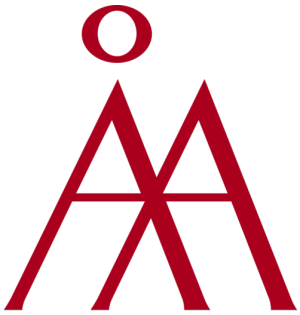


Åbo Akademi University

Software Testing

Assignment 2

Luis Araújo(2004624)

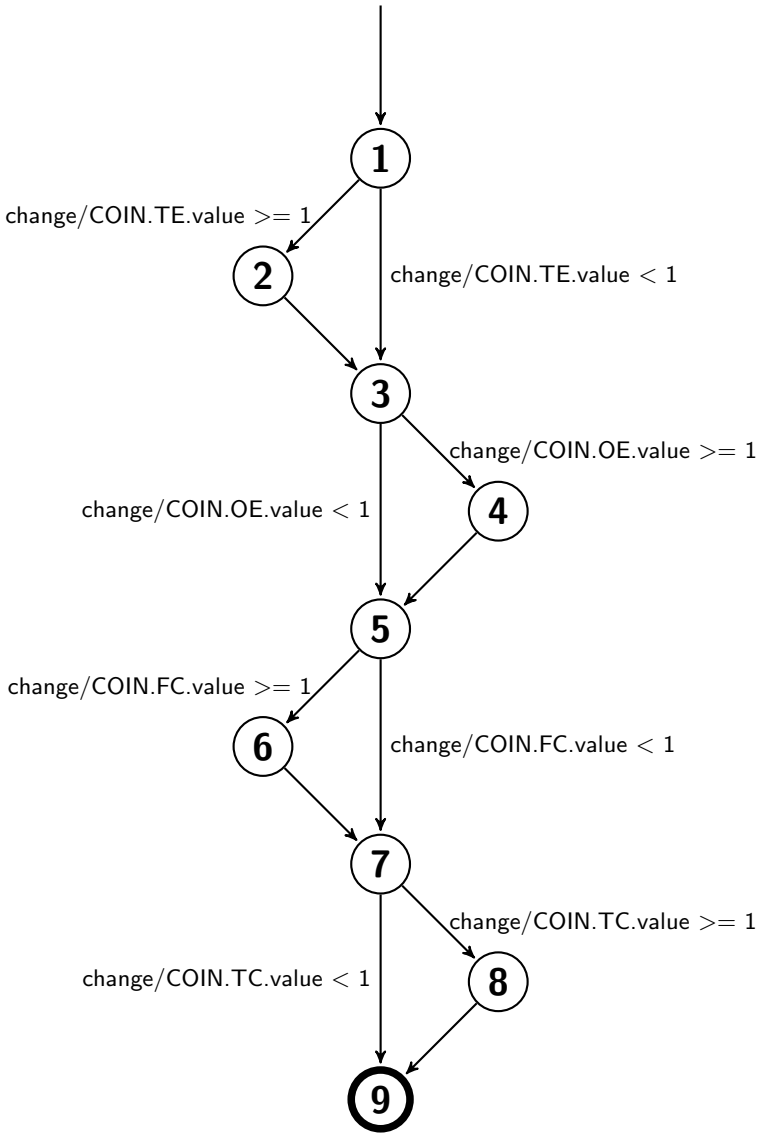
May 03, 2021

Contents

Chapter 1

Task 1

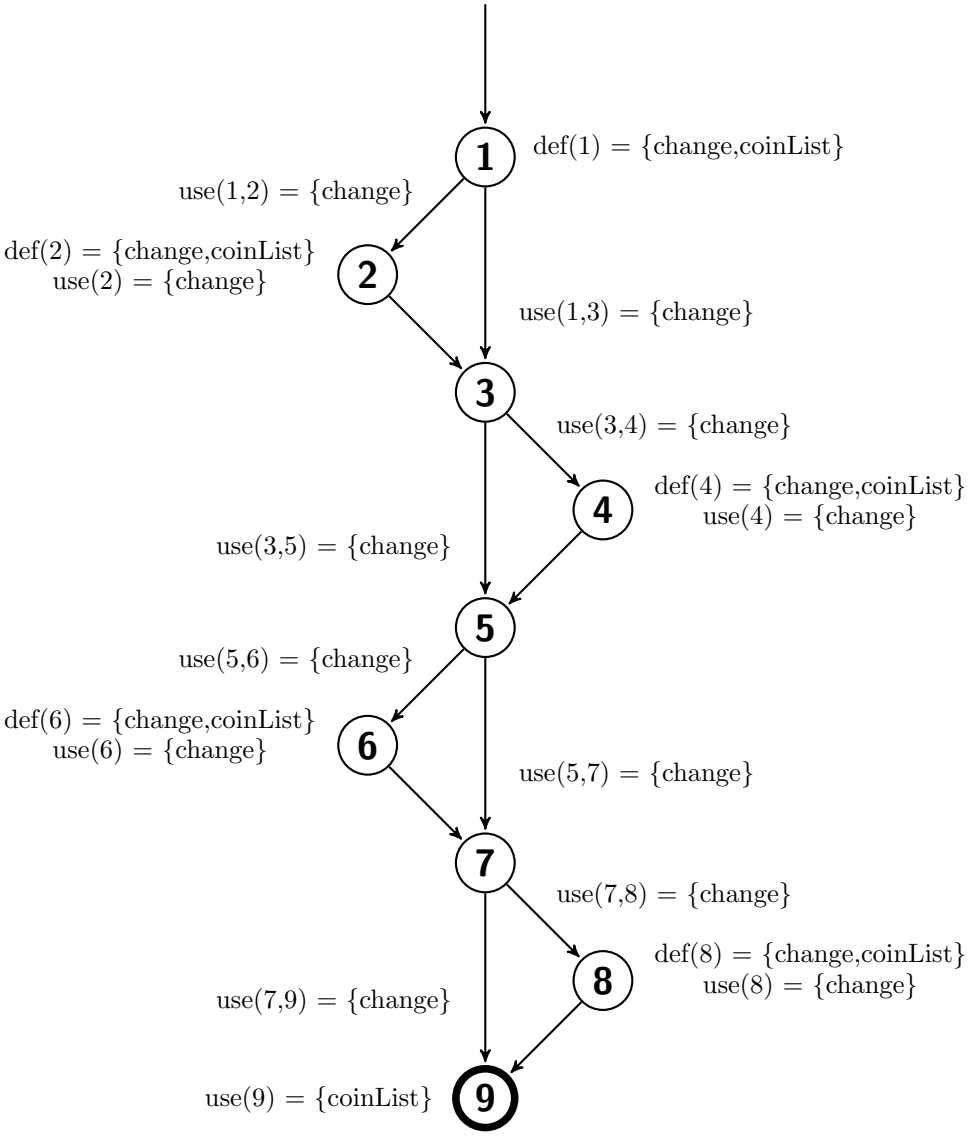
1.1 Create the CFG of the method



1.2 Annotate CFG with defs and uses for variables change and coinList

CoinList	
Defs	Uses
1,2,4,6,8	9

change	
Defs	Uses
1,2,4,6,8	2,4,6,8,(1,2),(1,3),(3,4),(3,5),(5,6),(5,7),(7,8),(7,9)



1.3 List all predicates and their reachability conditions

1.4 List Test requirements for Node, Edge and Edge pair coverage

Test requirements		
Node coverage	Edge coverage	Edge-Pair coverage
1,2,3,4,5,6,7,8,9	(1,2),(1,3),(2,3),(3,4),(3,5), (4,5),(5,6),(5,7),(6,7),(7,8), (7,9),(8,9)	(1,2,3),(1,3,4),(1,3,5),(2,3,4), (2,3,5),(3,4,5),(3,5,6),(3,5,7), (4,5,6),(4,5,7),(5,6,7),(5,7,8), (5,7,9),(6,7,8),(6,7,9),(7,8,9)

1.5 List test requirements for All-Defs coverage for variable change

All-Defs coverage	
change	[1,2,3,5,7,9] [1,3,4,5,7,9] [1,3,5,6,7,8,9]

1.6 List Test requirements for All Use coverage for variable coinList

All-Use coverage	
coinList	[1,3,5,7,9] [1,2,3,5,7,9] [1,3,4,5,7,9] [1,3,5,7,8,9]

1.7 Select one of the predicate and list test requirements for predicate coverage.

1.8 Create test paths to cover all test requirements specified above (first remove duplicated test requirements)

Test paths			
Coverage type	Tests paths	Individual Coverage Level	Total Coverage Level
Node Coverage	[1,2,3,4,5,6,7,8,9]	9/9 = 100%	9/9 = 100%
Edge Coverage	[1,3,4,5,7,8,9] [1,2,3,5,6,7,9]	6/12 = 50% 6/12 = 50%	6/12 = 50% 12/12 = 100%
Edge-Pair Coverage	[1,2,3,5,7,8,9] [1,3,5,6,7,9] [1,3,4,5,7,9] [1,2,3,4,5,6,7,8,9]	5/16 = 31% 4/16 = 25% 4/16 = 25% 3/16 = 19%	5/16 = 31% 9/16 = 56% 13/16 = 81% 16/16 = 100%

1.9 Create test inputs for each test path and assign the expected output

Input & Output			
Coverage type	Test paths	Input	Expected Output
Node Coverage	[1,2,3,4,5,6,7,8,9]	change = 3.7	coinList = [1,1,1,1]

```
@Test
void testCalculateReturningCoins() {
    assertArrayEquals(new int[]{1,1,1,1}, vm.calculateReturningCoins(3.8));
}
```

Figure 1.1: Node Coverage

Input & Output			
Coverage type	Test paths	Input	Expected Output
Edge Coverage	[1,3,4,5,7,8,9]	change = 1.2	coinList = [0,1,0,1]
	[1,2,3,5,6,7,9]	change = 2.5	coinList = [1,0,1,0]

```
@Test
void testCalculateReturningCoins() {
    assertArrayEquals(new int[]{0,1,0,1}, vm.calculateReturningCoins(1.2));
    assertArrayEquals(new int[]{1,0,1,0}, vm.calculateReturningCoins(2.5));
}
```

Figure 1.2: Edge Coverage

Input & Output			
Coverage type	Test paths	Input	Expected Output
Edge-Pair Coverage	[1,2,3,5,7,8,9]	change = 2.2	coinList = [1,0,0,1]
	[1,3,5,6,7,9]	change = 0.5	coinList = [0,0,1,0]
	[1,3,4,5,7,9]	change = 1	coinList = [0,1,0,0]
	[1,2,3,4,5,6,7,8,9]	change = 3.7	coinList = [1,1,1,1]

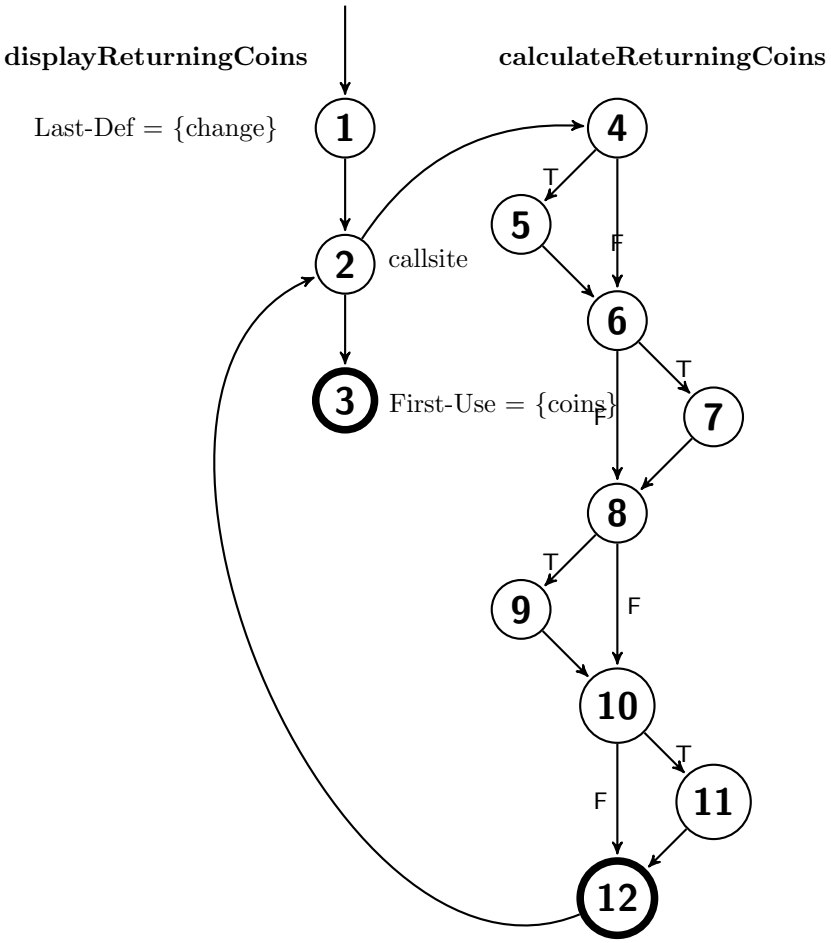
```
@Test
void testCalculateReturningCoins() {
    assertArrayEquals(new int[]{1,0,0,1}, vm.calculateReturningCoins(2.2));
    assertArrayEquals(new int[]{0,0,1,0}, vm.calculateReturningCoins(0.5));
    assertArrayEquals(new int[]{0,1,0,0}, vm.calculateReturningCoins(1));
    assertArrayEquals(new int[]{1,1,1,1}, vm.calculateReturningCoins(3.7));
}
```

Figure 1.3: Edge-Pair Coverage

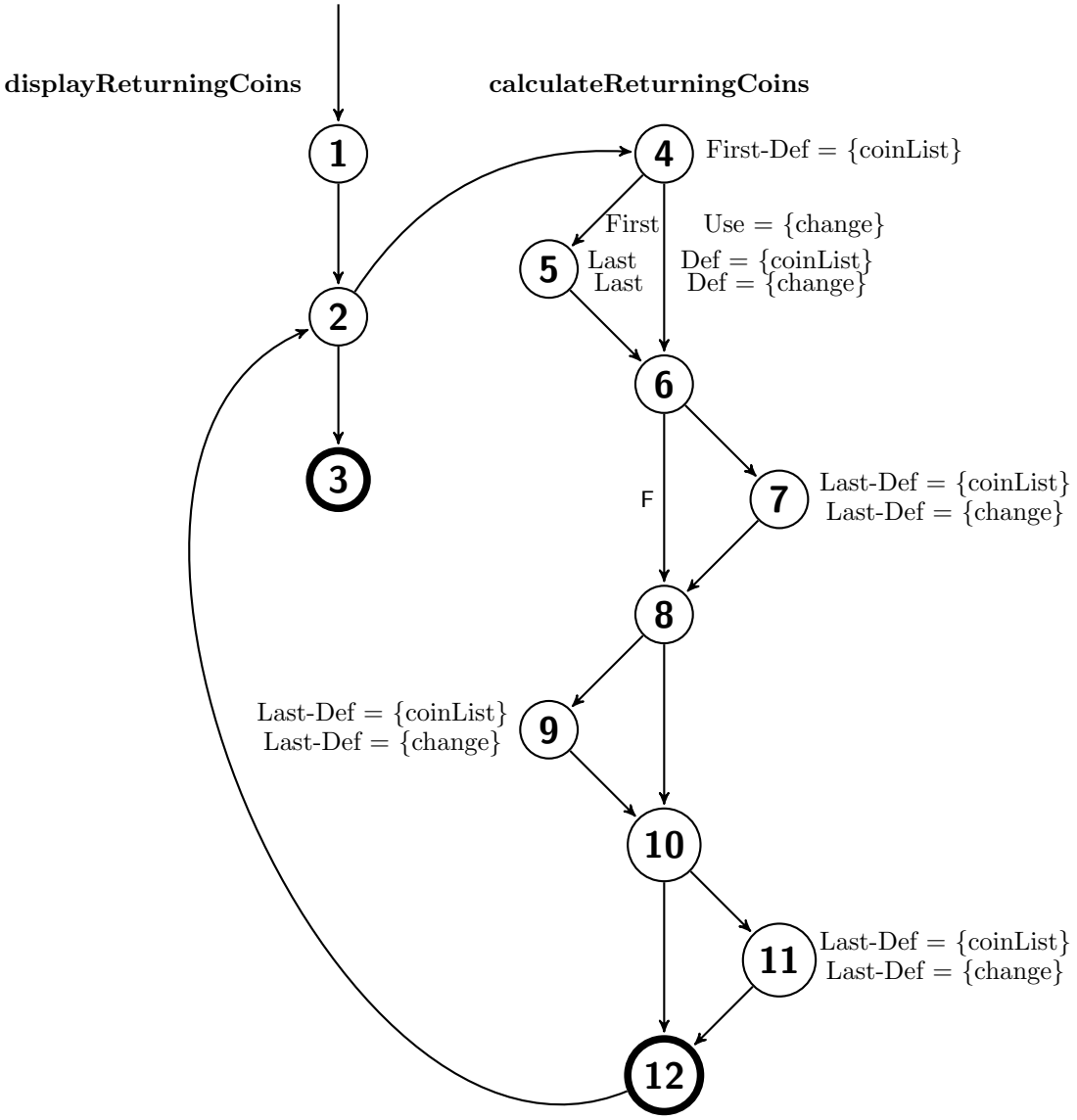
Chapter 2

Task 2

2.1 Create CFG for displayReturningCoins(double change) and annotate it with last defs and first uses for the call and return parameters



2.2 Annotate the CFG of calculateReturningCoins(double change) defined in Task 1 with last defs and first uses for call parameters



2.3 List test requirements for All-Coupling-Defs coverage for the direct call

All-Coupling-Def coverage
(1,change) − > ((4,5),change)
(5,coinList) − > (3,coin)

2.4 List test requirements for All-Coupling-Use coverage for the return

All-Coupling-Use coverage
(1,change) − > ((4,5),change)
(5,coinList) − > (3,coin)
(7,coinList) − > (3,coin)
(9,coinList) − > (3,coin)
(11,coinList) − > (3,coin)

2.5 List test paths to satisfy all test requirements above

Test paths		
Test paths	Individual Coverage Level	Total Coverage Level
[1,2,4,5,6,7,8,9,10,11,12,2,3]	7/7 = 100%	7/7 = 100%

2.6 Create test inputs for each test path and the expected output

Input & Output		
Test paths	Input	Expected Output
[1,2,4,5,6,7,8,9,10,11,12,2,3]	change = 3.7	Your Change is 1 x 2Euro 1 x 1Euro 1 x 50Cent 1 x 20Cent

```java
@Test
void testDisplayReturningCoins() {
    String NLC = System.getProperty("line.separator");

    PrintStream oldout = System.out;
    ByteArrayOutputStream newout = new ByteArrayOutputStream();
    System.setOut(new PrintStream(newout));

    vm.displayReturningCoins(3.8);

    String replay = newout.toString();
    System.setOut(oldout);

    assertTrue(replay.contains("Your Change is " + NLC + "\t1 x 2Euro" + NLC + "\t1 x 1Euro" + NLC + "\t1 x 50Cent" + NLC + "\t1 x 20Cent" + NLC));
}
```

Figure 2.1: Test

Chapter 3

Task 3

3.1 Write and execute JunitTests for the tests identified at Tasks 1.9 and 2.6 in two separate test classes. Name your tests according to the ID of the test paths

```
@Test
void testCalculateReturningCoins_Node() {
    assertArrayEquals(new int[]{1,1,1,1}, vm.calculateReturningCoins(3.8));
}

@Test
void testCalculateReturningCoins_Edge() {
    assertArrayEquals(new int[]{1,0,0,1}, vm.calculateReturningCoins(2.2));
    assertArrayEquals(new int[]{0,0,1,0}, vm.calculateReturningCoins(0.5));
}

@Test
void testCalculateReturningCoins_Edge_Pair() {
    assertArrayEquals(new int[]{1,0,0,1}, vm.calculateReturningCoins(2.2));
    assertArrayEquals(new int[]{0,0,1,0}, vm.calculateReturningCoins(0.5));
    assertArrayEquals(new int[]{0,1,0,0}, vm.calculateReturningCoins(1));
    assertArrayEquals(new int[]{1,1,1,1}, vm.calculateReturningCoins(3.7));
}

@Test
void testDisplayReturningCoins() {
    String NLC = System.getProperty("line.separator");

    PrintStream oldout = System.out;
    ByteArrayOutputStream newout = new ByteArrayOutputStream();
    System.setOut(new PrintStream(newout));

    vm.displayReturningCoins(3.8);

    String replay = newout.toString();
    System.setOut(oldout);
System.out.println(replay);
    assertTrue(replay.contains("Your Change is " + NLC + "\t1 x 2Euro" + NLC + "\t1 x 1Euro" + NLC + "\t1 x 50Cent" + NLC + "\t1 x 20Cent" + NLC));
}
```

Figure 3.1: Tests

Chapter 4

Task 4

4.1 The output of tasks T1.1 - T1.9 and T2.1 - T2.6

The output of the asked tasks are already included in the report.

4.2 Test report on how many tests passed and failed (screen shot from Junit test runner is enough)

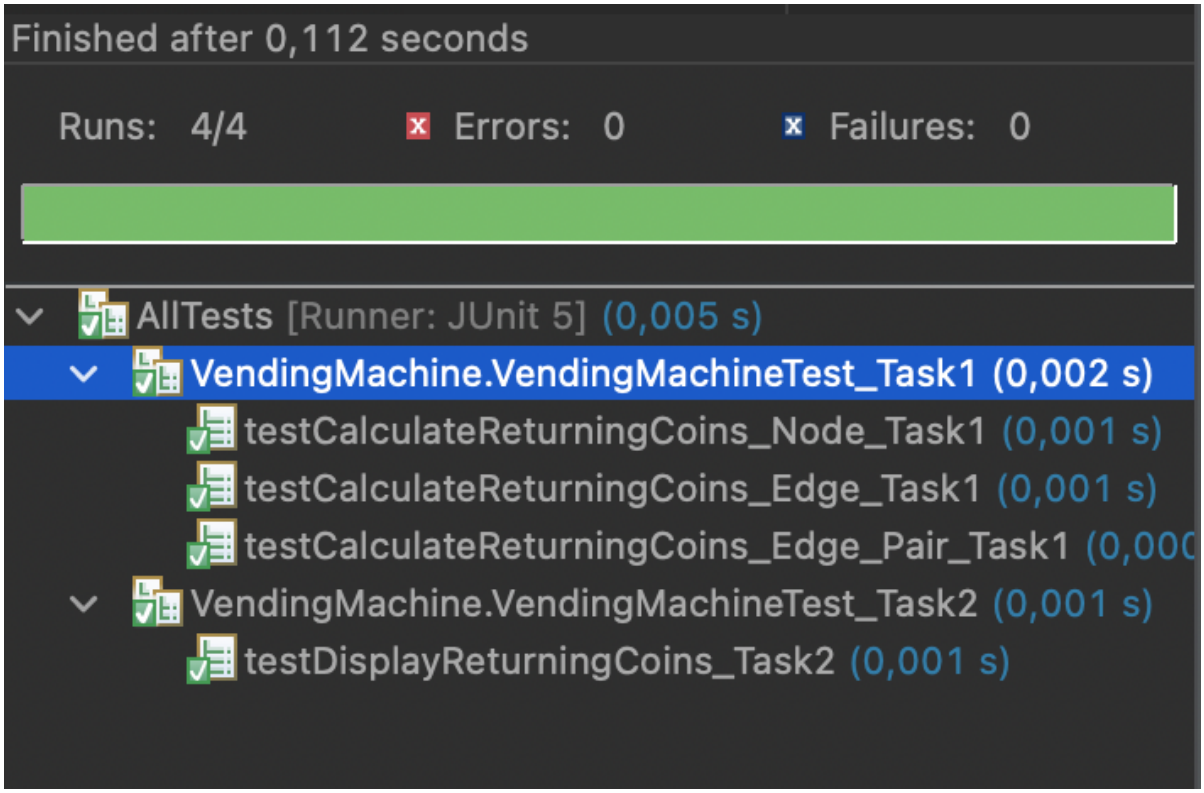


Figure 4.1: Tests

4.3 Report on coverage level for the 2 methods tested (by the entire test suit).

Figure 4.2: Test coverage