Name : Arshiya Tabassum A

Superset ID : 6424209

**WEEK 2**

**Module 4**

**Exercise 1: Mocking Databases and Repositories**

**CODE:**

Repository.java

public interface Repository {

String getData();

}

Service.java

public class Service {

private Repository repository;

public Service(Repository repository) {

this.repository = repository;

}

public String processData() {

return "Processed " + repository.getData();

}

}

ServiceTest.java

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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);

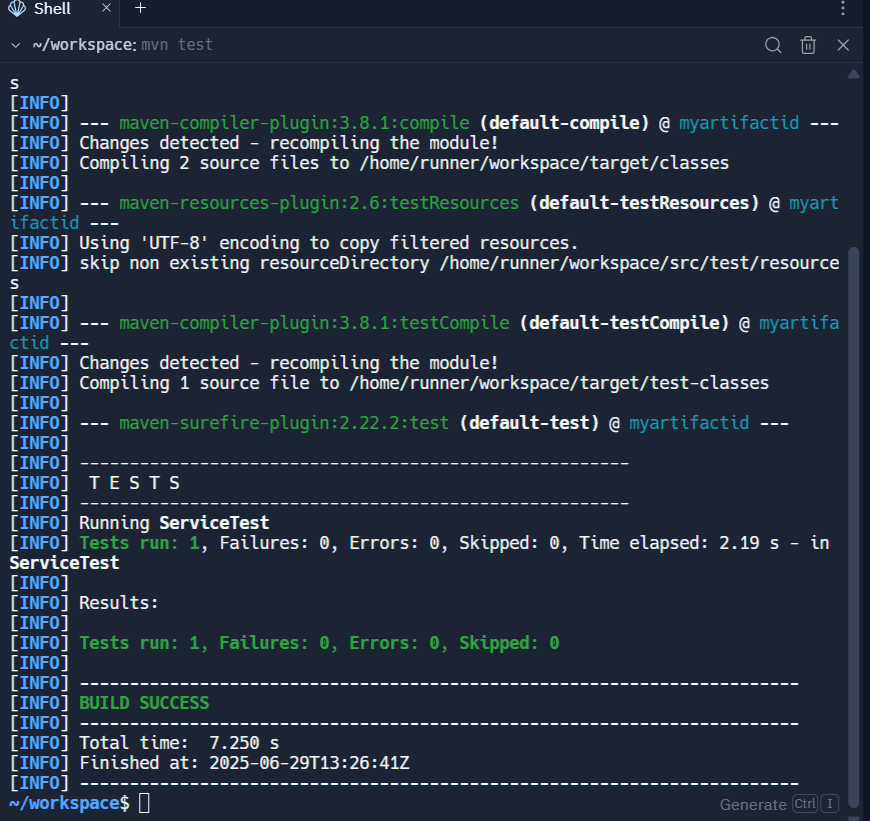
String result = service.processData();

assertEquals("Processed Mock Data", result);

}

}

**OUTPUT:**

****

**Exercise 2: Mocking External Services (RESTful APIs)**

**CODE:**

RestClient.java

public interface RestClient {

String getResponse();

}

ApiService.java

public class ApiService {

private RestClient restClient;

public ApiService(RestClient restClient) {

this.restClient = restClient;

}

public String fetchData() {

return "Fetched " + restClient.getResponse();

}

}

ApiServiceTest.java

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

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);

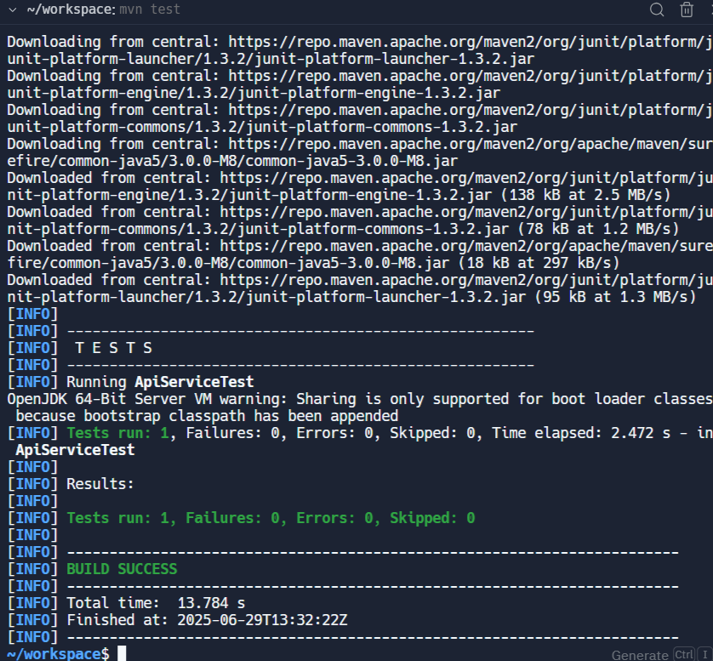
String result = apiService.fetchData();

assertEquals("Fetched Mock Response", result);

}

}

**OUTPUT:**

****

**Exercise 3: Mocking File I/O**

**CODE:**

FileReader.java

public interface FileReader {

String read();

}

FileWriter.java

public interface FileWriter {

void write(String data);

**}**

FileService.java

import io.FileReader;

import io.FileWriter;

public class FileService {

private FileReader reader;

private FileWriter writer;

public FileService(FileReader reader, FileWriter writer) {

this.reader = reader;

this.writer = writer;

}

public String processFile() {

String data = reader.read();

writer.write(data);

return "Processed " + data;

}

}

FileServiceTest.java

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

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

import io.FileReader;

import io.FileWriter;

import service.FileService;

public class FileServiceTest {

@Test

public void testServiceWithMockFileIO() {

FileReader mockReader = mock(FileReader.class);

FileWriter mockWriter = mock(FileWriter.class);

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

FileService fileService = new FileService(mockReader, mockWriter);

String result = fileService.processFile();

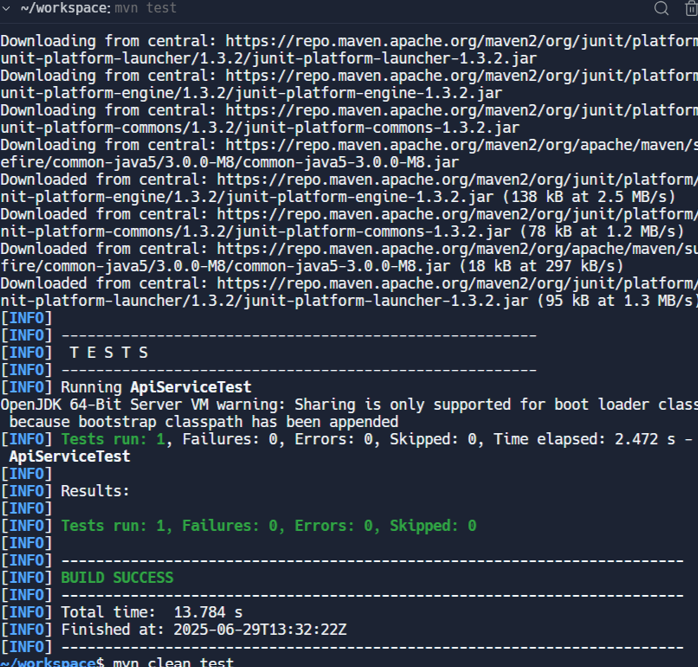
assertEquals("Processed Mock File Content", result);

verify(mockWriter).write("Mock File Content");

}

}

**OUTPUT:**



**Exercise 4: Mocking Network Interactions.**

**CODE:**

NetworkClient.java

package service;

public interface NetworkClient {

String connect();

}

NetworkService.java

package service;

public class NetworkService {

private NetworkClient client;

public NetworkService(NetworkClient client) {

this.client = client;

}

public String connectToServer() {

return "Connected to " + client.connect();

}

}

**NetworkServiceTest.java:**

package service;

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);

}

}

**OUTPUT:**

