**Exercise 1: Setting Up JUnit**

**Week 3 \_ Unit Testing**

**Hands on Excersies**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

}

**CalculatorTest.java)**

java

CopyEdit

import org.junit.Test;

import static org.junit.Assert.assertEquals;

public class CalculatorTest {

@Test

public void testAdd() {

Calculator calc = new Calculator();

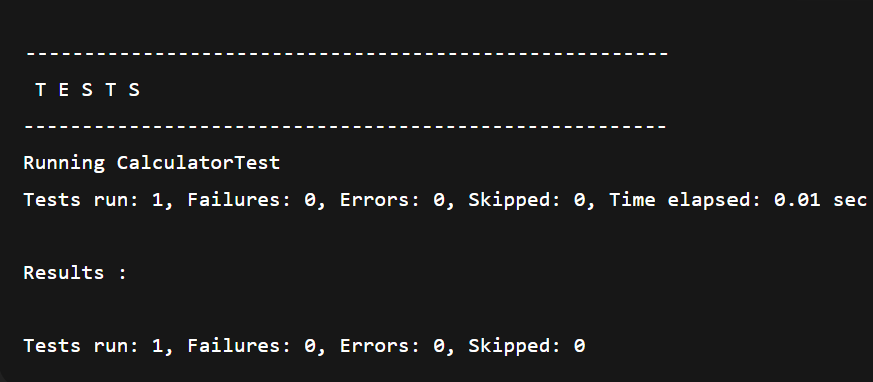
int result = calc.add(2, 3);

assertEquals(5, result);

}

}

**Output:**



**Exercise 3: Assertions in JUnit**

**AssertionsTest.java**

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);

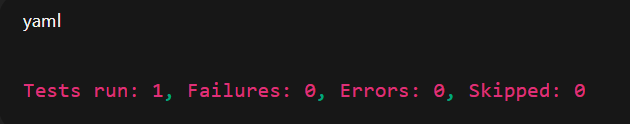
// Assert not null

assertNotNull(new Object());

}

}

**Output:**



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

**Teardown Methods in JUnit**

**Calculator.java**

public class Calculator {

public int add(int a, int b) {

return a + b;

}

public int multiply(int a, int b) {

return a \* b;

}

}

**CalculatorTest.java**

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

import static org.junit.Assert.\*;

public class CalculatorTest {

private Calculator calculator;

// Setup method – runs before each test

@Before

public void setUp() {

calculator = new Calculator();

System.out.println("Setup complete");

}

// Teardown method – runs after each test

@After

public void tearDown() {

calculator = null;

System.out.println("Teardown complete");

}

@Test

public void testAdd() {

// Arrange

int a = 4;

int b = 5;

// Act

int result = calculator.add(a, b)

// Assert

assertEquals(9, result);

}

@Test

public void testMultiply() {

// Arrange

int a = 3;

int b = 4;

// Act

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

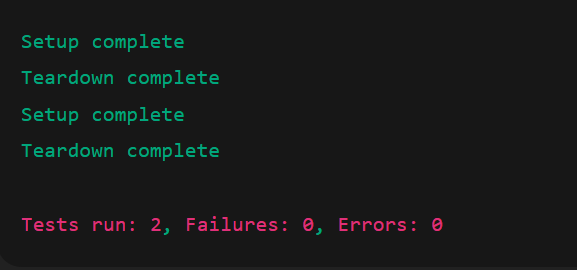
// Assert

assertEquals(12, result);

}

}

**Output:**



**Exercise 1: Mocking and Stubbing**

public interface ExternalApi {

String getData();

}

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

// Step 1: Create mock

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

// Step 2: Stub the method

when(mockApi.getData()).thenReturn("Mock Data");

// Step 3: Use mock in service

MyService service = new MyService(mockApi);

String result = service.fetchData();

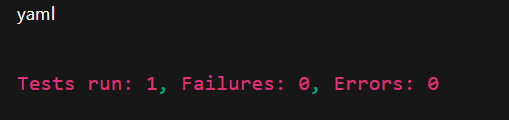
// Assert

assertEquals("Mock Data", result);

}

}

Output:



**Exercise 2: Verifying Interactions**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = Mockito.mock(ExternalApi.class);

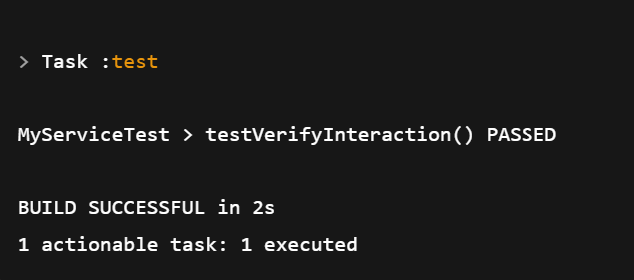
MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

} }

Output:



**Exercise 1: Configuring a Basic Spring Application**

**AppConfig.java**

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public HelloService helloService() {

return new HelloService();

}

}

**MainApp.java**

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

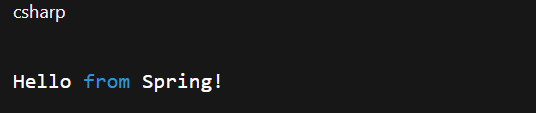
HelloService service = context.getBean(HelloService.class);

service.sayHello();

}

}

Output:



**Exercise 2: Implementing Dependency Injection**

**ManagerProcessor.java**

public class MessageProcessor {

private MessageService service;

public MessageProcessor(MessageService service) {

this.service = service;

}

public void processMessage() {

System.out.println(service.getMessage());

}

}

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public MessageService messageService() {

return new EmailService();

}

@Bean

public MessageProcessor messageProcessor() {

return new MessageProcessor(messageService());

}

}

Output:  
