**Advanced Mockito Hands-On Exercises**

**Exercise 1: Mocking Databases and Repositories**

**Repository.java**

public interface Repository {

    String getData();

}

**Service.java**

public class Service {

private final Repository repository;

public Service(Repository repository) {

this.repository = repository;

}

public String processData() {

String data = repository.getData();

return "Processing" + data;

}

}

**ServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

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();

    }

}

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>

    <artifactId>demo</artifactId>

    <version>1.0-SNAPSHOT</version>

    <properties>

        <maven.compiler.source>21</maven.compiler.source>

        <maven.compiler.target>21</maven.compiler.target>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.junit.jupiter</groupId>

            <artifactId>junit-jupiter-api</artifactId>

            <version>5.10.0</version>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>org.junit.jupiter</groupId>

            <artifactId>junit-jupiter-engine</artifactId>

            <version>5.10.0</version>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>org.mockito</groupId>

            <artifactId>mockito-core</artifactId>

            <version>5.11.0</version>

            <scope>test</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.apache.maven.plugins</groupId>

                <artifactId>maven-surefire-plugin</artifactId>

                <version>3.2.5</version>

                <configuration>

                    <useSystemOut>true</useSystemOut>

                    <redirectTestOutputToFile>false</redirectTestOutputToFile>

                    <printSummary>true</printSummary>

                </configuration>

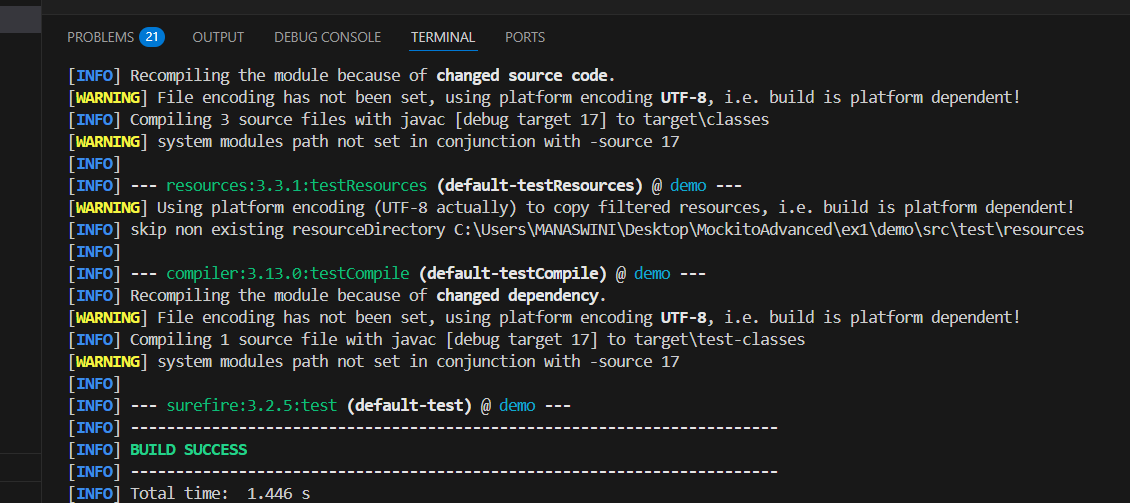
            </plugin>

        </plugins>

    </build>

</project>

**Output:**



Exercise 2: Mocking External Services (RESTful APIs)

**RestClient.java**

public interface RestClient {

    String getResponse();

}

**ApiService.java**

public class ApiService {

    private final RestClient restClient;

    public ApiService(RestClient restClient) {

        this.restClient = restClient;

    }

    public String fetchData() {

        String response = restClient.getResponse();

        return "Fetched " + response;

    }

}

**ApiServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

public class ApiServiceTest {

    @Test

    public void testServiceWithMockRestClient() {

        RestClient m = mock(RestClient.class);

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

        ApiService apiService = new ApiService(m);

        String result = apiService.fetchData();

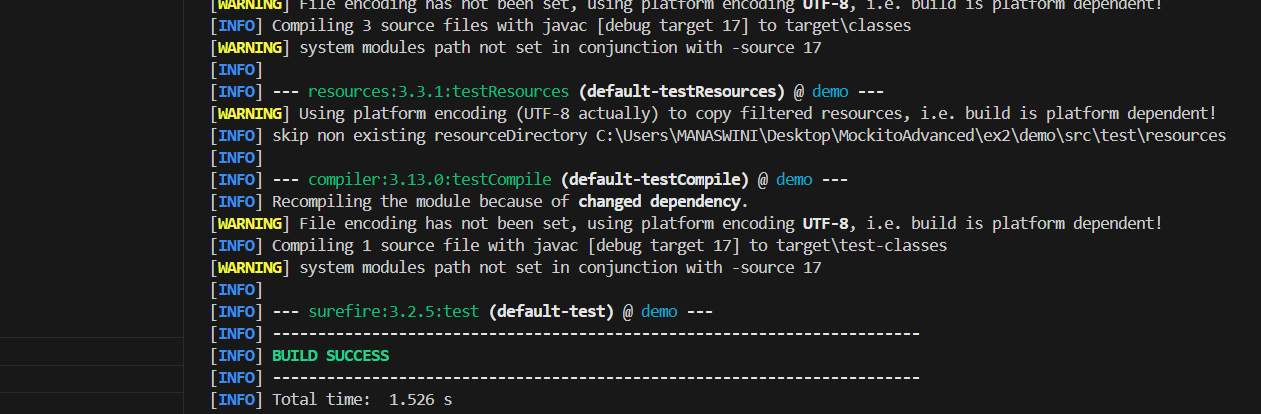
        assertEquals("Fetched result", result);

        verify(m).getResponse();

    }

}

**Output:**



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

**FileReader.java**

public interface FileReader {

    String read();

}

**FileWriter.java**

public interface FileWriter {

void write(String content);

}

**FileService.java**

public class FileService {

    private final FileReader fileReader;

    private final FileWriter fileWriter;

    public FileService(FileReader fr, FileWriter fw) {

        this.fileReader = fr;

        this.fileWriter = fw;

    }

    public String processFile() {

        String content = fileReader.read();

        String processedContent = "Processed " + content;

        fileWriter.write(processedContent);

        return processedContent;

    }

}

**FileServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

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();

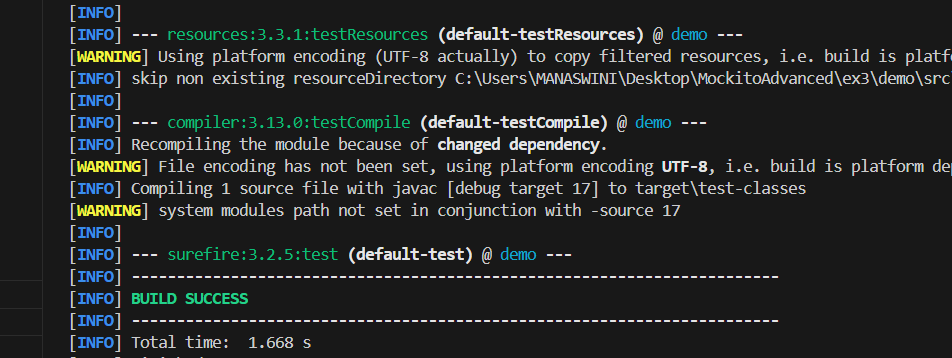
        assertEquals("Processed Mock File Content", result);

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

    }

}

**Output:**



**Exercise 4: Mocking Network Interactions**

**NetworkClient.java**

public interface NetworkClient {

    String connect();

}

**NetworkService.java**

public class NetworkService {

    private final NetworkClient networkClient;

    public NetworkService(NetworkClient networkClient) {

        this.networkClient = networkClient;

    }

    public String connectToServer() {

        String response = networkClient.connect();

        return "Connected to " + response;

    }

}

**NetworkServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

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();

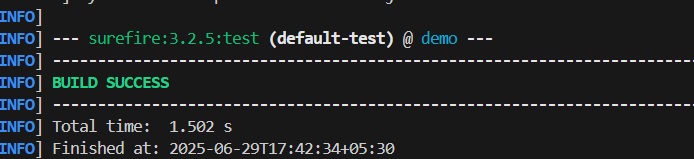
        assertEquals("Connected to Mock Connection", result);

        verify(mockNetworkClient).connect();

    }

}

**Output:**



**Exercise 5: Mocking Multiple Return Values**

**Repository.java**

public interface Repository {

    String getData();

}

**Service.java**

public class Service {

    private final Repository repository;

    public Service(Repository repository) {

        this.repository = repository;

    }

    public String processData() {

        return "Processed " + repository.getData();

    }

}

**ServiceTest.java**

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

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);

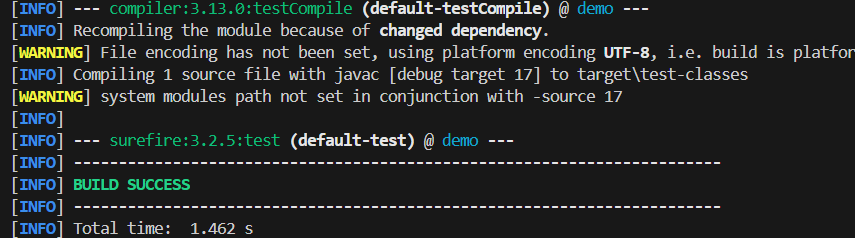
        assertEquals("Processed Second Mock Data", secondResult);

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

    }

}

**Output:**

****