**Mockito Excercises**

**Exercise 1: Mocking and Stubbing**

package com.example;

import static org.mockito.Mockito.\*;

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

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

// External API interface

interface ExternalApi {

    String getData();

}

// Service that depends on ExternalApi

class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public String fetchData() {

        return api.getData();

    }

}

public class MyServiceTest {

    @Test

    public void testExternalApi() {

        // 1. Create a mock object for the external API

        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        // 2. Stub the methods to return predefined values

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

        // 3. Write a test case that uses the mock object

        MyService service = new MyService(mockApi);

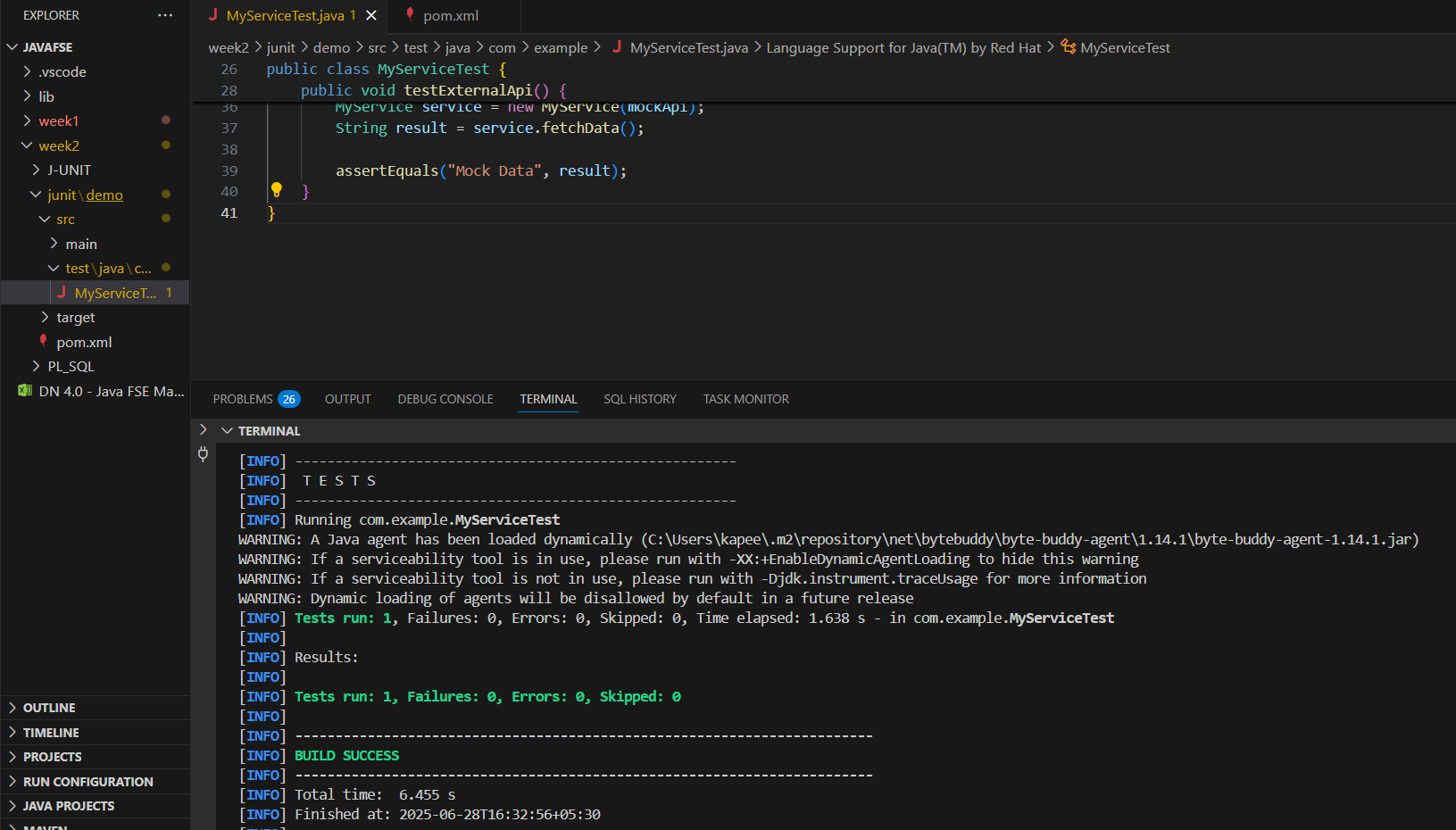
        String result = service.fetchData();

        assertEquals("Mock Data", result);

    }

}

Output:



Exercise 2: Verifying Interactions

**MyServiceTestRun.java**

package com.example;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

import static org.mockito.Mockito.\*;

// External API interface

interface ExternalApi {

    String getData();

    void sendData(String data);

}

// Service that depends on ExternalApi

class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public void fetchData() {

        api.getData();

    }

    public void send(String data) {

        api.sendData(data);

    }

}

public class MyServiceTestRun {

    @Test

    public void testVerifyInteraction() {

        //mock project is initialized and created

        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        //to call the methods

        MyService service = new MyService(mockApi);

        service.fetchData();

        verify(mockApi).getData();

    }

    @Test

    public void testVerifySendDataWithArgument() {

        ExternalApi mockApi = Mockito.mock(ExternalApi.class);

        MyService service = new MyService(mockApi);

        service.send("Hello");

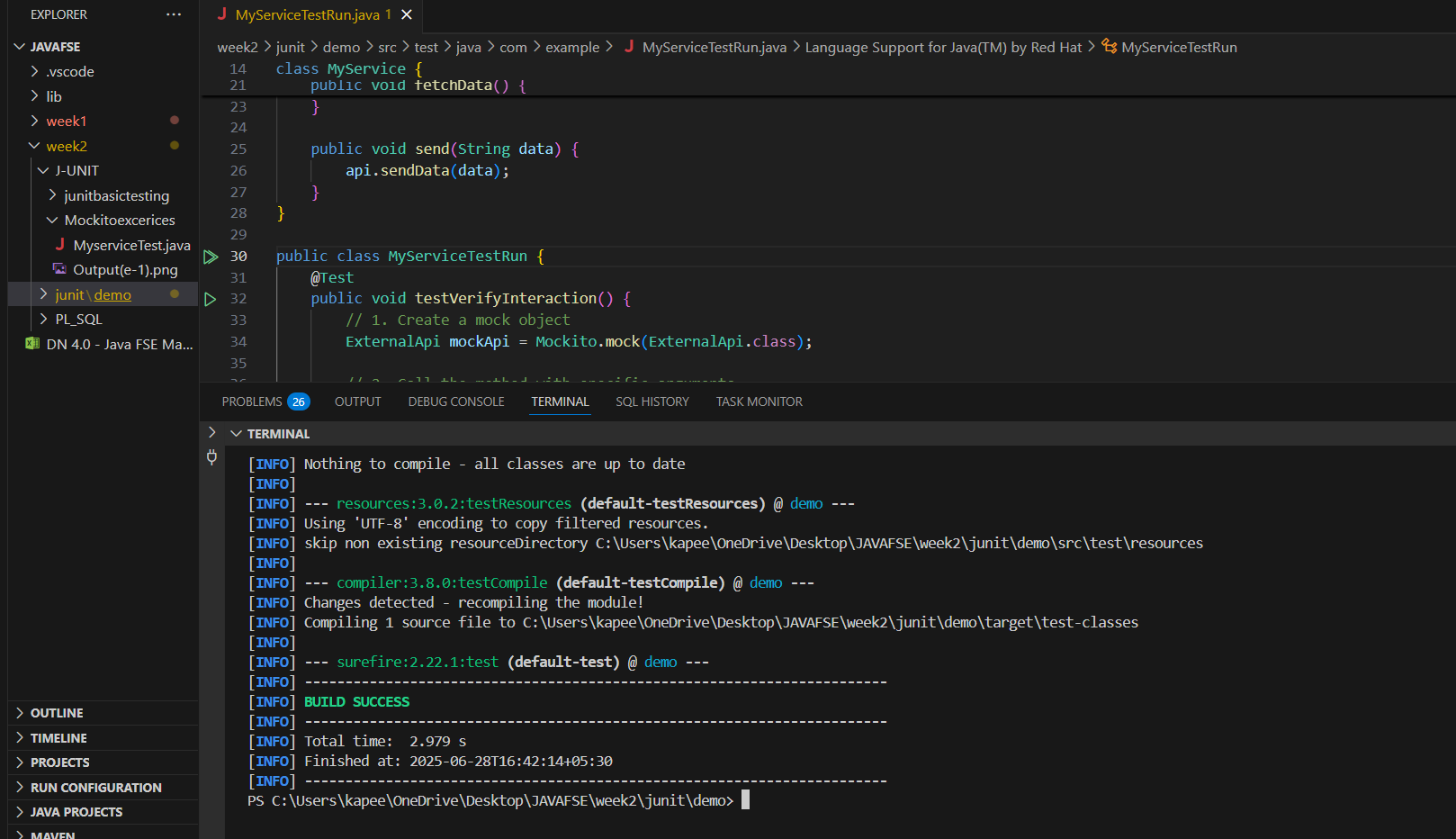
        // Verify sendData was called with "Hello"

        verify(mockApi).sendData("Hello");

    }

}

**Output:**



Exercise 3: Argument Matching

**ArgumentMatcherTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.\*;

// External API interface

interface ExternalApi {

    void process(String data, int count);

}

// Service that depends on ExternalApi

class MyService {

    private ExternalApi api;

    public MyService(ExternalApi api) {

        this.api = api;

    }

    public void doWork() {

        api.process("TestData", 5);

    }

}

public class ArgumentMatcherTest {

    @Test

    public void testVerifyWithArgumentMatchers() {

        // 1. Create a mock object

        ExternalApi mockApi = mock(ExternalApi.class);

        // 2. Call the method with specific arguments

        MyService service = new MyService(mockApi);

        service.doWork();

        // 3. Use argument matchers to verify the interaction

        verify(mockApi).process(eq("TestData"), anyInt());

    }

}

Output:

