**Exercise 1: Setting Up Junit**

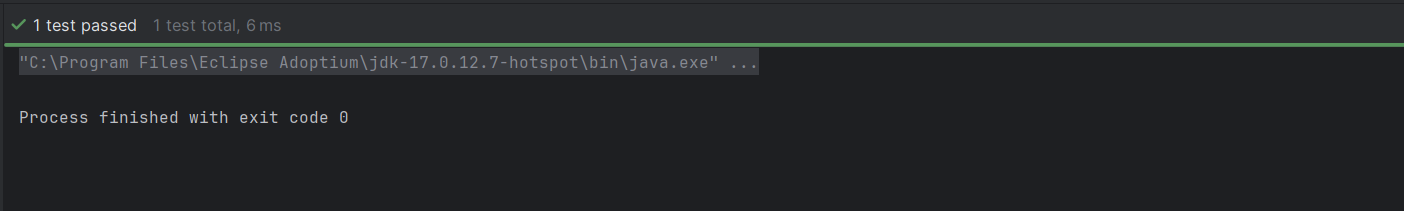
**Example.java**

public class Example {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

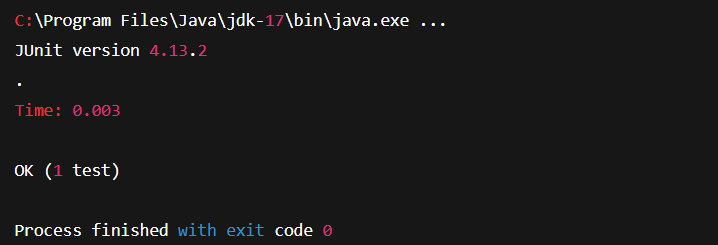
**ExampleTest.java**

import org.junit.Test;  
import static org.junit.Assert.\*;  
  
public class ExampleTest {  
  
 @Test  
 public void testAdd() {  
 Example example = new Example();  
 int result = example.add(3, 4);  
 *assertEquals*(7, result);  
 }  
}

**Output :**

****

**Exercise 3: Assertions in JUnit Scenario:**

****

**Exercise 4 :**

**MathUtils.java**

public class MathUtils {

public int square(int number) {

return number \* number;

}

public int modulus(int a, int b) {

if (b == 0) throw new ArithmeticException("Modulus by zero not allowed");

return a % b;

}

public boolean isEven(int number) {

return number % 2 == 0;

}

public boolean isPositive(int number) {

return number > 0;

}

}

**MathUtilsTest.java**

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsTest {

private MathUtils mathUtils;

@Before

public void setUp() {

mathUtils = new MathUtils();

System.out.println("MathUtils object created");

}

@After

public void tearDown() {

mathUtils = null;

System.out.println("MathUtils object cleared");

}

@Test

public void testSquare() {

int input = 7;

int result = mathUtils.square(input);

assertEquals(49, result);

}

@Test

public void testModulus() {

int result = mathUtils.modulus(20, 6);

assertEquals(2, result);

}

@Test(expected = ArithmeticException.class)

public void testModulusByZero() {

mathUtils.modulus(10, 0);

}

@Test

public void testIsEven() {

assertTrue(mathUtils.isEven(12));

assertFalse(mathUtils.isEven(13));

}

@Test

public void testIsPositive() {

assertTrue(mathUtils.isPositive(5));

assertFalse(mathUtils.isPositive(-3));

}

}

**Output :**

****