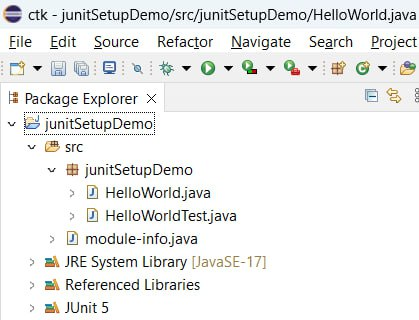
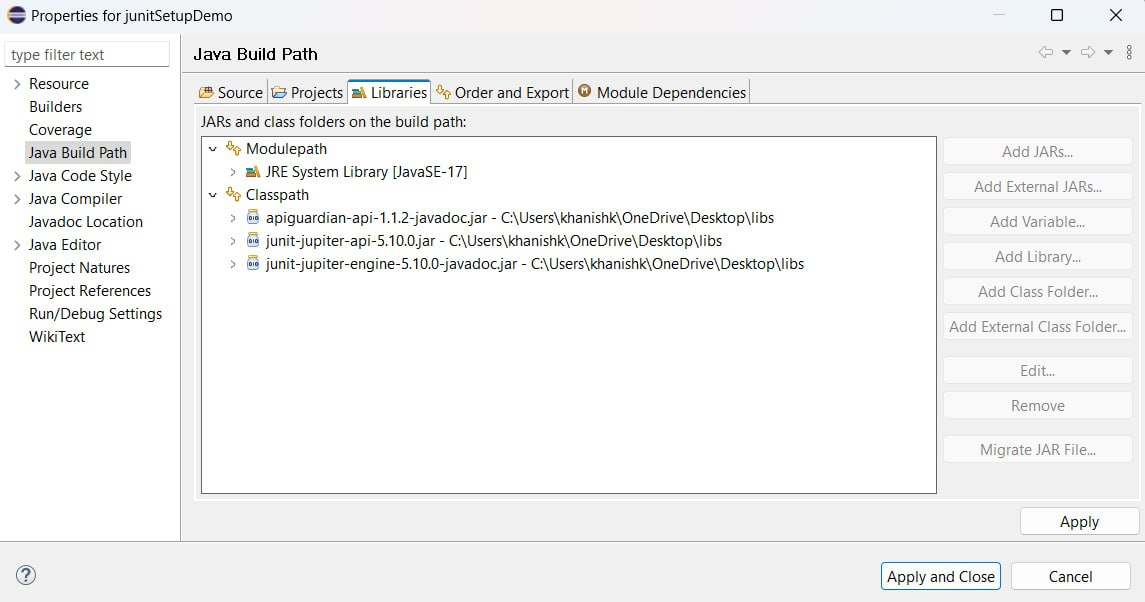
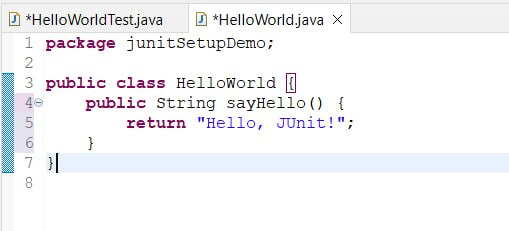
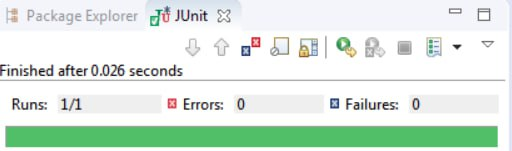
Exercise 1: Setting Up JUnit

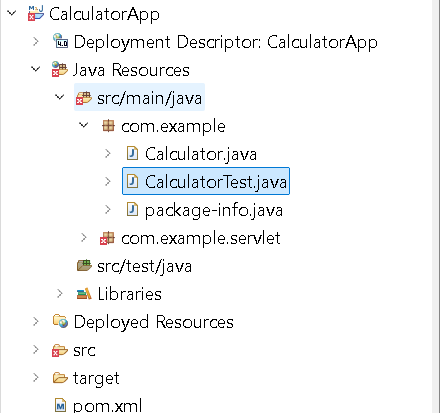


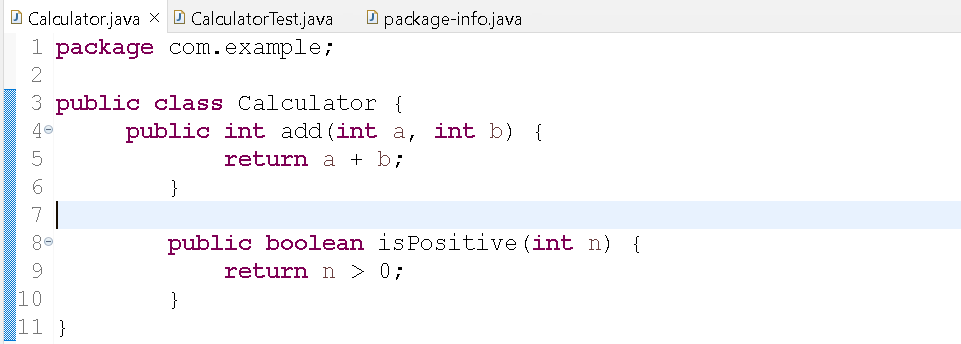


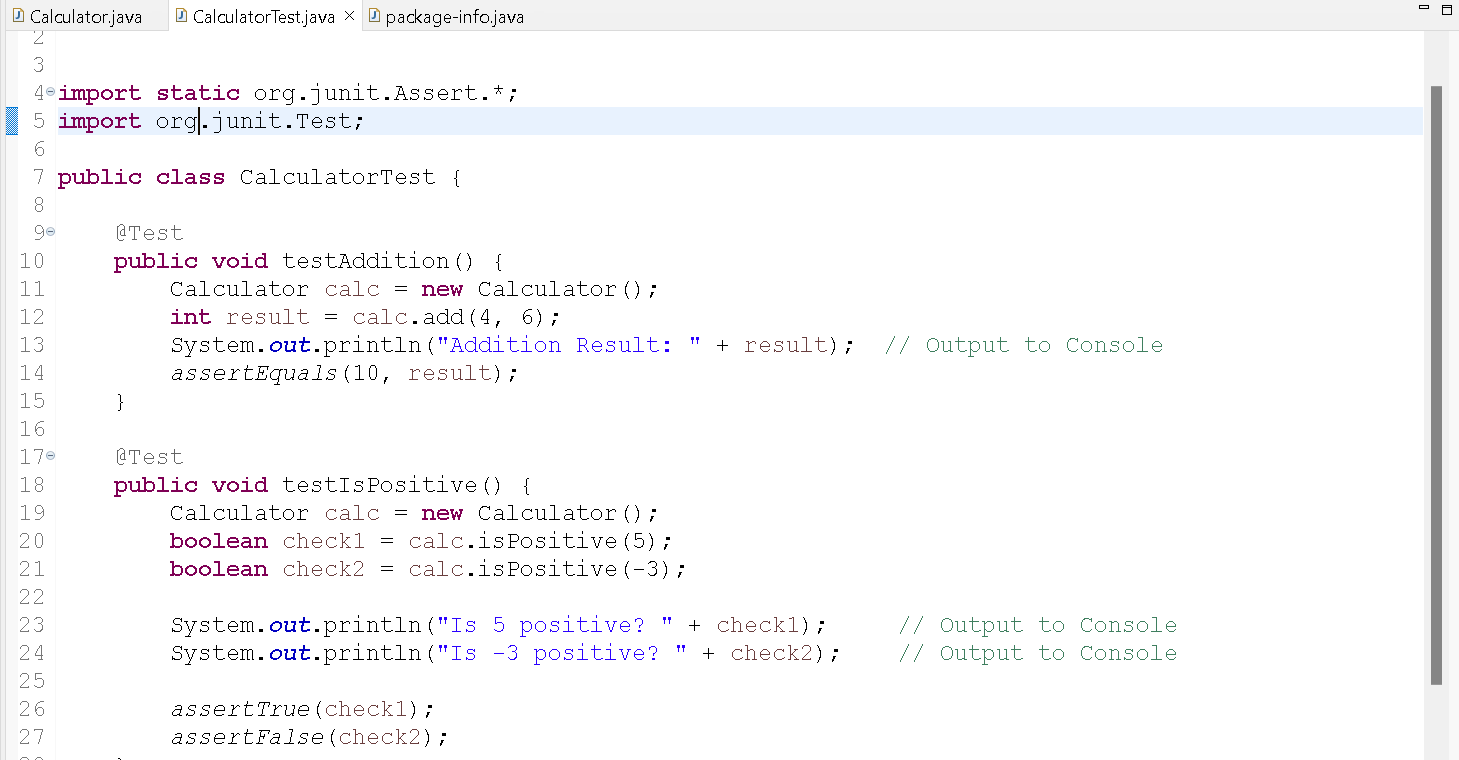


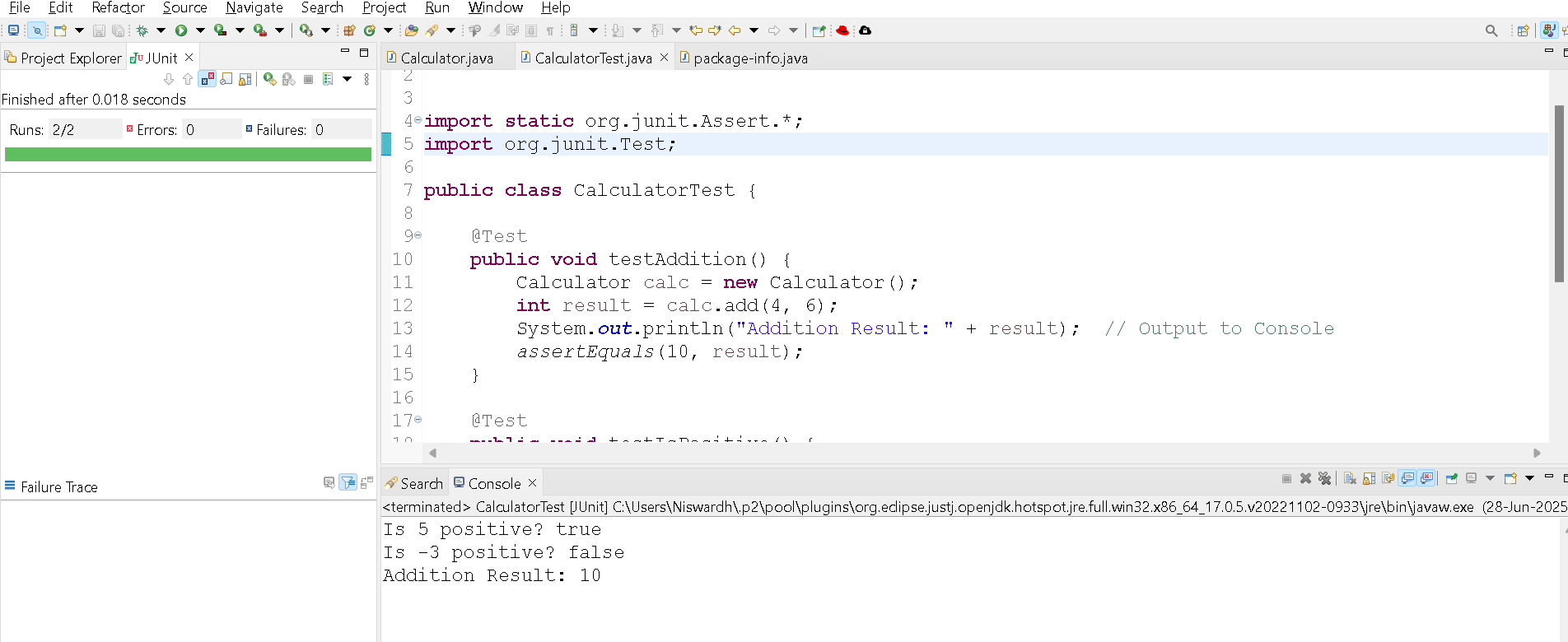


**Exercise 3: Assertions in Junit**

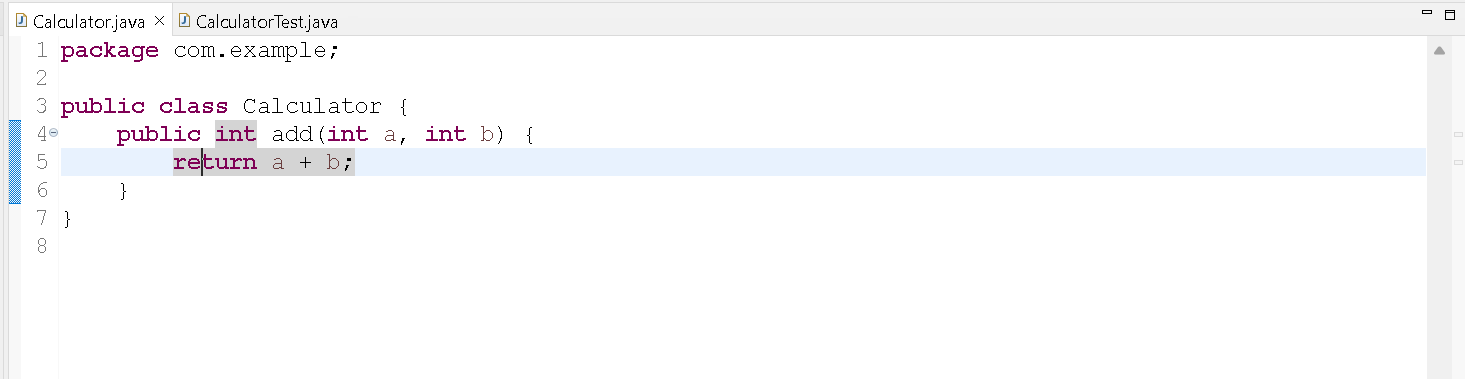


Calculator.java  


CalculatorTest.java  


Output  


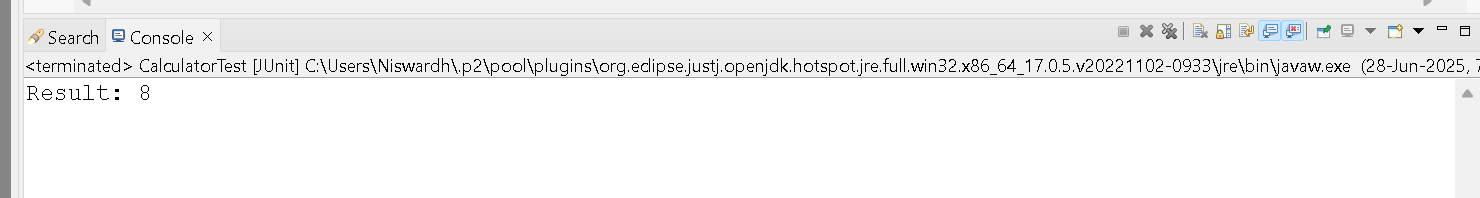
**Calculator.java**



**CalculatorTest.java**



Output:



# Exercise 1: Mocking and Stubbing (Mockito + JUnit 5)

## Objective

Use Mockito to mock an external API and stub its methods to return predefined values.

## Scenario

You have a service class (MyService) that calls an external API (ExternalApi). You want to mock the API and stub its getData() method to return a predefined string like 'Mock Data'. This allows you to test MyService independently of the real API.

## Step-by-Step Implementation

### Step 1: Create Interface (ExternalApi.java)

package com.example;  
  
public interface ExternalApi {  
 String getData();  
}

### Step 2: Create Service Class (MyService.java)

package com.example;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

### Step 3: Create Test Class (MyServiceTest.java)

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;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 when(mockApi.getData()).thenReturn("Mock Data");  
  
 MyService service = new MyService(mockApi);  
 String result = service.fetchData();  
  
 assertEquals("Mock Data", result);  
 }  
}

### Step 4: Run in Eclipse

1. Right-click MyServiceTest.java

2. Choose 'Run As → JUnit Test'

3. You should see a green bar (test passed)

## Expected Output

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

**Exercise 2: Verifying Interactions (Mockito + JUnit 5)**

**Objective**

Ensure that a method (e.g., getData()) is called with specific arguments using Mockito.

**Scenario**

You have a service class (MyService) that calls a method getData() from an external API (ExternalApi). You want to verify that getData() was actually called when fetchData() is invoked.

**Step-by-Step Implementation**

**Step 1: Create Interface (ExternalApi.java)**

**package com.example;  
  
public interface ExternalApi {  
 String getData();  
}**

**Step 2: Create Service Class (MyService.java)**

**package com.example;  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}**

**Step 3: Create Test Class (MyServiceTest.java)**

**package com.example;  
import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
  
public class MyServiceTest {  
  
 @Test  
 public void testVerifyInteraction() {  
 ExternalApi mockApi = mock(ExternalApi.class);  
 MyService service = new MyService(mockApi);  
 service.fetchData();  
 verify(mockApi).getData();  
 }  
}**

**Step 4: Run in Eclipse**

1. Right-click MyServiceTest.java

2. Choose 'Run As → JUnit Test'

3. You should see a green bar (test passed)

**Expected Output**

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0