Exercise 7: Handling Void Methods with Exceptions (Mockito)

**Externalapi.java:**

package com.example.mockito;

public interface Externalapi {

String getData();

String getDetails(String userId);

void logAccess(String userId);

void startSession();

void endSession();

}

**Myservice.java:**

package com.example.mockito;

public class Myservice {

private Externalapi api;

public Myservice(Externalapi api) {

this.api = api;

}

public String fetchData() {

return api.getData(); }

public String fetchUserDetails(String userId) {

return api.getDetails(userId); }

public void access(String userId) {

api.logAccess(userId);

}

public String fetchDataTwice() {

String first = api.getData();

String second = api.getData();

return first+" & "+second; }

public void performSession() {

api.startSession();

api.getData();

api.endSession();

} }

**Myservicetest.java:**

package com.example.mockito;

import org.junit.jupiter.api.Test;

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

import static org.mockito.Mockito.\*;

public class Myservicetest {

@Test

public void testVoidMethodThrowsException() {

Externalapi mockApi = *mock*(Externalapi.class);

*doThrow*(new RuntimeException("Logging failed"))

.when(mockApi).logAccess("U999");

Myservice service = new Myservice(mockApi);

RuntimeException thrown = *assertThrows*(RuntimeException.class,()->{

service.access("U999");

});

*assertEquals*("Logging failed", thrown.getMessage());

}

}