JUnit Testing Exercises

**Exercise 1**: Setting Up JUnit

Scenario:

You need to set up JUnit in your Java project to start writing unit tests.

Steps:

1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).

2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml: junit junit 4.13.2 test

3. Create a new test class in your project.

**CODE**

**//**Calculator.java

package com.example;

public class Calculator {

// TODO Auto-generated method stub

public int add(int a, int b) {

return a + b;

}

}

**//** CalculatorTest.java

package com.example;

import org.junit.Test;

import static org.junit.Assert.*assertEquals*;

public class CalculatorTest {

*@Test*

// TODO Auto-generated method stub

public void testAdd() {

Calculator calc = new Calculator();

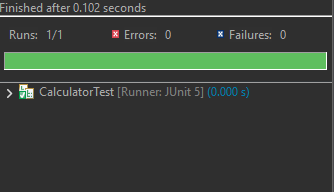
int result = calc.add(5, 3);

*assertEquals*(8, result);

}

}

**OUTPUT**

****

**Exercise 3:** Assertions in JUnit

Scenario: You need to use different assertions in JUnit to validate your test results.

Steps: 1. Write tests using various JUnit assertions.

**CODE**

**//** AssertionsTest.java

package com.example;

import org.junit.Test;

import static org.junit.Assert.\*;

public class AssertionsTest {

*@Test*

public void testAssertions() {

// Assert equals

*assertEquals*(5, 2 + 3);

// Assert true

*assertTrue*(5 > 3);

// Assert false

*assertFalse*(5 < 3);

// Assert null

*assertNull*(null);

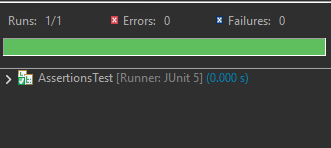
// Assert not null

*assertNotNull*(new Object());

}

}

**OUTPUT**

****

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

Scenario: You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

**CODE:**

// BankAccount.java

package com.example;

public class BankAccount {

private double balance;

public BankAccount() {

this.balance = 0.0;

}

public void deposit(double amount) {

if (amount > 0) {

balance += amount;

}

}

public boolean withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

return true;

}

return false;

}

public double getBalance() {

return balance;

}

}

//BankAccountTest.java

package com.example;

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();

System.*out*.println("Setup: New BankAccount created");

}

*@After*

public void tearDown() {

account = null;

System.*out*.println("Teardown: BankAccount cleared");

}

*@Test*

public void testDeposit() {

double depositAmount = 100.0;

account.deposit(depositAmount);

*assertEquals*(100.0, account.getBalance(), 0.001);

}

*@Test*

public void testWithdrawSuccess() {

account.deposit(200.0);

boolean result = account.withdraw(150.0);

*assertTrue*(result);

*assertEquals*(50.0, account.getBalance(), 0.001);

}

*@Test*

public void testWithdrawFail\_InsufficientFunds() {

// Arrange

account.deposit(50.0);

boolean result = account.withdraw(100.0);

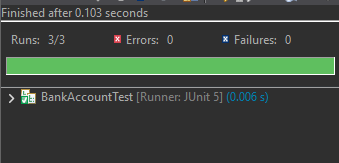
*assertFalse*(result);

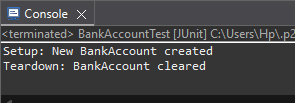
*assertEquals*(50.0, account.getBalance(), 0.001);

}

}

**OUTPUT:**

****

****