Advanced Mockito Exercises Solutions

Exercise 1: Mocking Databases and Repositories

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class ServiceTest {

@Test

public void testServiceWithMockRepository() {

Repository mockRepository = mock(Repository.class);

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

Service service = new Service(mockRepository);

String result = service.processData();

assertEquals("Processed Mock Data", result);

verify(mockRepository).getData();

}

}

Exercise 2: Mocking External Services (RESTful APIs)

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class ApiServiceTest {

@Test

public void testServiceWithMockRestClient() {

RestClient mockRestClient = mock(RestClient.class);

when(mockRestClient.getResponse()).thenReturn("Mock Response");

ApiService apiService = new ApiService(mockRestClient);

String result = apiService.fetchData();

assertEquals("Fetched Mock Response", result);

verify(mockRestClient).getResponse();

}

}

Exercise 3: Mocking File I/O

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class FileServiceTest {

@Test

public void testServiceWithMockFileIO() {

FileReader mockFileReader = mock(FileReader.class);

FileWriter mockFileWriter = mock(FileWriter.class);

when(mockFileReader.read()).thenReturn("Mock File Content");

FileService fileService = new FileService(mockFileReader, mockFileWriter);

String result = fileService.processFile();

assertEquals("Processed Mock File Content", result);

verify(mockFileReader).read();

verify(mockFileWriter, times(1)).write(anyString());

}

}

Exercise 4: Mocking Network Interactions

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class NetworkServiceTest {

@Test

public void testServiceWithMockNetworkClient() {

NetworkClient mockNetworkClient = mock(NetworkClient.class);

when(mockNetworkClient.connect()).thenReturn("Mock Connection");

NetworkService networkService = new NetworkService(mockNetworkClient);

String result = networkService.connectToServer();

assertEquals("Connected to Mock Connection", result);

verify(mockNetworkClient).connect();

}

}

Exercise 5: Mocking Multiple Return Values

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class MultiReturnServiceTest {

@Test

public void testServiceWithMultipleReturnValues() {

Repository mockRepository = mock(Repository.class);

when(mockRepository.getData())

.thenReturn("First Mock Data")

.thenReturn("Second Mock Data");

Service service = new Service(mockRepository);

String firstResult = service.processData();

String secondResult = service.processData();

assertEquals("Processed First Mock Data", firstResult);

assertEquals("Processed Second Mock Data", secondResult);

verify(mockRepository, times(2)).getData();

}

}

Additional Advanced Exercises

Exercise 6: Mocking with Argument Captors

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

import org.mockito.ArgumentCaptor;

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

public class ArgumentCaptorTest {

@Test

public void testArgumentCaptor() {

Repository mockRepository = mock(Repository.class);

Service service = new Service(mockRepository);

service.saveData("Test Data");

ArgumentCaptor<String> captor = ArgumentCaptor.forClass(String.class);

verify(mockRepository).save(captor.capture());

assertEquals("Test Data", captor.getValue());

}

}

Exercise 7: Mocking with Callbacks

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

public class CallbackTest {

@Test

public void testWithCallback() {

Repository mockRepository = mock(Repository.class);

when(mockRepository.processWithCallback(anyString()))

.thenAnswer(invocation -> {

String input = invocation.getArgument(0);

return "Processed " + input;

});

Service service = new Service(mockRepository);

String result = service.processWithCallback("Data");

assertEquals("Processed Data", result);

}

}