 -> 13

[1,2,3,4,6,11,12,13]

Or

[1,2,3,5,6,11,12,13]

taxExemption:

6 -> 7

[6,11,12,13,14,15,16,13,17,7]

Or

[6,11,12,13,14,13,17,7]

3.

childAgeList:

1 -> 13

[1,2,3,4,6,11,12,13]

And

[1,2,3,5,6,11,12,13]

taxExemption:

6 -> 7

[6,11,12,13,14,15,16,13,17,7]

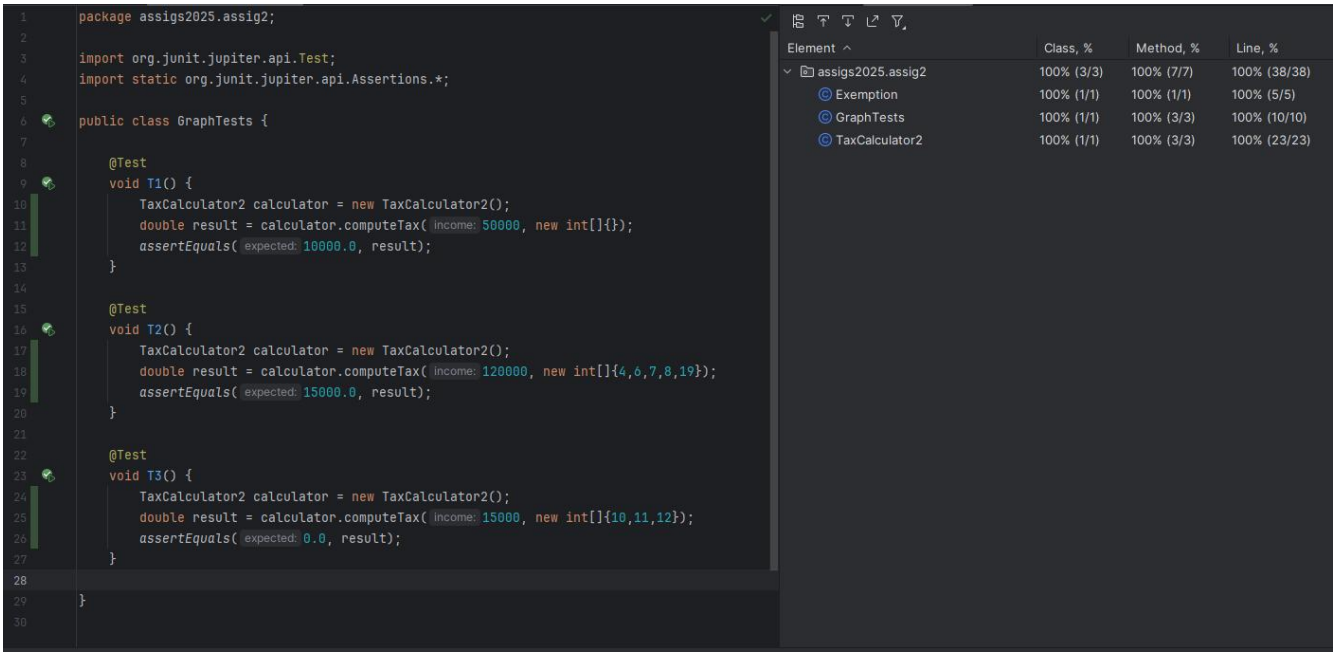
And

[6,11,12,13,14,13,17,7]

4.

Test	Test Path	Input	Expected
T1	[1,2,3,4,6,11,12,13,17,7,8,10]	income = 50000 childAgeList = []	50000 * 0.2 = 10000
T2	[1,2,3,5,6,11,12,13,14,15(x4),16,13,17,7,8,10]	income = 120000 childAgeList = [4, 6, 7, 8, 19]	120000 * 0.3 - (21000) = 15000
T3	[1,2,3,4,6,11,12,13,14,15(x3),16,13,17,7,8,9,10]	income = 15000 childAgeList = [10, 11, 12]	15000 * 0.2 - 15000 = -12000 = 0

5.



```java
package assigs2025.assig2;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class GraphTests {

    @Test
    void T1() {
        TaxCalculator2 calculator = new TaxCalculator2();
        double result = calculator.computeTax( income: 50000, new int[]{});
        assertEquals( expected: 10000.0, result);
    }

    @Test
    void T2() {
        TaxCalculator2 calculator = new TaxCalculator2();
        double result = calculator.computeTax( income: 120000, new int[]{4,6,7,8,19});
        assertEquals( expected: 15000.0, result);
    }

    @Test
    void T3() {
        TaxCalculator2 calculator = new TaxCalculator2();
        double result = calculator.computeTax( income: 15000, new int[]{10,11,12});
        assertEquals( expected: 0.0, result);
    }

}
```

Element	Class, %	Method, %	Line, %
assigs2025.assig2	100% (3/3)	100% (7/7)	100% (38/38)
Exemption	100% (1/1)	100% (1/1)	100% (5/5)
GraphTests	100% (1/1)	100% (3/3)	100% (10/10)
TaxCalculator2	100% (1/1)	100% (3/3)	100% (23/23)

Task 2

The mutators killed were on line 9, 39, 56 in TaxCalculator2. Mutators killed on line 18 in Exemption.

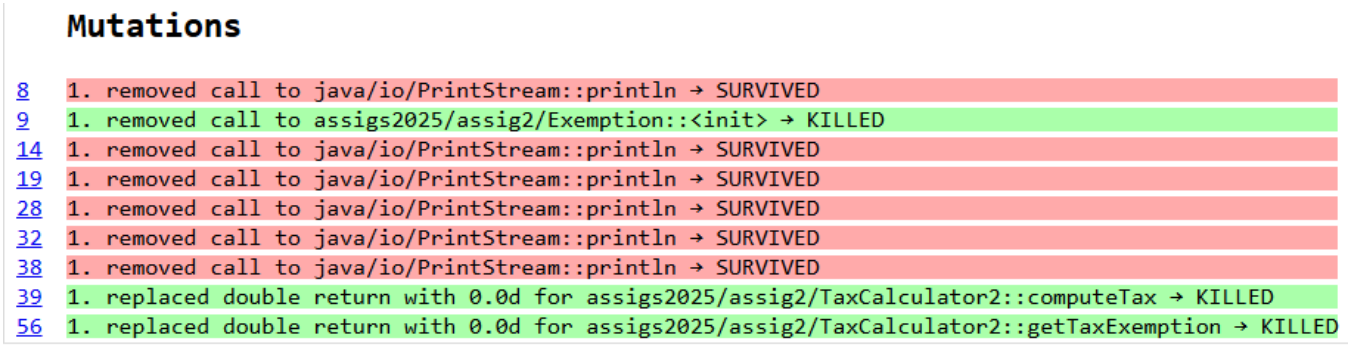
Line 9, line 56 and line 18 in Exemption.java: test T3 was used with input values income: 15 000, children: 10,11,12, expected value: 0

Line 39: test T2 was used with input values income: 120 000, children: 4,6,7,8,19, expected value: 15 000

Test T1 did not kill any mutants with input values income: 50 000, children: [empty], expected value: 10 000.

The mutators that were applied are shown in the screenshots below.

The surviving mutants are from print statements, and therefore have no impact on return values or the functionality of the code.



Mutations

8	1. removed call to java/io/PrintStream::println → SURVIVED
9	1. removed call to assigs2025/assig2/Exemption::<init> → KILLED
14	1. removed call to java/io/PrintStream::println → SURVIVED
19	1. removed call to java/io/PrintStream::println → SURVIVED
28	1. removed call to java/io/PrintStream::println → SURVIVED
32	1. removed call to java/io/PrintStream::println → SURVIVED
38	1. removed call to java/io/PrintStream::println → SURVIVED
39	1. replaced double return with 0.0d for assigs2025/assig2/TaxCalculator2::computeTax → KILLED
56	1. replaced double return with 0.0d for assigs2025/assig2/TaxCalculator2::getTaxExemption → KILLED

Active mutators

- CONSTRUCTOR_CALLS
- EMPTY_RETURNS
- FALSE_RETURNS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Mutations

18 1. replaced double return with 0.0d for assigs2025/assig2/Exemption::getExemptionValue → KILLED

Active mutators

- CONSTRUCTOR_CALLS
- EMPTY_RETURNS
- FALSE_RETURNS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

Below is the mutation coverage of the tests. The surviving mutants are print statements.

Pit Test Coverage Report

Package Summary

assigs2025.assig2

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	100%	29/29	40%	4/10	40%	4/10

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
Exemption.java	100%	5/5	100%	1/1	100%	1/1
TaxCalculator2.java	100%	24/24	33%	3/9	33%	3/9

Task 4

In Exemption.java, when running pit with only Mockito tests, all mutations survive due to the mocks being used instead of the real code.

In line 9 in TaxCalculator2, the initialization of the exemption also survived due to Mockito injecting the mock (the real Exemption is never tested).

Below the two images show the mutators that were killed/survived, and the mutation score with only MockitoTests being used.

Pit Test Coverage Report

Package Summary

assigs2025.assig2

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	83%	24/29	20%	2/10	22%	2/9

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
Exemption.java	20%	1/5	0%	0/1	100%	0/0
TaxCalculator2.java	96%	23/24	22%	2/9	22%	2/9

Mutations

```
8   1. removed call to java/io/PrintStream::println → SURVIVED
9   1. removed call to assigs2025/assig2/Exemption::<init> → SURVIVED
14  1. removed call to java/io/PrintStream::println → SURVIVED
19  1. removed call to java/io/PrintStream::println → SURVIVED
28  1. removed call to java/io/PrintStream::println → SURVIVED
32  1. removed call to java/io/PrintStream::println → SURVIVED
38  1. removed call to java/io/PrintStream::println → SURVIVED
39  1. replaced double return with 0.0d for assigs2025/assig2/TaxCalculator2::computeTax → KILLED
56  1. replaced double return with 0.0d for assigs2025/assig2/TaxCalculator2::getTaxExemption → KILLED
```

Active mutators

- CONSTRUCTOR_CALLS
- EMPTY_RETURNS
- FALSE_RETURNS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS