Mockito exercises

Exercise 1: Mocking and Stubbing

You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

**ExternalApi.java:**

package com.example.junit\_test\_project2;

public interface ExternalApi {

String getData();

}

**MyService.java:**

package com.example.junit\_test\_project2;

public class MyService {

private ExternalApi externalApi;

public MyService(ExternalApi externalApi) {

this.externalApi = externalApi;

}

public String fetchData() {

return externalApi.getData();

}

}

**MyServiceTest.java:**

package com.example.junit\_test\_project2;

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

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class MyServiceTest {

@Test

public void testExternalApi() {

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

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

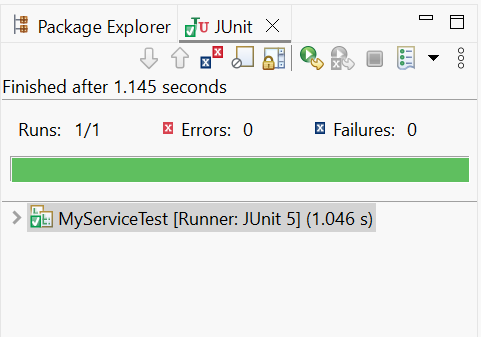
MyService service = new MyService(mockApi);

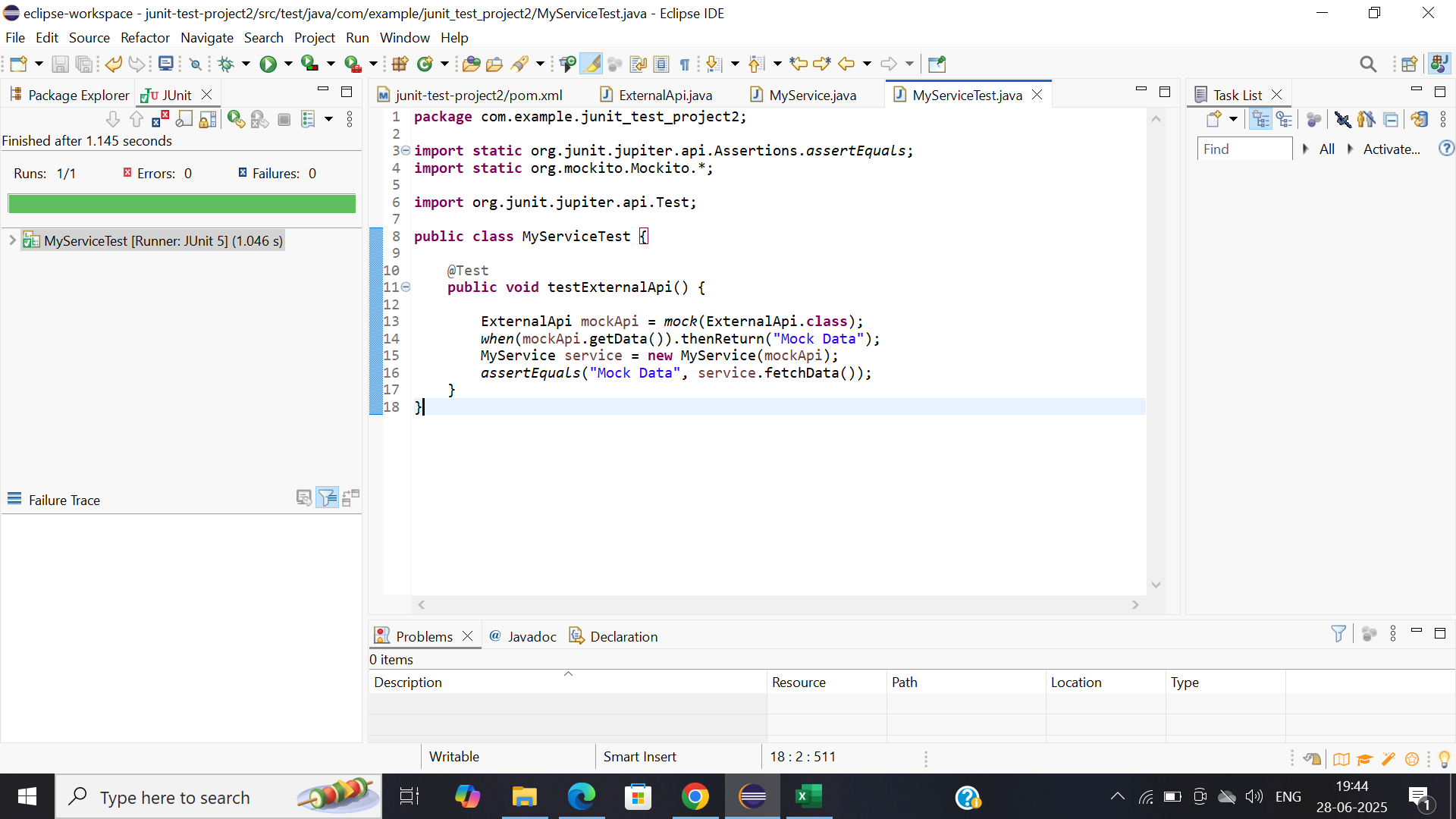
*assertEquals*("Mock Data", service.fetchData());

}

}

OUTPUT:





Exercise 2: Verifying Interactions

Scenario:

You need to ensure that a method is called with specific arguments.

**UserApi.java:**

package com.example.junit\_test\_project2;

public interface UserApi {

String getInfo();

String getUserInfo(String userId);

}

**UserService.java:**

package com.example.junit\_test\_project2;

public class UserService {

private UserApi userApi;

public UserService(UserApi userApi) {

this.userApi = userApi;

}

public String fetchInfo() {

return userApi.getInfo();

}

public String fetchUser(String userId) {

return userApi.getUserInfo(userId);

}

}

**UserServiceTest.java:**

package com.example.junit\_test\_project2;

import static org.mockito.Mockito.\*;

import org.junit.jupiter.api.Test;

public class UserServiceTest {

@Test

public void testVerifySimpleInteraction() {

UserApi mockApi = *mock*(UserApi.class);

UserService service = new UserService(mockApi);

service.fetchInfo();

*verify*(mockApi).getInfo();

}

@Test

public void testVerifyMethodWithArgument() {

UserApi mockApi = *mock*(UserApi.class);

UserService service = new UserService(mockApi);

service.fetchUser("abc123");

*verify*(mockApi).getUserInfo("abc123");

}

}

OUTPUT:

