

Software Testing Practical Session Report

Younes Menfalouti

November 26, 2025

1 Introduction

This report documents the practical session on software testing, focusing on unit testing with JUnit in IntelliJ IDEA. The session involved implementing a Calculator class, a TemperatureRegulator class, and writing comprehensive unit tests using the Arrange-Act-Assert pattern.

2 Exercise 1: Calculator

2.1 Implementation

The Calculator class provides basic arithmetic operations.

```
public class Calculator {  
    public int add(int a, int b) {  
        return a + b;  
    }  
    public int sub(int a, int b) {  
        return a - b;  
    }  
    public int mul(int a, int b) {  
        return a * b;  
    }  
    public int div(int a, int b) {  
        return a / b;  
    }  
}
```

2.2 Unit Tests

The CalculatorTest class contains unit tests for the Calculator.

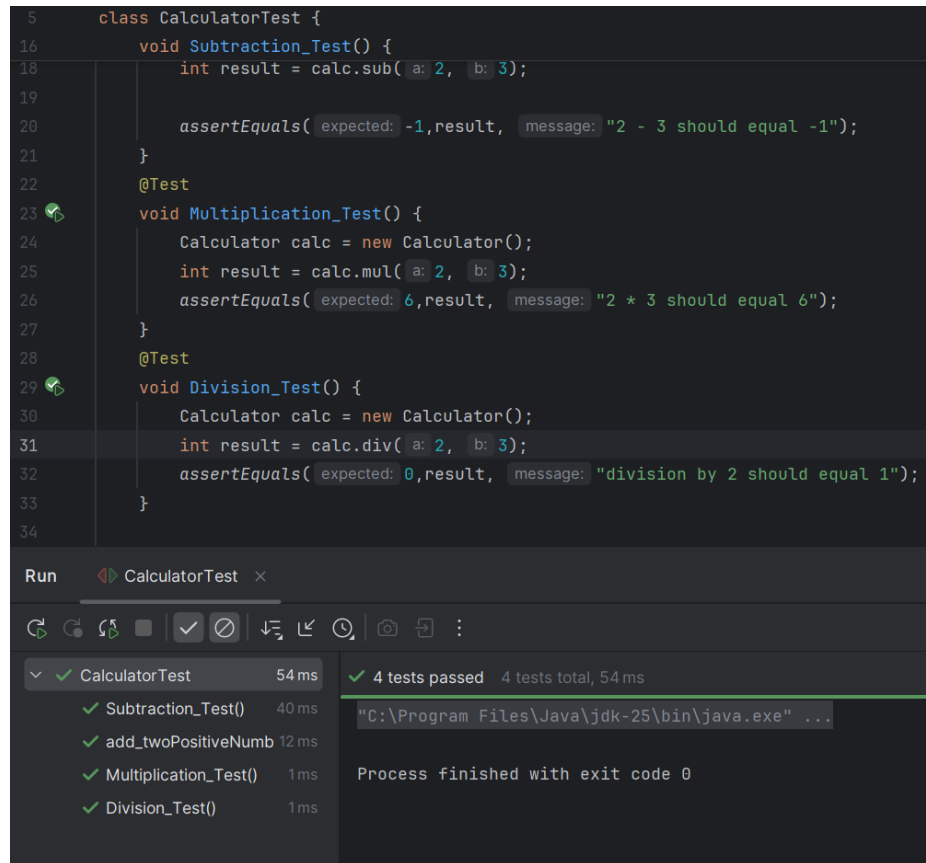
```
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.*;
```

```

class CalculatorTest {
    @Test
    void add_twoPositiveNumbers_shouldReturnSum() {
        Calculator calc = new Calculator();
        int result = calc.add(2, 3);
        assertEquals(5, result, "2 + 3 should equal 5");
    }
    @Test
    void Subtraction_Test() {
        Calculator calc = new Calculator();
        int result = calc.sub(2, 3);
        assertEquals(-1, result, "2 - 3 should equal -1");
    }
    @Test
    void Multiplication_Test() {
        Calculator calc = new Calculator();
        int result = calc.mul(2, 3);
        assertEquals(6, result, "2 * 3 should equal 6");
    }
    @Test
    void Division_Test() {
        Calculator calc = new Calculator();
        int result = calc.div(2, 3);
        assertEquals(0, result, "division by 2 should equal 1");
    }
}

```

2.3 Test Execution



The screenshot displays an IDE with a Java class `CalculatorTest` and its execution results. The code defines three test methods: `Subtraction_Test`, `Multiplication_Test`, and `Division_Test`. Each method uses `assertEquals` to verify the output of a `Calculator` instance. The `Division_Test` method has a message that says "division by 2 should equal 1".

```
5 class CalculatorTest {
16     void Subtraction_Test() {
18         int result = calc.sub( a: 2, b: 3);
19
20         assertEquals( expected: -1,result, message: "2 - 3 should equal -1");
21     }
22     @Test
23     void Multiplication_Test() {
24         Calculator calc = new Calculator();
25         int result = calc.mul( a: 2, b: 3);
26         assertEquals( expected: 6,result, message: "2 * 3 should equal 6");
27     }
28     @Test
29     void Division_Test() {
30         Calculator calc = new Calculator();
31         int result = calc.div( a: 2, b: 3);
32         assertEquals( expected: 0,result, message: "division by 2 should equal 1");
33     }
34 }
```

The Run window shows the execution of `CalculatorTest`. It lists four tests: `Subtraction_Test()` (40 ms), `add_twoPositiveNumb` (12 ms), `Multiplication_Test()` (1 ms), and `Division_Test()` (1 ms). All tests passed. The output pane shows the command `"C:\Program Files\Java\jdk-25\bin\java.exe" ...` and the message `Process finished with exit code 0`.

Here, the add method was temporarily modified to return `a - b` instead of `a + b`.

```
1 import org.junit.jupiter.api.Test ;
2 import static org.junit.jupiter.api.Assertions.*;
3
4
5 class CalculatorTest {
6     @Test
7     void add_twoPositiveNumbers_shouldReturnSum() {
8
9         Calculator calc = new Calculator();
10        int result = calc.add( a: 2, b: 3);
11
12        assertEquals( expected: 5,result, message: "2 + 3 should equal 5");
13    }
14
15    @Test
16    void Subtraction_Test() {
17        Calculator calc = new Calculator();
18        int result = calc.sub( a: 2, b: 3);
19    }
20 }
```

Run CalculatorTest

65 ms 1 test failed, 3 passed 4 tests total, 65 ms

Subtraction_Test() 37 ms
add_twoPositiveNumb 27 ms
Multiplication_Test() 1 ms
Division_Test()

"C:\Program Files\Java\jdk-25\bin\java.exe" ...
org.opentest4j.AssertionFailedError: 2 + 3 should equal 5 ==>
Expected :5
Actual :-1
<Click to see difference>

Running the add test again results in failure, as the expected sum (5) does not match the incorrect subtraction result (-1).

3 Exercise 2: TemperatureRegulator

3.1 Implementation

The TemperatureRegulator class computes the action based on current and target temperatures.

```
public class TemperatureRegulator {

    public enum Action { HEAT, COOL, STANDBY }

    public Action compute(double current, double target) {
        final double TOL = 0.5;

        double diff = current - target;

        if (diff < -TOL) {
```

```

        return Action.HEAT;
    } else if (diff > TOL) {
        return Action.COOL;
    } else {
        return Action.STANDBY;
    }
}
}

```

3.2 Unit Tests

The `TemperatureRegulatorTest` class contains unit tests for the `TemperatureRegulator`.

```

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

class TemperatureRegulatorTest {

    @Test
    void compute_currentMuchLowerThanTarget_shouldReturnHeat() {
        TemperatureRegulator regulator = new TemperatureRegulator();
        double current = 20.0;
        double target = 22.0;
        TemperatureRegulator.Action result = regulator.compute(current, target);
        assertEquals(TemperatureRegulator.Action.HEAT, result);
    }

    @Test
    void compute_currentMuchHigherThanTarget_shouldReturnCool() {
        TemperatureRegulator regulator = new TemperatureRegulator();
        double current = 22.0;
        double target = 20.0;
        TemperatureRegulator.Action result = regulator.compute(current, target);
        assertEquals(TemperatureRegulator.Action.COOL, result);
    }

    @Test
    void compute_currentWithinTolerance_shouldReturnStandby() {
        TemperatureRegulator regulator = new TemperatureRegulator();
        double current = 21.0;
        double target = 21.0;
        TemperatureRegulator.Action result = regulator.compute(current, target);
        assertEquals(TemperatureRegulator.Action.STANDBY, result);
    }
}

```

```

@Test
void compute_currentSlightlyHigherThanTarget_shouldReturnStandby() {
    TemperatureRegulator regulator = new TemperatureRegulator();
    double current = 20.7;
    double target = 21.0;
    TemperatureRegulator.Action result = regulator.compute(current, target);
    assertEquals(TemperatureRegulator.Action.STANDBY, result);
}

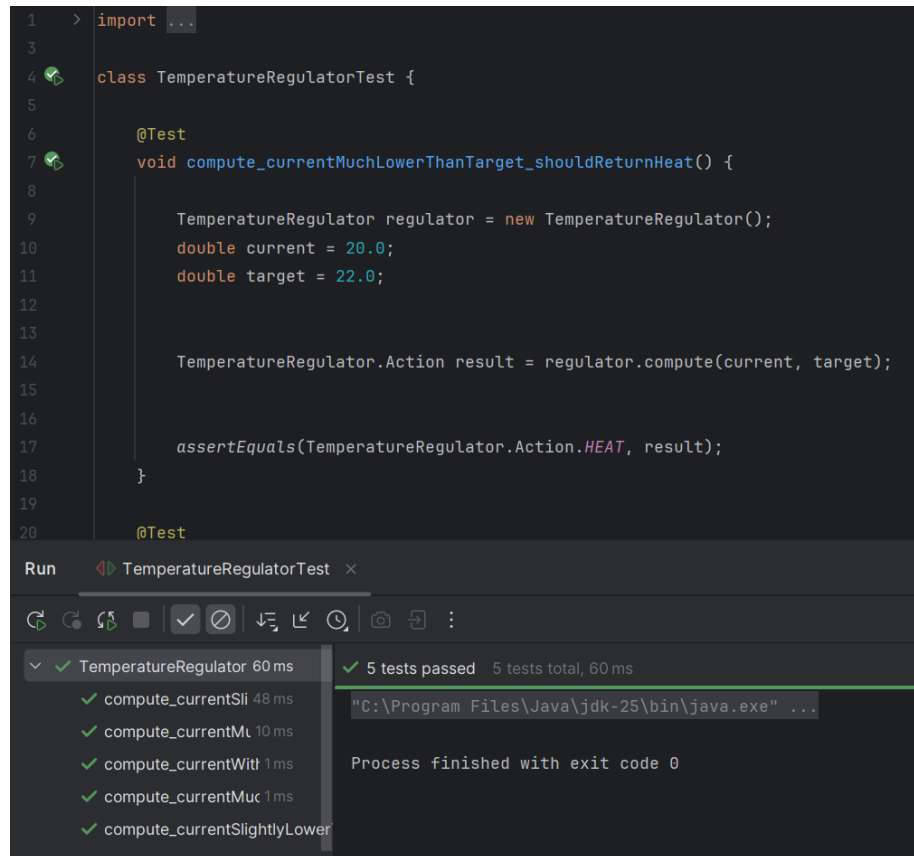
@Test
void compute_currentSlightlyHigherThanTarget_shouldReturnStandby()
    TemperatureRegulator regulator = new TemperatureRegulator();
    double current = 20.7;
    double target = 21.0;
    TemperatureRegulator.Action result = regulator.compute(current, target);
    assertEquals(TemperatureRegulator.Action.STANDBY, result);
}
}

```

3.3 Analysis

The test results indicate that the TemperatureRegulator class behaves according to the specification. The compute method correctly determines the action based on the temperature difference and tolerance: - Returns HEAT when current temperature is more than 0.5 degrees below the target. - Returns COOL when current temperature is more than 0.5 degrees above the target. - Returns STANDBY when the difference is within the tolerance (± 0.5 degrees).

3.4 Test Execution



```
1  > import ...
3
4  class TemperatureRegulatorTest {
5
6      @Test
7      void compute_currentMuchLowerThanTarget_shouldReturnHeat() {
8
9          TemperatureRegulator regulator = new TemperatureRegulator();
10         double current = 20.0;
11         double target = 22.0;
12
13
14         TemperatureRegulator.Action result = regulator.compute(current, target);
15
16
17         assertEquals(TemperatureRegulator.Action.HEAT, result);
18     }
19
20     @Test
```

Run TemperatureRegulatorTest x

✓ 5 tests passed 5 tests total, 60 ms

"C:\Program Files\Java\jdk-25\bin\java.exe" ...

Process finished with exit code 0

- ✓ TemperatureRegulator 60 ms
 - ✓ compute_currentSli 48 ms
 - ✓ compute_currentMl 10 ms
 - ✓ compute_currentWith 1 ms
 - ✓ compute_currentMuc 1 ms
 - ✓ compute_currentSlightlyLower

4 Conclusion

The session demonstrated the basics of unit testing in Java using JUnit.