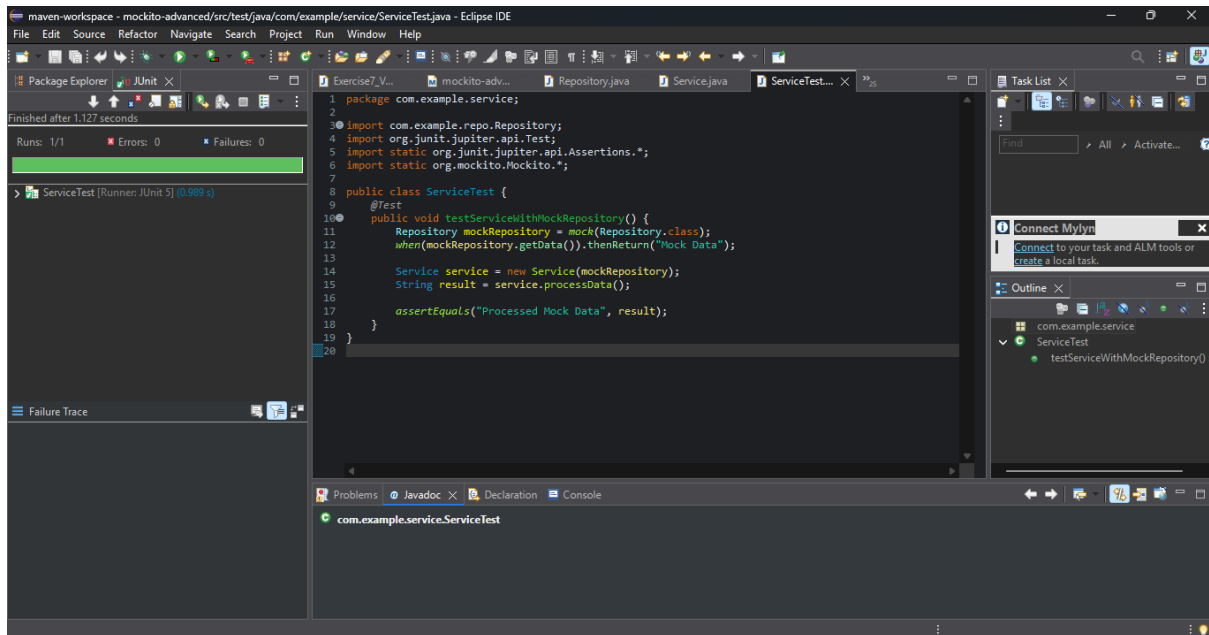


Advanced Mockito Output

Exercise 1: Mocking Databases and Repositories

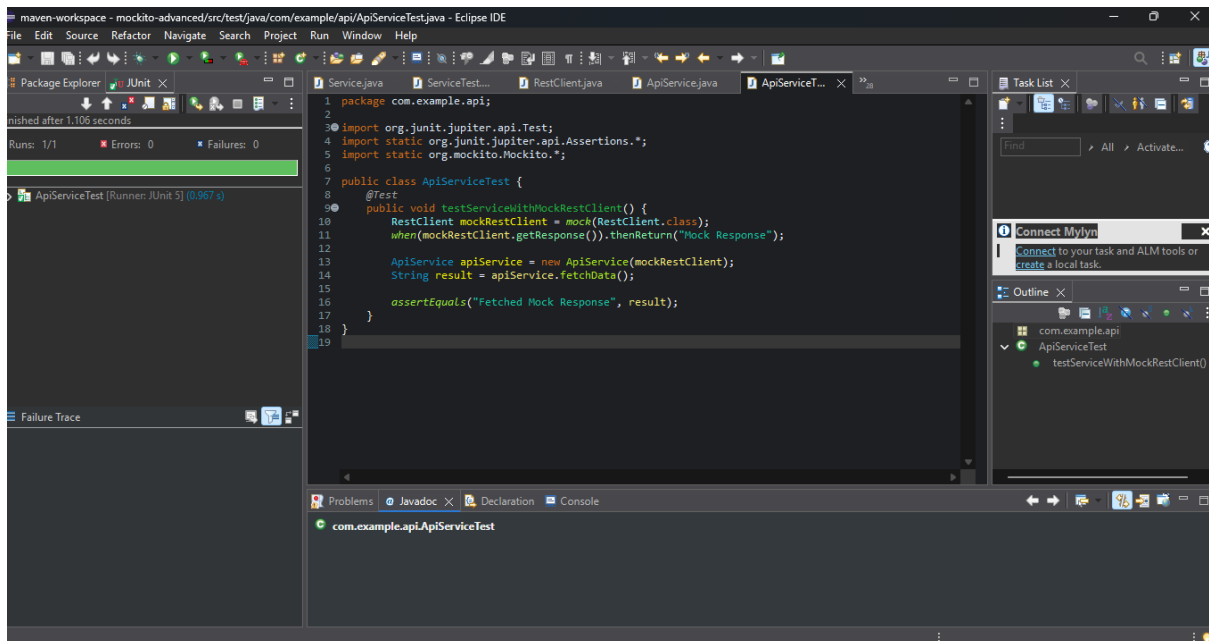
Output:



Mocking repositories helps isolate and validate business logic without relying on actual database calls.

Exercise 2: Mocking External Services (RESTful APIs)

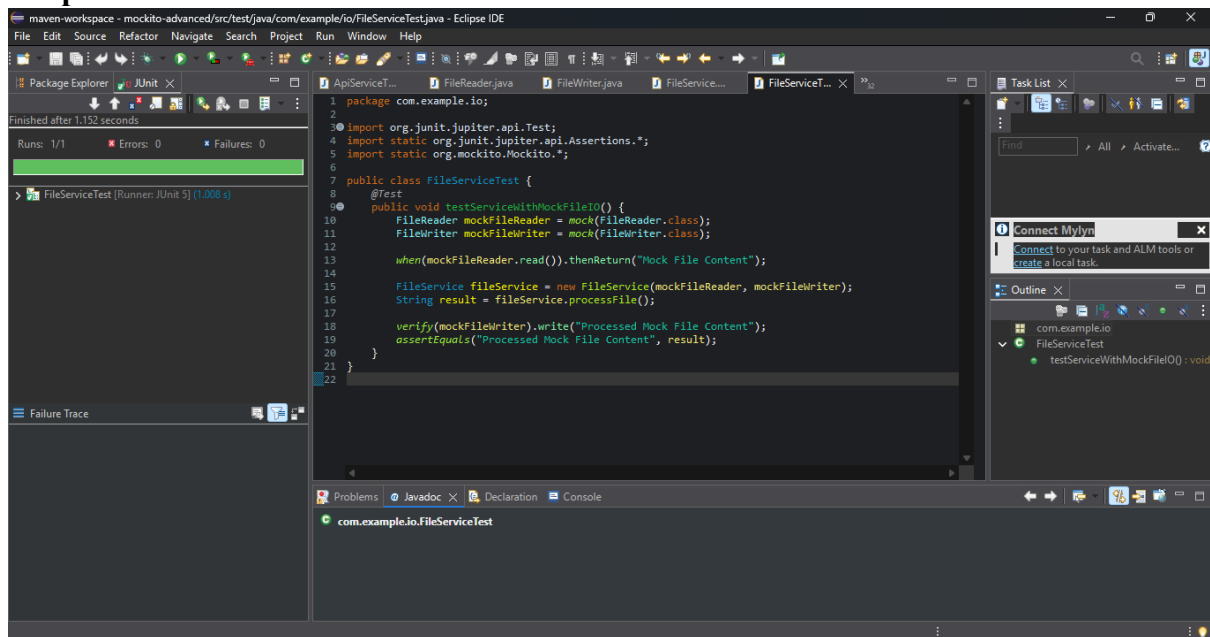
Output:



By mocking REST clients, you can simulate API responses and test service behavior without real HTTP calls.

Exercise 3: Mocking File I/O

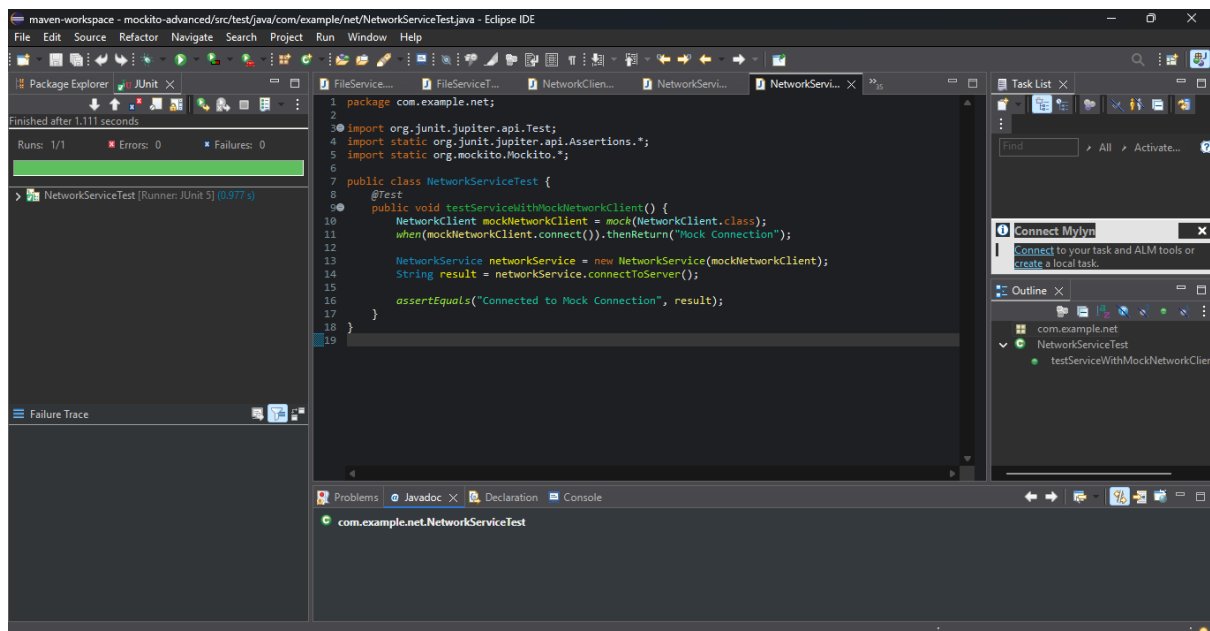
Output:



Mocking file readers/writers lets you test file-processing logic without touching the actual filesystem.

Exercise 4: Mocking Network Interactions

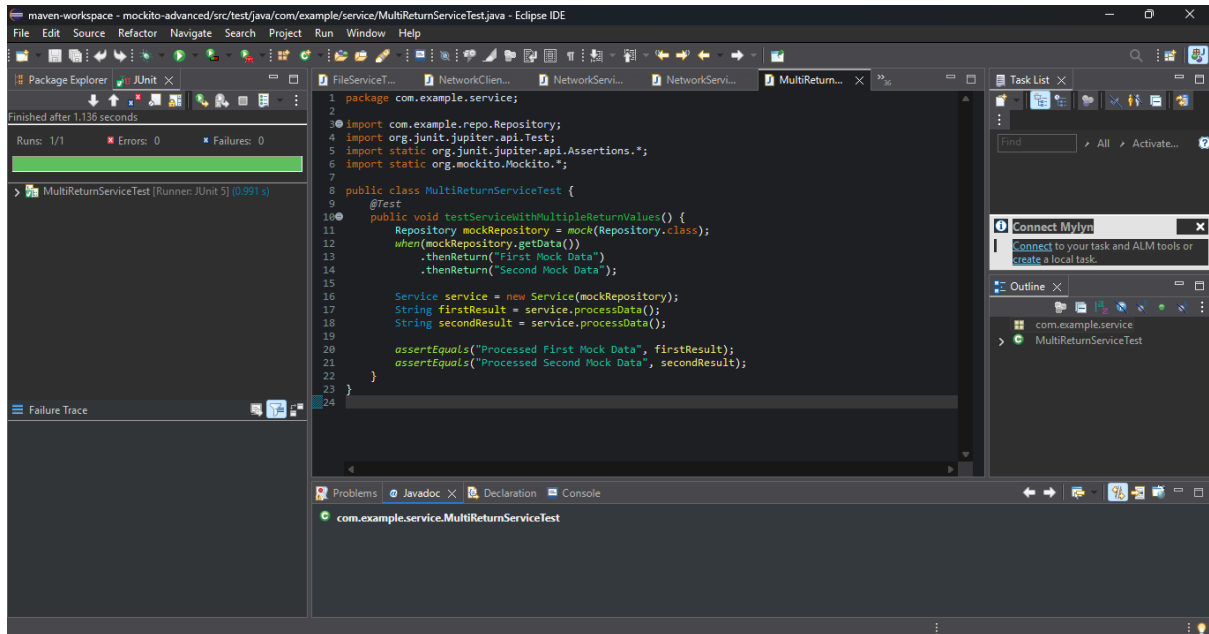
Output:



Simulating network clients avoids real network dependencies and ensures predictable test outcomes.

Exercise 5: Mocking Multiple Return Values

Output:



```
1 package com.example.service;
2
3 import com.example.repo.Repository;
4 import org.junit.jupiter.api.Test;
5 import static org.junit.jupiter.api.Assertions.*;
6 import static org.mockito.Mockito.*;
7
8 public class MultiReturnServiceTest {
9     @Test
10     public void testServiceWithMultipleReturnValues() {
11         Repository mockRepository = mock(Repository.class);
12         when(mockRepository.getData())
13             .thenReturn("First Mock Data")
14             .thenReturn("Second Mock Data");
15
16         Service service = new Service(mockRepository);
17         String firstResult = service.processData();
18         String secondResult = service.processData();
19
20         assertEquals("Processed First Mock Data", firstResult);
21         assertEquals("Processed Second Mock Data", secondResult);
22     }
23 }
24
```

Finished after 1.136 seconds

Runs: 1/1 Errors: 0 Failures: 0

MultiReturnServiceTest (Runner: JUnit 5) [0.991 s]

Failure Trace

com.example.service.MultiReturnServiceTest

Use consecutive stubs to test how services react to changing results from repeated method calls.