**WEEK-2**

**Advanced Mockito Hands-On Exercises**

**Superset ID: 6419740**

**Exercise 1: Mocking Databases and Repositories**

**Code:**

**Repository.java:**

public class Service {

private Repository repository;

public Service(Repository repository) {

this.repository = repository;

}

public String processData() {

String data = repository.getData();

return "Processed " + data;

}

}

**Service.java:**

public class Service {

private Repository repository;

public Service(Repository repository) {

this.repository = repository;

}

public String processData() {

String data = repository.getData();

return "Processed " + data;

}

}

**ServiceTest.java:**

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);

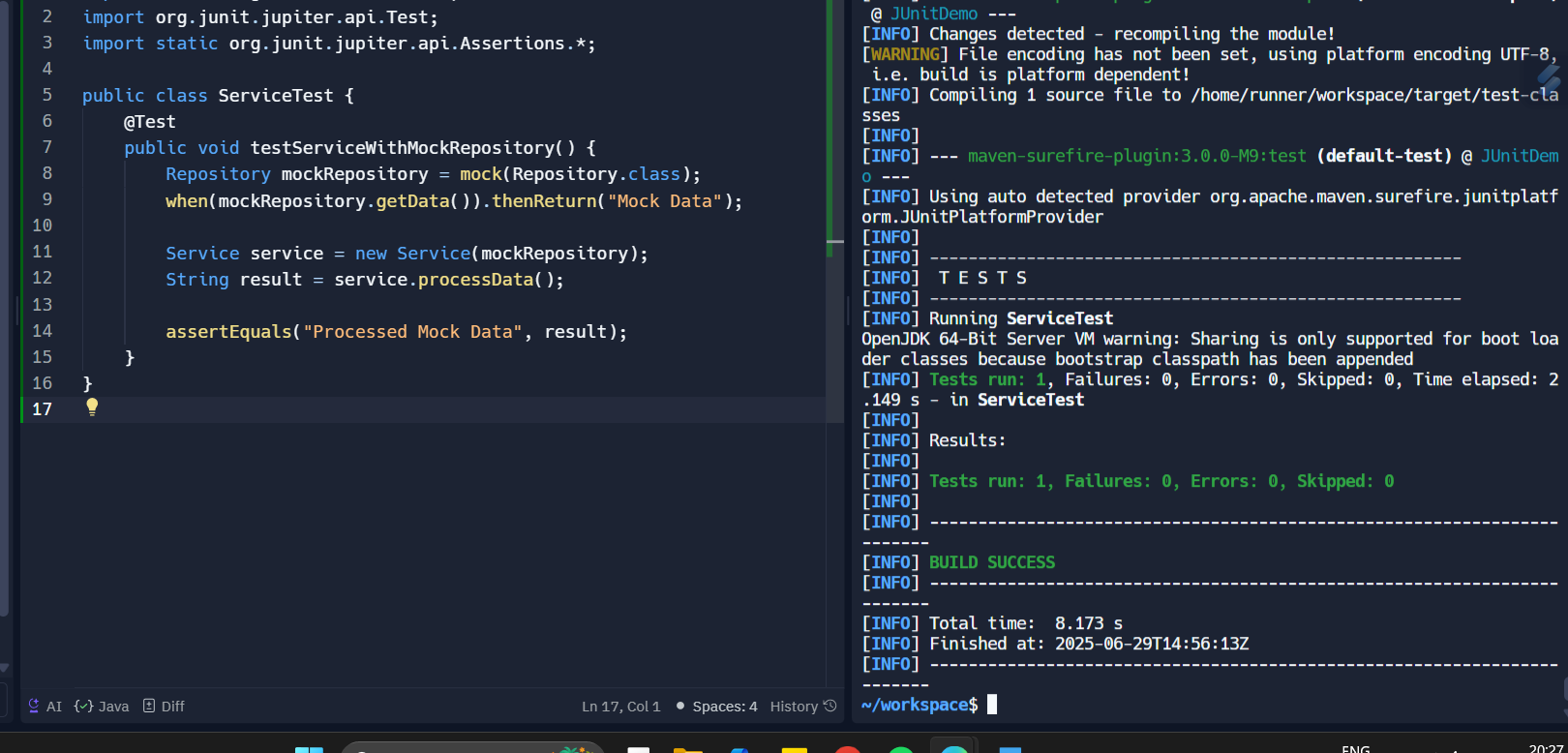
String result = service.processData();

assertEquals("Processed Mock Data", result);

}

}

**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 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);

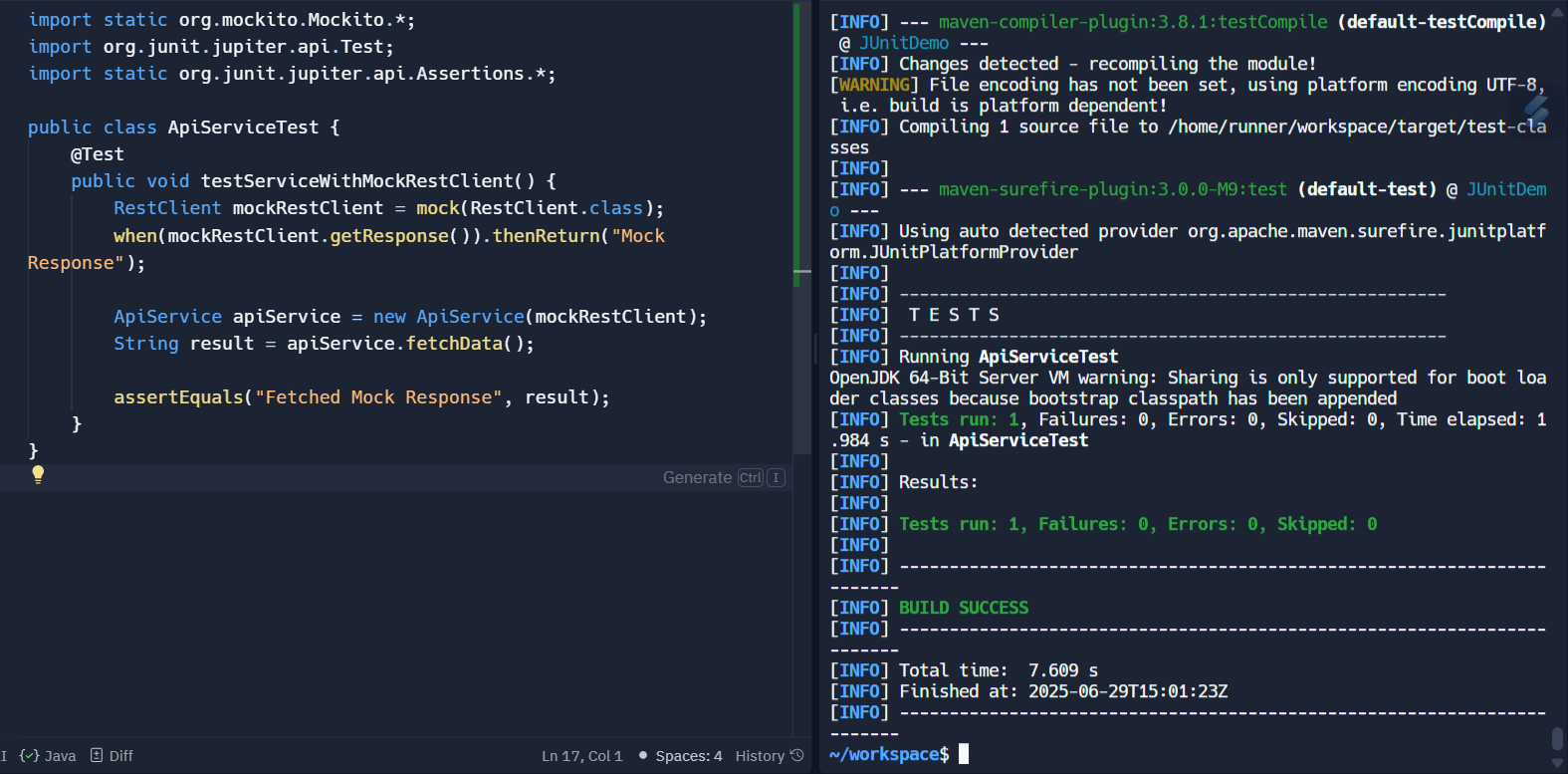
String result = apiService.fetchData();

assertEquals("Fetched Mock Response", result);

}

}

**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:**

public class FileService {

private FileReader fileReader;

private FileWriter fileWriter;

public FileService(FileReader fileReader, FileWriter fileWriter) {

this.fileReader = fileReader;

this.fileWriter = fileWriter;

}

public String processFile() {

String content = fileReader.read();

String processed = "Processed " + content;

fileWriter.write(processed);

return processed;

}

}

**FileServiceTest.java:**

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();

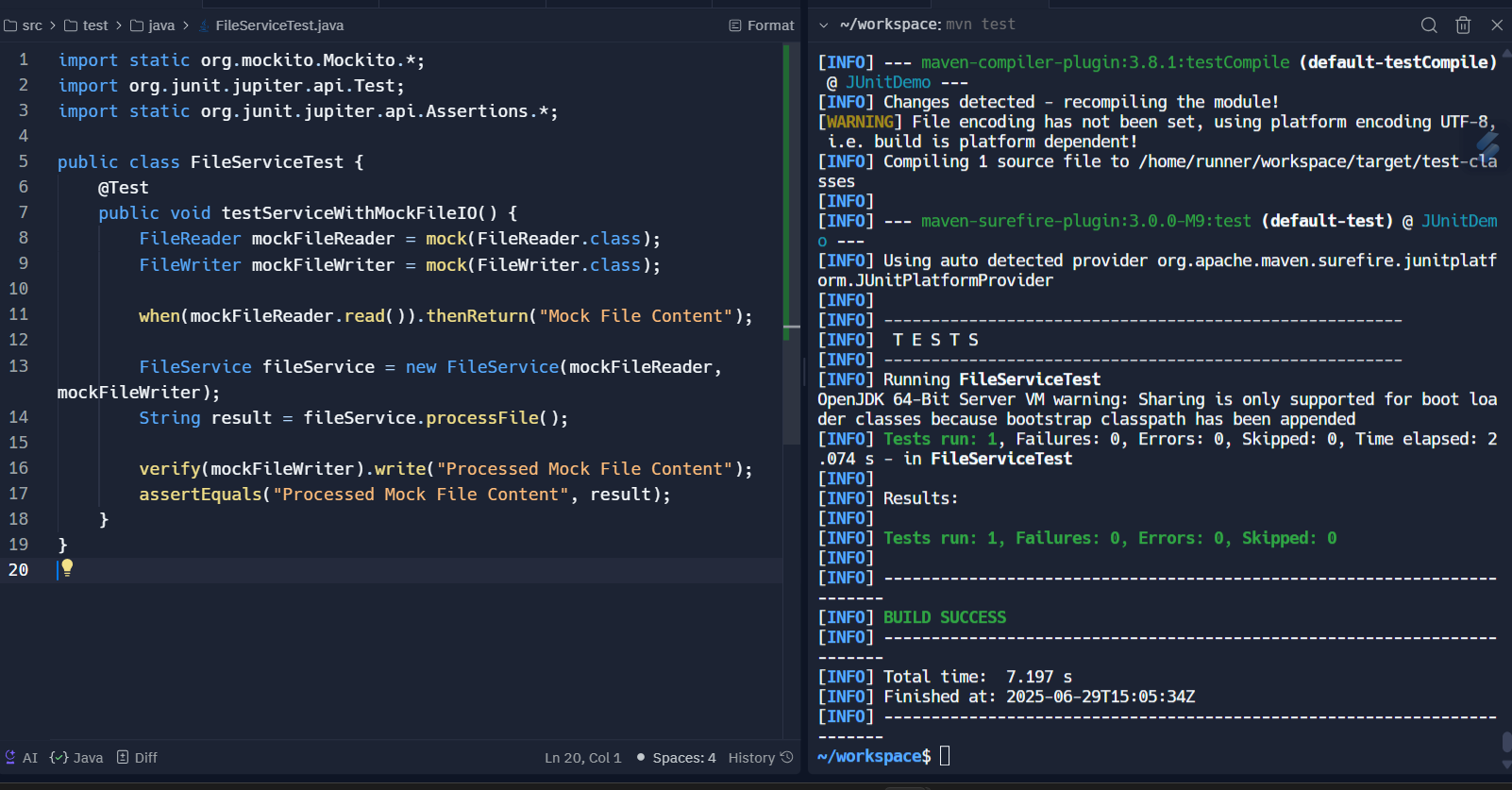
verify(mockFileWriter).write("Processed Mock File Content");

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

}

}

**Output:**

****

**Exercise 4: Mocking Network Interactions**

**Code:**

**NetworkClient.java:**

public interface NetworkClient {

String connect();

}

**NetworkService.java:**

public class NetworkService {

private NetworkClient networkClient;

public NetworkService(NetworkClient networkClient) {

this.networkClient = networkClient;

}

public String connectToServer() {

String connection = networkClient.connect();

return "Connected to " + connection;

}

}

**NetworkServiceTest.java:**

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);

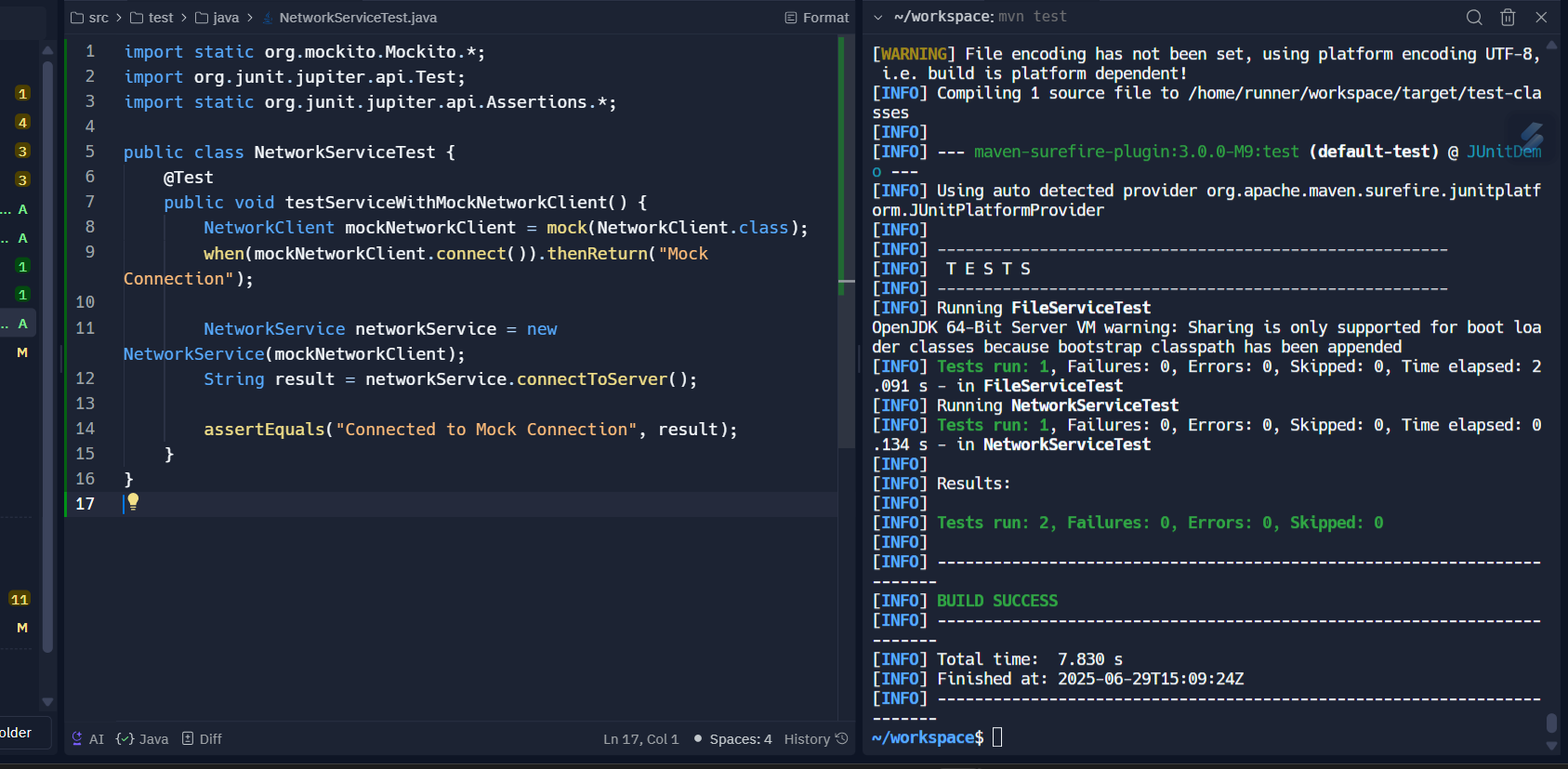
String result = networkService.connectToServer();

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

}

}

**Output:**

****

**Exercise 5: Mocking Multiple Return Values**

**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() {

String data = repository.getData();

return "Processed " + data;

}

}

**MultiReturnServiceTest.java:**

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();

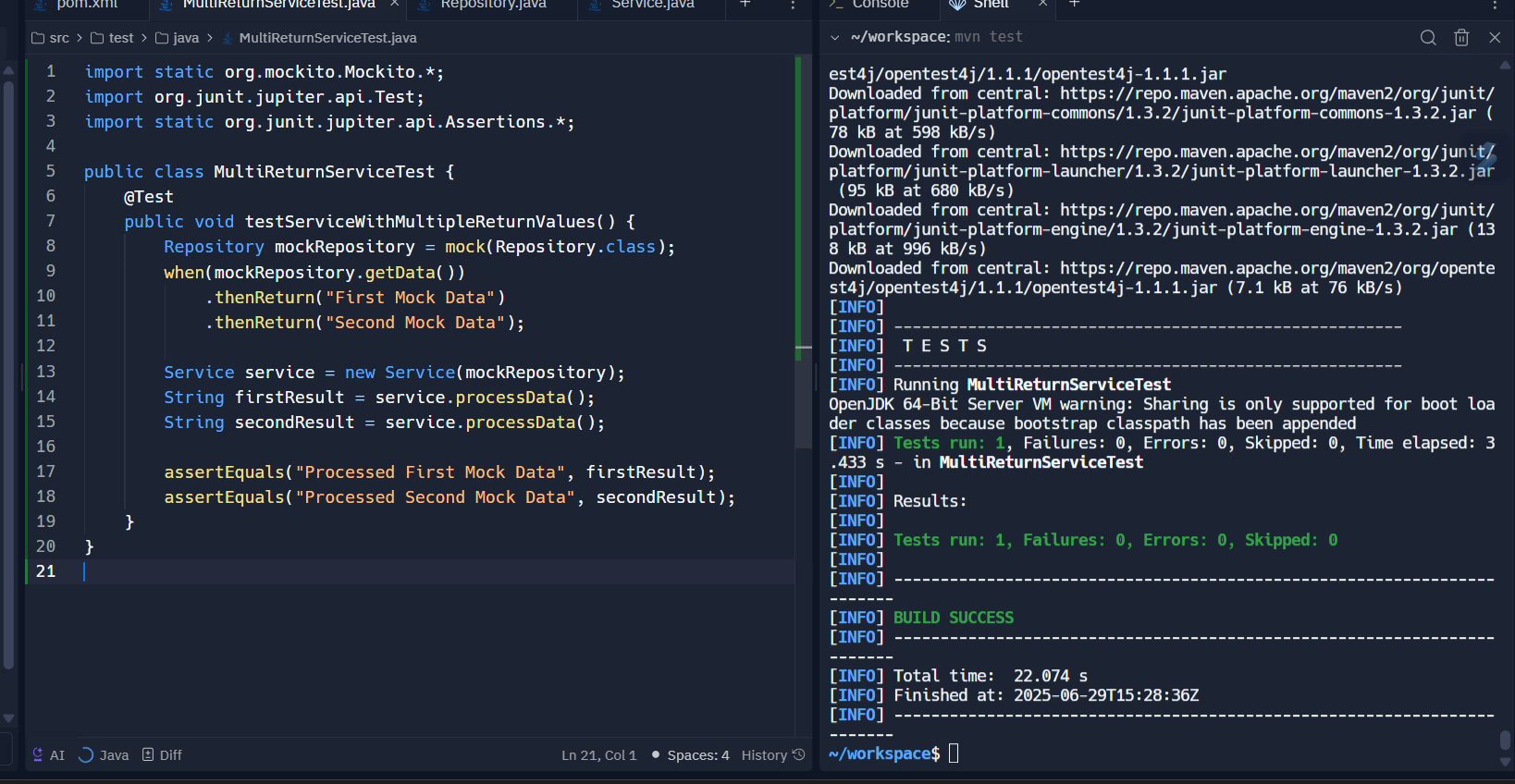
assertEquals("Processed First Mock Data", firstResult);

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

}

}

**Output:**

****