MOCKITO EXERCISE

Exercise 1: Mocking and Stubbing

**Class: MyServiceTest**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

public class MyServiceTest {

@Test

public void testExternalApi() {

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

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

MyService service = new MyService(mockApi);

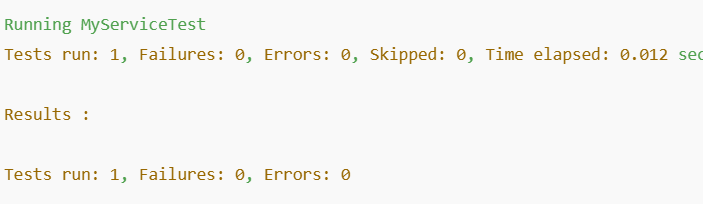
String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**OUTPUT:**



**Exercise 2: Verifying Interactions**

**Class: MyServiceTestimport 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);

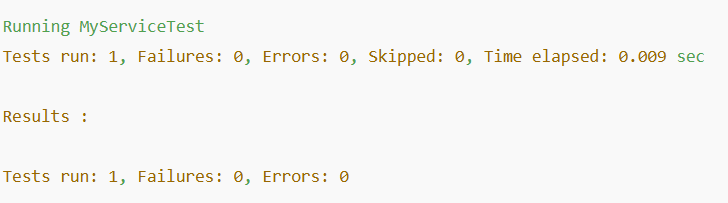
service.fetchData();

verify(mockApi).getData();

}

}

**OUTPUT:**

**Exercise 3: Argument Matching**

**Class: MyServiceTest**

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.anyString;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testArgumentMatching() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

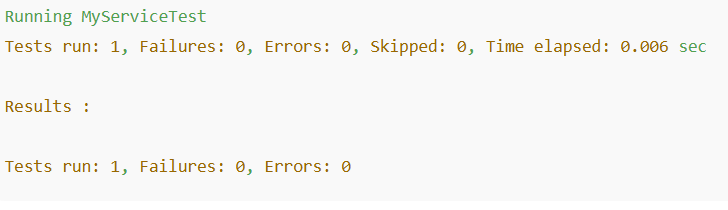
service.sendData("Hello");

verify(mockApi).send(anyString());

}

}

**OUTPUT:**

****

**Exercise 4: Handling Void Methods**

**Class: MyServiceTest**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testVoidMethod() {

ExternalApi mockApi = mock(ExternalApi.class);

doNothing().when(mockApi).logActivity();

MyService service = new MyService(mockApi);

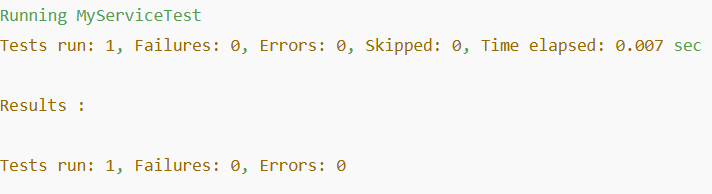
service.logUserActivity();

verify(mockApi).logActivity();

}

}

**OUTPUT:**

**Exercise 5: Mocking and Stubbing with Multiple Returns**

**Class: MyServiceTest**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testMultipleReturns() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData())

.thenReturn("First Call")

.thenReturn("Second Call");

MyService service = new MyService(mockApi);

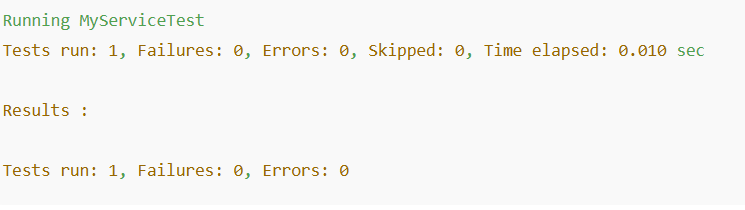
assertEquals("First Call", service.fetchData());

assertEquals("Second Call", service.fetchData());

}

}

**OUTPUT:**

****

**Exercise 6: Verifying Interaction Order**

**Class: MyServiceTest**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.InOrder;

public class MyServiceTest {

@Test

public void testInteractionOrder() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.process();

InOrder inOrder = inOrder(mockApi);

inOrder.verify(mockApi).start();

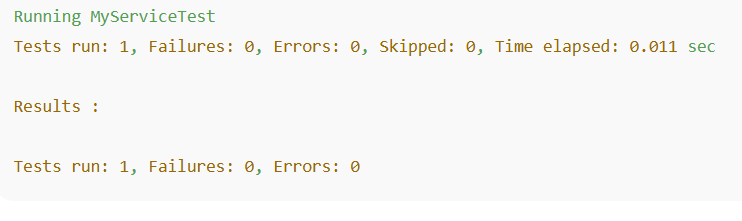
inOrder.verify(mockApi).execute();

inOrder.verify(mockApi).finish();

}

}

**OUTPUT:**

****

**Exercise 7: Handling Void Methods with Exceptions**

**Class: MyServiceTest**

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testVoidException() {

ExternalApi mockApi = mock(ExternalApi.class);

doThrow(new RuntimeException("Error")).when(mockApi).logActivity();

MyService service = new MyService(mockApi);

assertThrows(RuntimeException.class, () -> service.logUserActivity());

}

}

**OUTPUT:**

