

# SEPQM-Lab\_02

## Test case 01 – Addition and SubtractionS

```
1 package Calculation;
2
3 public class Calculation {
4
5     public static int addition(int a, int b) {
6         return a + b;
7     }
8
9     public static int subtraction(int x, int y) {
10        return x - y;
11    }
12 }
13
```

Calculation.java

```
1 package Calculation;
2
3 import static org.junit.Assert.*;
4
5 public class CalculationTest {
6
7     @Test
8     public void testAddition() {
9         int a = 15;
10        int b = 20;
11        int expectedResult = 35; // Corrected expected value
12        int result = Calculation.addition(a, b);
13        assertEquals(expectedResult, result);
14    }
15
16     @Test
17     public void testSubtraction() {
18        int x = 20;
19        int y = 10;
20        int expectedResult = 10;
21        int result = Calculation.subtraction(x, y);
22        assertEquals(expectedResult, result);
23    }
24 }
25
26
27
28
29
30
31
```

CalculationTest.java

- This test case verifies both the addition and subtraction methods. It ensures that the addition method correctly returns the sum of two numbers, while the subtraction method accurately computes the difference between two integers. The test checks whether the actual results match the expected values.

## Test case 02 – Division

```
1 package Calculation;
2
3 public class Devision {
4
5     public static int devision (int a, int b) {
6         return a/b;
7     }
8
9
10
11
12 }
13
```

Devision.java

```
1 package Calculation;
2
3 import static org.junit.Assert.*;
4
5
6
7 public class DevisionTest {
8
9     @Test
10     public void testDevision1() {
11         int a = 100;
12         int b = 20;
13         int expectedResult = 5; // Corrected expected value
14         int result = Devision.devision(a, b);
15         assertEquals(expectedResult, result); }
16
17
18 |
19
20
21 }
22
```

DevisionTest.java

- This test case checks whether the `devision` method correctly calculates the quotient when dividing two numbers. It also ensures that edge cases, such as division by zero, are handled appropriately.

## Test case 03 – Max

```
1 package Calculation;
2
3 public class Max {
4
5     public static int max(int a, int b) {
6         return Math.max(a, b);
7     }
8
9
10
11 }
12
```

Max.java

```
1 package Calculation;
2
30 import static org.junit.Assert.*;
6
7 public class MaxTest {
8
9     @Test
10     public void testMax() {
11
12         int a = 5;
13         int b = 2;
14         int expectedResult = 5; // Corrected expected value
15         int result = Max.max(a, b);
16         assertEquals(expectedResult, result);
17
18     }
19
20 }
21 }
22
```

MaxTest.java

- This test case verifies the `max` method, which returns the larger of two numbers. It ensures that the correct maximum value is determined and returned.

## Test case 04 – Modulus

```
1 package Calculation;
2
3 public class Modulus {
4
5     public static int modulus(int a, int b) {
6
7         return a % b;
8     }
9
10
11
12
13
14 }
15
```

Modulus.java

```
1 package Calculation;
2
3 import static org.junit.Assert.*;
4
5
6
7 public class ModulusTest {
8
9     @Test
10     public void modulusTest() {
11         int a = 15;
12         int b = 7 ;
13         int expectedResult = 1; // Corrected expected value
14         int result = Modulus.modulus(a, b);
15         assertEquals(expectedResult, result); }
16
17 }
18
```

ModulusTest.java

- This test case ensures that the `modulus` method correctly calculates the remainder when one number is divided by another. It verifies that the computed remainder is as expected.

## Test case 05 – Power

```
1 package Calculation;
2
3 public class Power {
4
5     public static int power(int a, int b) {
6         return (int) Math.pow(a, b);
7     }
8
9
10
11
12 }
13
```

Power.java

```
1 package Calculation;
2
3 import static org.junit.Assert.*;
4
5
6
7 public class PowerTest {
8
9
10     @Test
11     public void testPower() {
12         int a = 5;
13         int b = 2;
14         int expectedResult = 25; // Corrected expected value
15         int result = Power.power(a, b);
16         assertEquals(expectedResult, result);
17
18
19
20
21
22     }
23
24
25
```

PowerTest.java

- This test case verifies the correctness of the `power` method, which computes the exponentiation of a base number raised to a given exponent. It ensures the result matches the expected value.

# Test Suite

```
1 package Calculation;
2
3 import org.junit.Before;
4
5 @RunWith(Suite.class)
6
7 @SuiteClasses({ CalculationTest.class , DevisionTest.class , MaxTest.class , ModulasTest.class ,PowerTest.class})
8
9
10
11 public class AllTests {
12
13     @Before
14     public void beforeAnnotation() {
15         System.out.println("Test all class");
16     }
17 }
18
19
20
21
22
```

JUnit Console

Finished after 0.021 seconds

Runs: 6/6    Errors: 0    Failures: 0

Calculation.AllTests [Runner: JUnit 4] (0.002 s)

- > Calculation.CalculationTest (0.001 s)
- > Calculation.DevisionTest (0.000 s)
- > Calculation.MaxTest (0.000 s)
- > Calculation.ModulasTest (0.000 s)
- > Calculation.PowerTest (0.001 s)

Failure Trace

- **The Test Suite is designed to run all individual test cases (Addition & Subtraction, Power, Division, Modulus, and Max) together in a single execution.**