**simulating JUnit-style testing using basic Java code.**

**Exercise 1: Setting Up JUnit**

public class Main {

static class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

public int multiply(int a, int b) {

return a \* b;

}

public int divide(int a, int b) {

if (b == 0) {

throw new ArithmeticException("Division by zero");

}

return a / b;

}

}

public static void testAdd() {

Calculator calc = new Calculator();

int result = calc.add(3, 4);

if (result == 7) {

System.out.println("Test Add PASSED");

} else {

System.out.println("Test Add FAILED: expected 7 but got " + result);

}

}

public static void testDivide() {

Calculator calc = new Calculator();

try {

int result = calc.divide(10, 2);

if (result == 5) {

System.out.println("Test Divide PASSED");

} else {

System.out.println("Test Divide FAILED: expected 5 but got " + result);

}

} catch (ArithmeticException e) {

System.out.println("Test Divide FAILED with exception: " + e.getMessage());

}

}

public static void main(String[] args) {

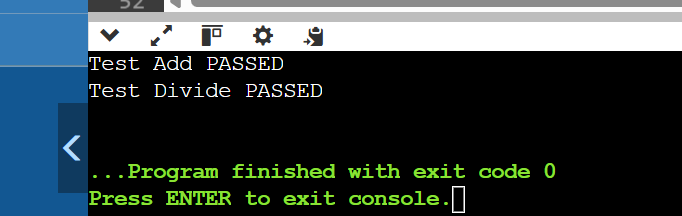
testAdd();

testDivide();

}

}

OUTPUT :



**Exercise 3: Assertions in Junit**

public class main {

public static void main(String[] args) {

testAssertions();

}

public static void testAssertions() {

if (2 + 3 == 5) {

System.out.println("assert Equals PASSED");

} else {

System.out.println("assert Equals FAILED");

}

if (5 > 3) {

System.out.println("assert True PASSED");

} else {

System.out.println("assert True FAILED");

}

if (!(5 < 3)) {

System.out.println("assert False PASSED");

} else {

System.out.println("assert False FAILED");

}

Object obj = null;

if (obj == null) {

System.out.println("assert Null PASSED");

} else {

System.out.println("assert Null FAILED");

}

Object obj2 = new Object();

if (obj2 != null) {

System.out.println("assert NotNull PASSED");

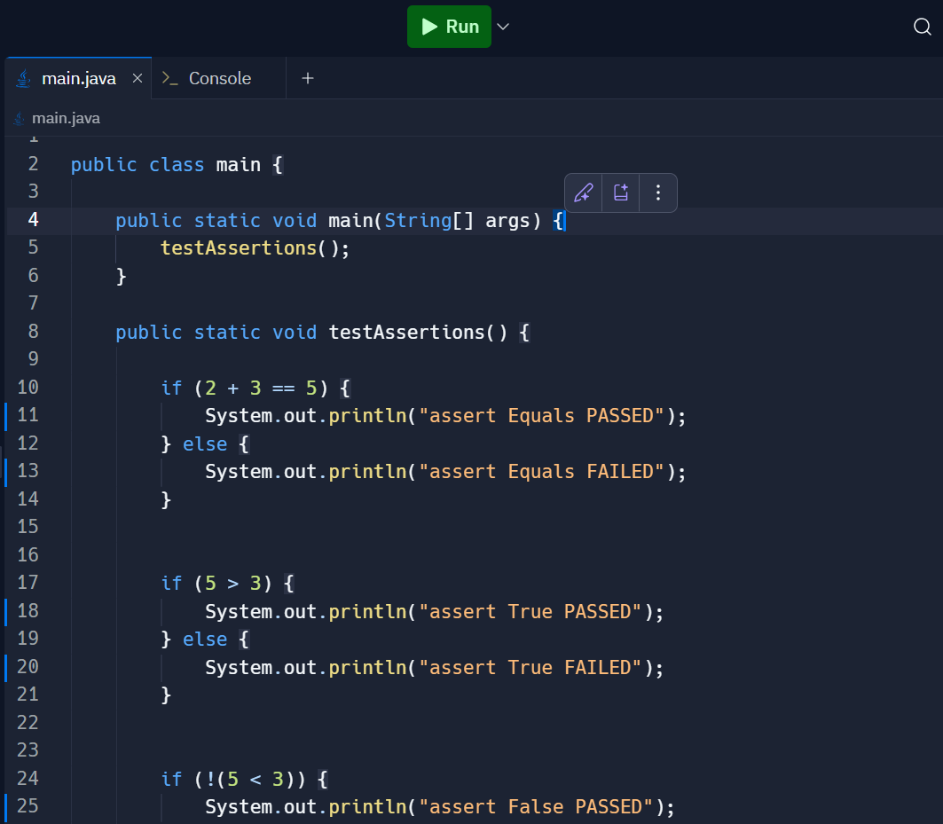
} else {

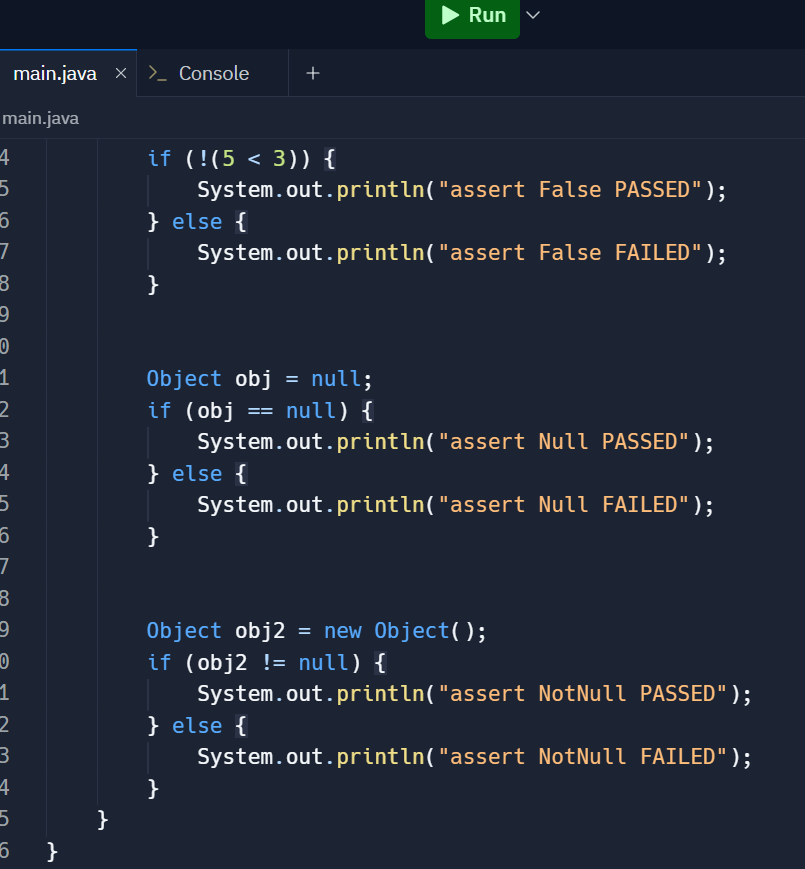
System.out.println("assert NotNull FAILED");

}

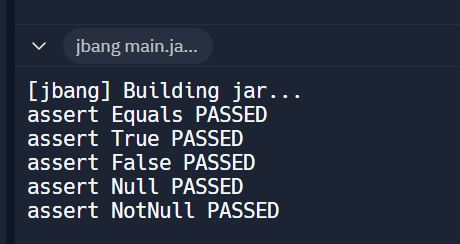
}

}





OUTPUT :



**Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit**

public class main {

public static void main(String[] args) {

CalculatorTest test = new CalculatorTest();

test.setUp();

test.testAdd();

test.tearDown();

test.setUp();

test.testSubtract();

test.tearDown();

}

}

class Calculator {

public int add(int a, int b) {

return a + b;

}

public int subtract(int a, int b) {

return a - b;

}

}

class CalculatorTest {

Calculator calculator;

public void setUp() {

calculator = new Calculator();

System.out.println("Setup: Calculator instance created.");

}

public void tearDown() {

calculator = null;

System.out.println("Teardown: Calculator instance destroyed.\n");

}

public void testAdd() {

int a = 5;

int b = 3;

int result = calculator.add(a, b);

if (result == 8) {

System.out.println("test Add PASSED");

} else {

System.out.println("test Add FAILED: expected 8 but got " + result);

}

}

public void testSubtract() {

int a = 10;

int b = 4;

int result = calculator.subtract(a, b);

if (result == 6) {

System.out.println("test Subtract PASSED");

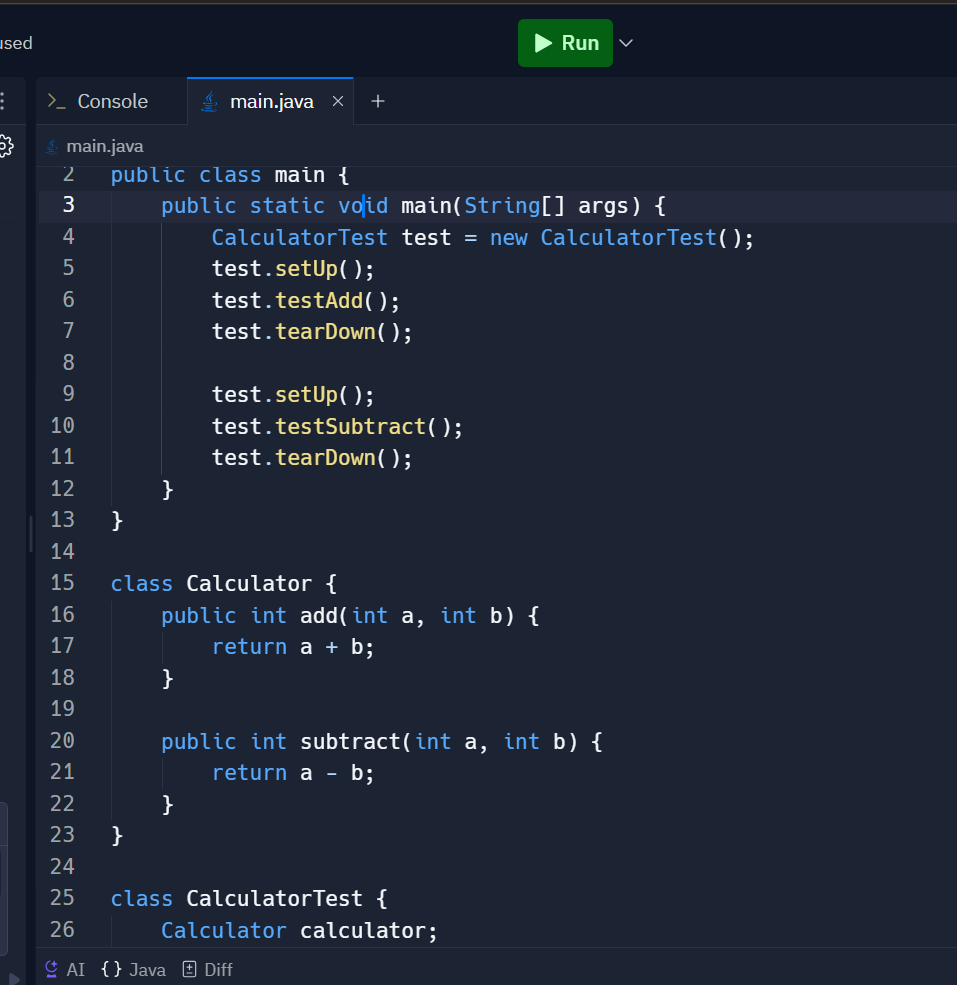
} else {

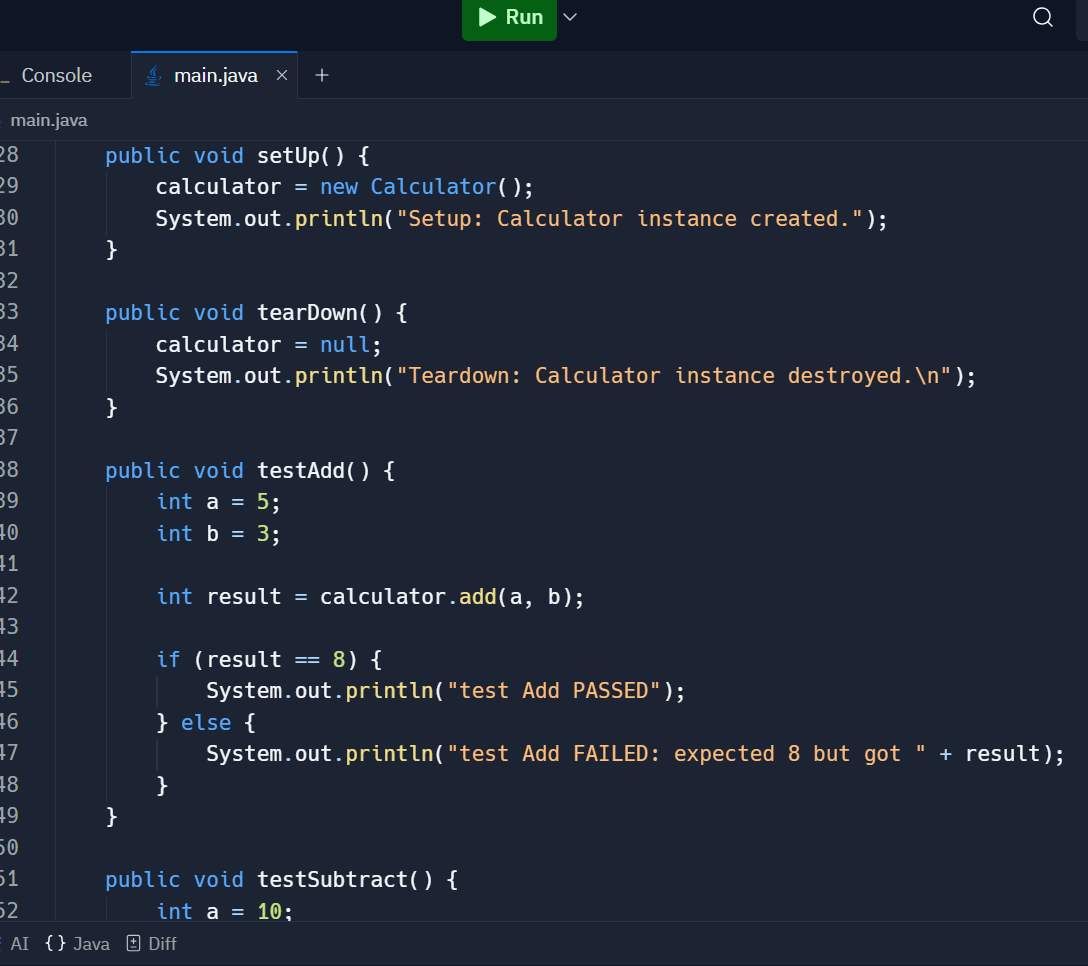
System.out.println("test Subtract FAILED: expected 6 but got " + result);

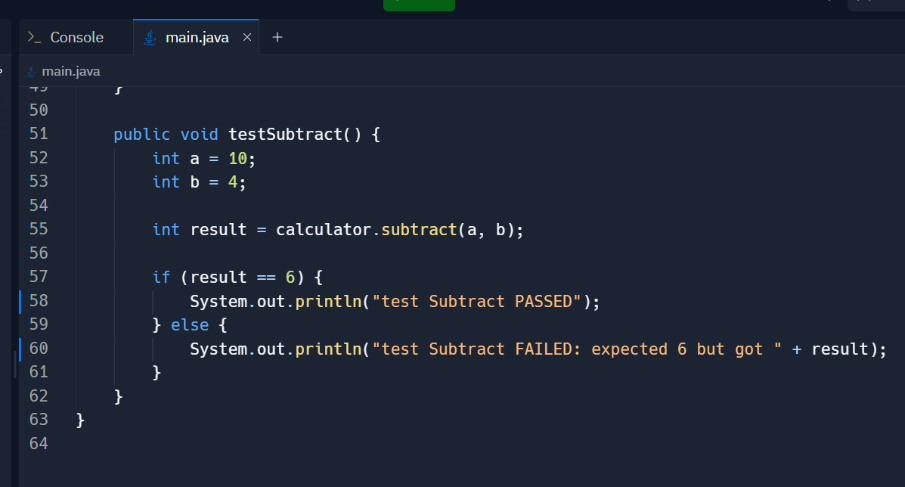
}

}

}







OUTPUT :

