

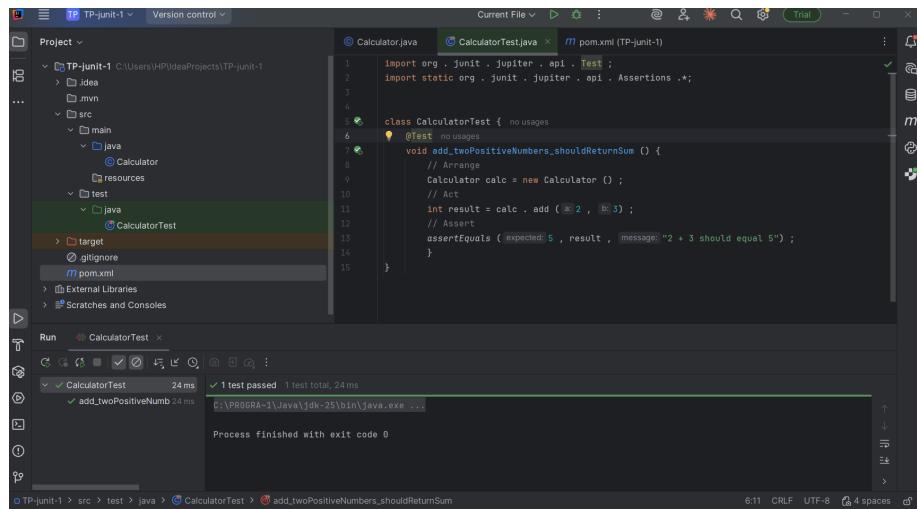
# Testing

El Gourji Nasreddine

November 2025

## 1 EX1

### 1.1 Success Test :



The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure with modules like TP-junit-1, Calculator, and CalculatorTest.
- Code Editor:** Displays the `CalculatorTest.java` file containing a single test method: `void add_twoPositiveNumbers_shouldReturnSum()`.
- Run Tab:** Shows the test results:
  - 1 test passed
  - 1 test total, 24 ms
  - Process finished with exit code 0
- Console Tab:** Shows the command line output of the test execution.

## 1.2 Failed Test after :

The screenshot shows the IntelliJ IDEA interface with the project 'TP-junit-1' open. The 'CalculatorTest.java' file is selected in the editor. The code contains a single test method:

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

class CalculatorTest {
    @Test
    void add_twoPositiveNumbers_shouldReturnSum() {
        // Arrange
        Calculator calc = new Calculator();
        // Act
        int result = calc.add(2, 3);
        // Assert
        assertEquals(5, result, "2 + 3 should equal 5");
    }
}
```

In the 'Run' tool window, a test run for 'CalculatorTest' is shown with one test failing:

- CalculatorTest (32 ms)
- add\_twoPositiveNumbers (32 ms) - Failed

The failure message is:

```
org.opentest4j.AssertionFailedError: 2 + 3 should equal 5
Expected :5
Actual   :1
<Click to see difference>
```

## 1.3 Success Test after adding the sub and mul features :

The screenshot shows the IntelliJ IDEA interface with the project 'TP-junit-1' open. The 'CalculatorTest.java' file is selected in the editor. The code now includes additional test methods for multiplication and division:

```
class CalculatorTest {
    ...
    @Test
    void mul_twoPositiveNumbers_shouldReturnSum() {
        // Arrange
        Calculator calc = new Calculator();
        // Act
        int result = calc.mul(2, 3);
        // Assert
        assertEquals(6, result, "2 * 3 should equal 6");
    }

    @Test
    void div_twoPositiveNumbers_shouldReturnSum() {
        // Arrange
        Calculator calc = new Calculator();
        // Act
        int result = calc.div(2, 3);
        // Assert
    }
}
```

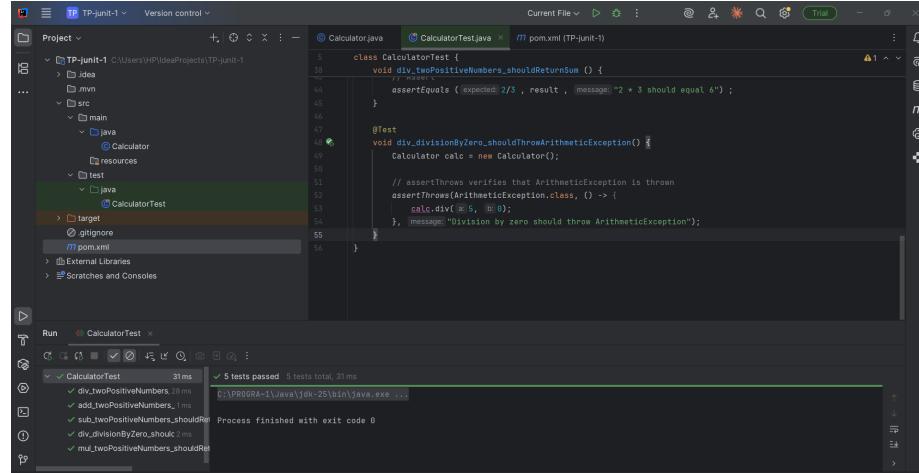
In the 'Run' tool window, a test run for 'CalculatorTest' is shown with four tests passed:

- CalculatorTest (26 ms)
- mul\_twoPositiveNumbers (24 ms) - Passed
- add\_twoPositiveNumbers (31 ms) - Passed
- sub\_twoPositiveNumbers (shouldFail) - Passed
- mul\_twoPositiveNumbers (shouldFail) - Passed

The output shows:

```
4 tests passed 4 tests total, 26 ms
C:\PR06RA-1\Java\jdk-25\bin\java.exe ...
```

## 1.4 Success Test for handling the zero exception:



```
class CalculatorTest {
    void div_twoPositiveNumbers_shouldReturnSum () {
        assertEquals( expected: 2/3 , result , message:"2 * 3 should equal 6");
    }

    @Test
    void div_divisionByZero_shouldThrowArithmeticException() {
        Calculator calc = new Calculator();
        // assertThrows verifies that ArithmeticException is thrown
        assertThrows(ArithmeticException.class, () -> {
            calc.div(0, 0);
        }, message:"Division by zero should throw ArithmeticException");
    }
}
```

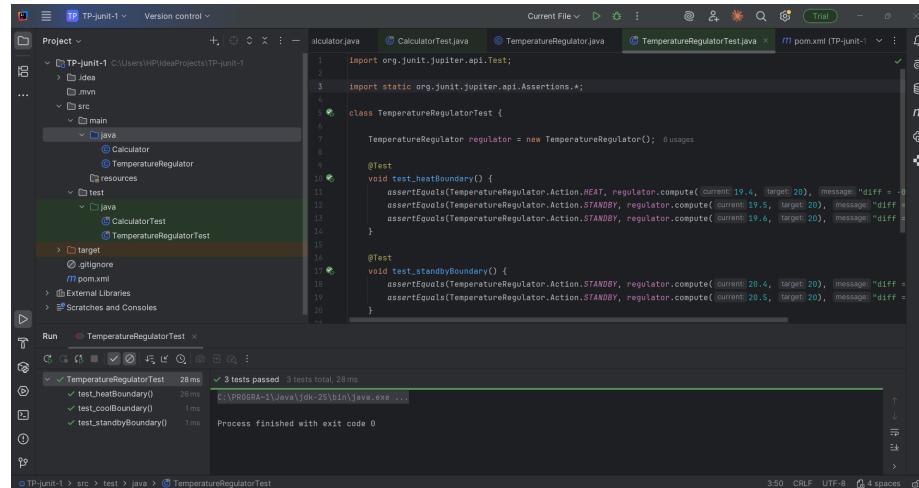
Run

- CalculatorTest 31ms
  - ✓ div\_twoPositiveNumbers 28ms
  - ✓ add\_twoPositiveNumbers 1ms
  - ✓ sub\_twoPositiveNumbers\_shouldReturnSum 1ms
  - ✓ div\_divisionByZero\_shouldThrowArithmeticException 0ms
  - ✓ mul\_twoPositiveNumbers\_shouldReturnProduct 1ms

5 tests passed 5 tests total, 31ms

## 2 EX1

### 2.1 Success Test :



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

class TemperatureRegulatorTest {
    TemperatureRegulator regulator = new TemperatureRegulator(); 6 usages

    @Test
    void test_heatBoundary() {
        assertEquals(TemperatureRegulator.Action.HEAT, regulator.compute( current: 19.4, target: 20 ), message: "diff = -0.6");
        assertEquals(TemperatureRegulator.Action.STANDBY, regulator.compute( current: 19.5, target: 20 ), message: "diff = -0.5");
        assertEquals(TemperatureRegulator.Action.STANDBY, regulator.compute( current: 19.6, target: 20 ), message: "diff = -0.4");
    }

    @Test
    void test_standbyBoundary() {
        assertEquals(TemperatureRegulator.Action.STANDBY, regulator.compute( current: 20.4, target: 20 ), message: "diff = 0.4");
        assertEquals(TemperatureRegulator.Action.STANDBY, regulator.compute( current: 20.5, target: 20 ), message: "diff = 0.5");
    }
}
```

Run

- TemperatureRegulatorTest 28ms
  - ✓ test\_heatBoundary() 26ms
  - ✓ test\_coolBoundary() 1ms
  - ✓ test\_standbyBoundary() 1ms

3 tests passed 3 tests total, 28ms