# 20.SquareandCubeCalculationProgram(White-BoxTesting)

Aim:

Todevelopaprogramthatcalculatesthesquareandcubeofagivendecimalnumberand verify the output using white-box testing with JUnit.

Algorithm:

1. **Step1:**Accept adecimal numberas input.
2. **Step2:**Calculate thesquareofthenumberbymultiplyingthe numberbyitself.
3. **Step3:**Calculatethecubeofthenumberbymultiplyingthenumberbyitselfthree times.
4. **Step4:**Return both thesquareandcubeof thegivennumber.
5. **Step5:**WriteJUnittestcasestovalidatethesquareandcubecalculationlogic, including edge cases such as 0, 1, and negative numbers.

**Code:**

**Step1:**

public class SquareAndCubeCalculator {

public double calculateSquare(double number) {

return number \* number;

}

public double calculateCube(double number) {

return number \* number \* number;

}

}

**Step2:**

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

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

public class SquareAndCubeCalculatorTest {

private SquareAndCubeCalculator calculator;

@BeforeEach

public void setUp() {

calculator = new SquareAndCubeCalculator();

}

@Test

public void testSquareAndCubeOf2\_5() {

assertEquals(6.25, calculator.calculateSquare(2.5), 0.0001);

assertEquals(15.625, calculator.calculateCube(2.5), 0.0001);

}

@Test

public void testSquareAndCubeOfNegative3\_5() {

assertEquals(12.25, calculator.calculateSquare(-3.5), 0.0001);

assertEquals(-42.875, calculator.calculateCube(-3.5), 0.0001);

}

@Test

public void testSquareAndCubeOf0() {

assertEquals(0, calculator.calculateSquare(0), 0.0001);

assertEquals(0, calculator.calculateCube(0), 0.0001);

}

@Test

public void testSquareAndCubeOf1() {

assertEquals(1, calculator.calculateSquare(1), 0.0001);

assertEquals(1, calculator.calculateCube(1), 0.0001);

}

@Test

public void testSquareAndCubeOfNegative1() {

assertEquals(1, calculator.calculateSquare(-1), 0.0001);

assertEquals(-1, calculator.calculateCube(-1), 0.0001);

}

}

Sample Input:

# TestCase1:

* + - Input: 2.5
    - ExpectedOutput:
      * Square:6.25
      * Cube:15.625

# TestCase2:

* + - Input:-3.5
    - ExpectedOutput:
      * Square:12.25
      * Cube:-42.875

# TestCase3:

* + - Input:0
    - ExpectedOutput:
      * Square:0
      * Cube:0

# TestCase4:

* + - Input:1
    - ExpectedOutput:
      * Square:1
      * Cube:1

# TestCase5:

* + - Input:-1
    - ExpectedOutput:
      * Square:1
      * Cube:-1

SampleOutput:

# TestCase1:

* + - Input: 2.5
    - Output:
      * Square:6.25
      * Cube:15.625

# TestCase2:

* + - Input:-3.5
    - Output:
      * Square:12.25
      * Cube:-42.875

# TestCase3:

* + - Input:0
    - Output:
      * Square:0
      * Cube:0

# TestCase4:

* + - Input:1
    - Output:
      * Square:1
      * Cube:1

# TestCase5:

* + - Input:-1
    - Output:
      * Square:1
      * Cube:-1

Results:

* + **TestCase1:**Theprogramcorrectlycalculatesthesquare(6.25)andcube(15.625)for 2.5.
  + **TestCase2:**Theprogramcorrectlycalculatesthesquare(12.25)andcube(-42.875) for -3.5.
  + **TestCase3:**Theprogramcorrectlycalculates thesquare(0)andcube(0) for0.
  + **TestCase4:**Theprogramcorrectlycalculates thesquare(1)andcube(1) for1.
  + **TestCase5:**Theprogramcorrectlycalculates thesquare(1)andcube(-1) for-1.

