* **JUnit Testing Exercises**

**Exercise 1: Setting Up JUnit**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>JUnitSetupExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>2.22.2</version>

</plugin>

</plugins>

</build>

</project>

**Calculator.java**

package com.example.test;

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

public boolean isPositive(int number) {

return number > 0;

}

public int divide(int a, int b) {

return a / b;

}

}

**CalculatorTest.java**

package com.example.test;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

int result = calc.add(10, 5);

assertEquals(15, result);

}

@Test

public void testMultiply() {

Calculator calc = new Calculator();

int result = calc.multiply(4, 6);

assertEquals(24, result);

}

@Test

public void testIsPositive() {

Calculator calc = new Calculator();

boolean result = calc.isPositive(7);

assertTrue(result);

}

@Test

public void testDivide() {

Calculator calc = new Calculator();

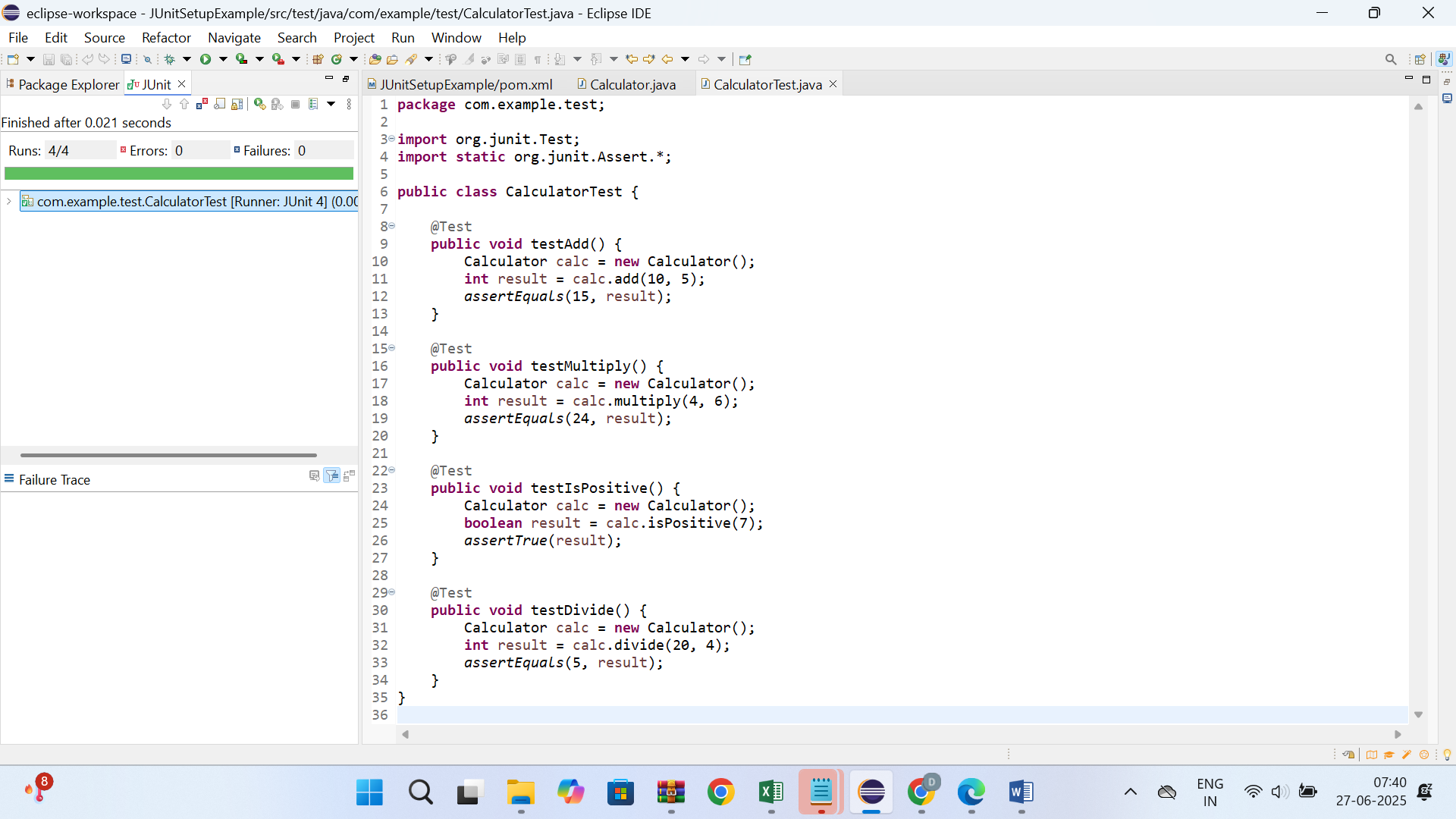
int result = calc.divide(20, 4);

assertEquals(5, result);

}

}

**OUTPUT**



* **Exercise 2: Writing Basic JUnit Tests**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>BasicJUnitTestExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**MathUtils.java**

package com.example.test;

public class MathUtils {

public int add(int a, int b) {

return a + b;

}

public int subtract(int x, int y) {

return x - y;

}

public int square(int n) {

return n \* n;

}

public boolean isEven(int n) {

return n % 2 == 0;

}

}

**MathUtilsTest.java**

package com.example.test;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MathUtilsTest {

@Test

public void testAdd() {

MathUtils utils = new MathUtils();

int result = utils.add(5, 3);

assertEquals(8, result);

}

@Test

public void testSubtract() {

MathUtils utils = new MathUtils();

int result = utils.subtract(10, 4);

assertEquals(6, result);

}

@Test

public void testSquare() {

MathUtils utils = new MathUtils();

int result = utils.square(7);

assertEquals(49, result);

}

@Test

public void testIsEvenTrue() {

MathUtils utils = new MathUtils();

boolean result = utils.isEven(10);

assertTrue(result);

}

@Test

public void testIsEvenFalse() {

MathUtils utils = new MathUtils();

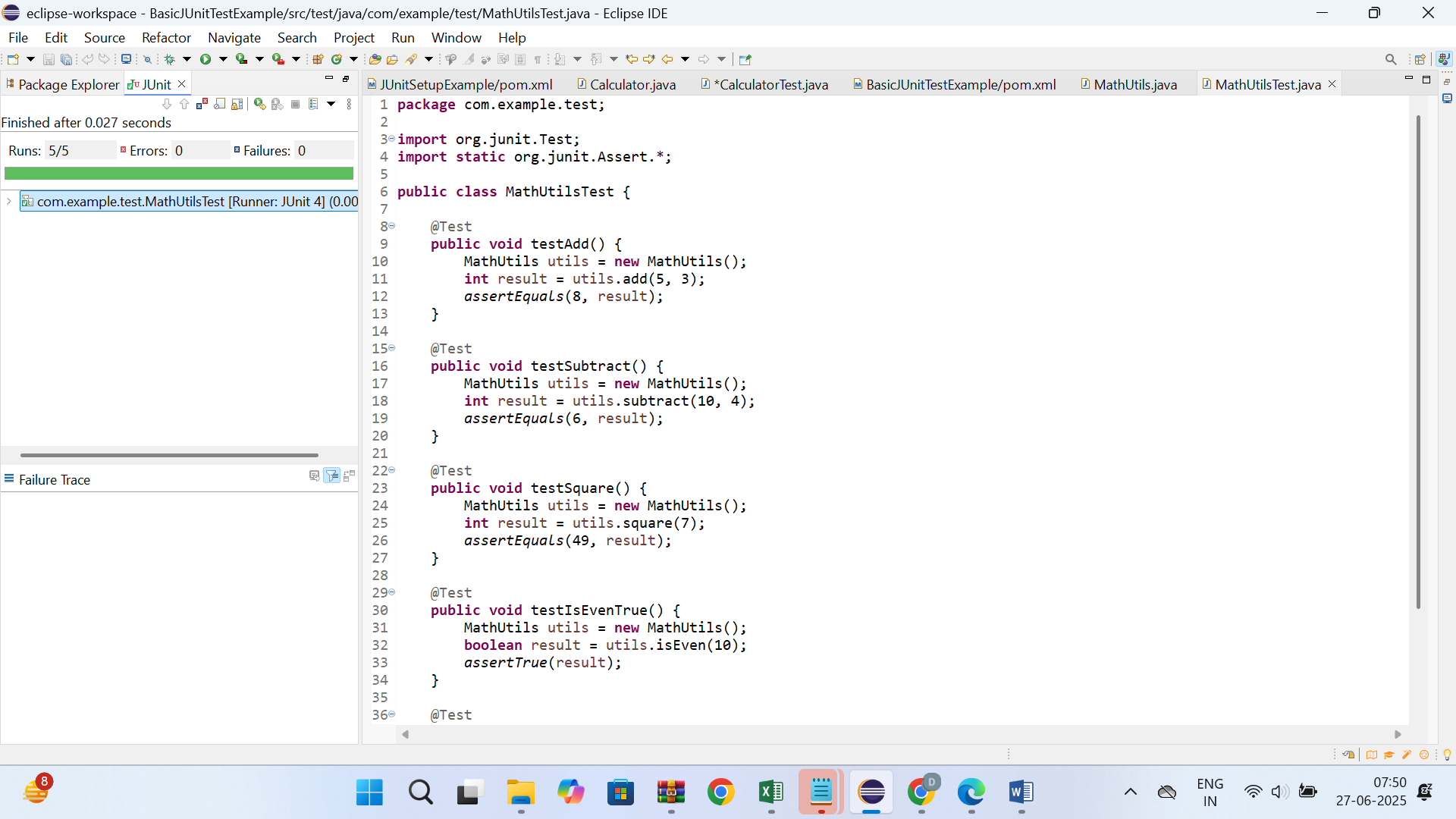
boolean result = utils.isEven(9);

assertFalse(result);

}

}

**OUTPUT**



* **Exercise 3: Assertions in JUnit**

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>AssertionsExample</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**AssertionsTest.java**

package com.example.test;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

@Test

public void testAssertions() {

assertEquals(5, 2 + 3);

assertTrue(5 > 3);

assertFalse(5 < 3);

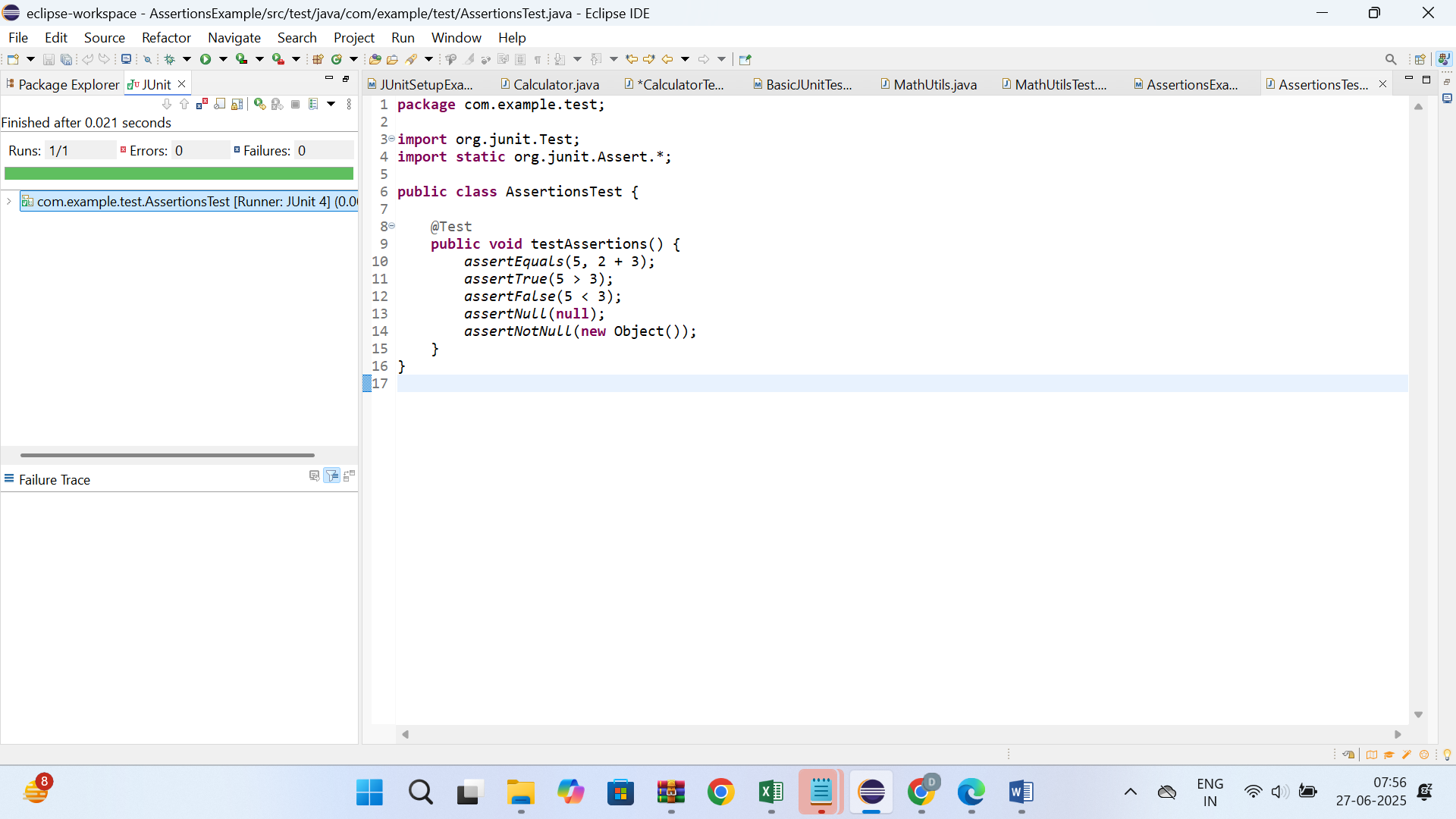
assertNull(null);

assertNotNull(new Object());

}

}

**OUTPUT**



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

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>AAAPatternTest</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**BankAccount.java**

package com.example.test;

public class BankAccount {

private String owner;

private int balance;

public BankAccount(String owner, int initialBalance) {

this.owner = owner;

this.balance = initialBalance;

}

public String getOwner() {

return owner;

}

public int getBalance() {

return balance;

}

public void deposit(int amount) {

balance += amount;

}

public void withdraw(int amount) {

if (amount <= balance) {

balance -= amount;

}

}

public void reset() {

balance = 0;

}

}

**BankAccountTest.java**

package com.example.test;

import org.junit.Before;

import org.junit.After;

import org.junit.Test;

import static org.junit.Assert.\*;

public class BankAccountTest {

private BankAccount account;

@Before

public void setUp() {

account = new BankAccount("John", 1000);

}

@After

public void tearDown() {

account = null;

}

@Test

public void testDeposit() {

int initial = account.getBalance();

account.deposit(500);

int finalBalance = account.getBalance();

assertEquals(initial + 500, finalBalance);

}

@Test

public void testWithdraw() {

int initial = account.getBalance();

account.withdraw(300);

int finalBalance = account.getBalance();

assertEquals(initial - 300, finalBalance);

}

@Test

public void testWithdrawOverdraft() {

account.withdraw(1500);

int finalBalance = account.getBalance();

assertEquals(1000, finalBalance);

}

@Test

public void testReset() {

account.reset();

int finalBalance = account.getBalance();

assertEquals(0, finalBalance);

}

}

**OUTPUT**

