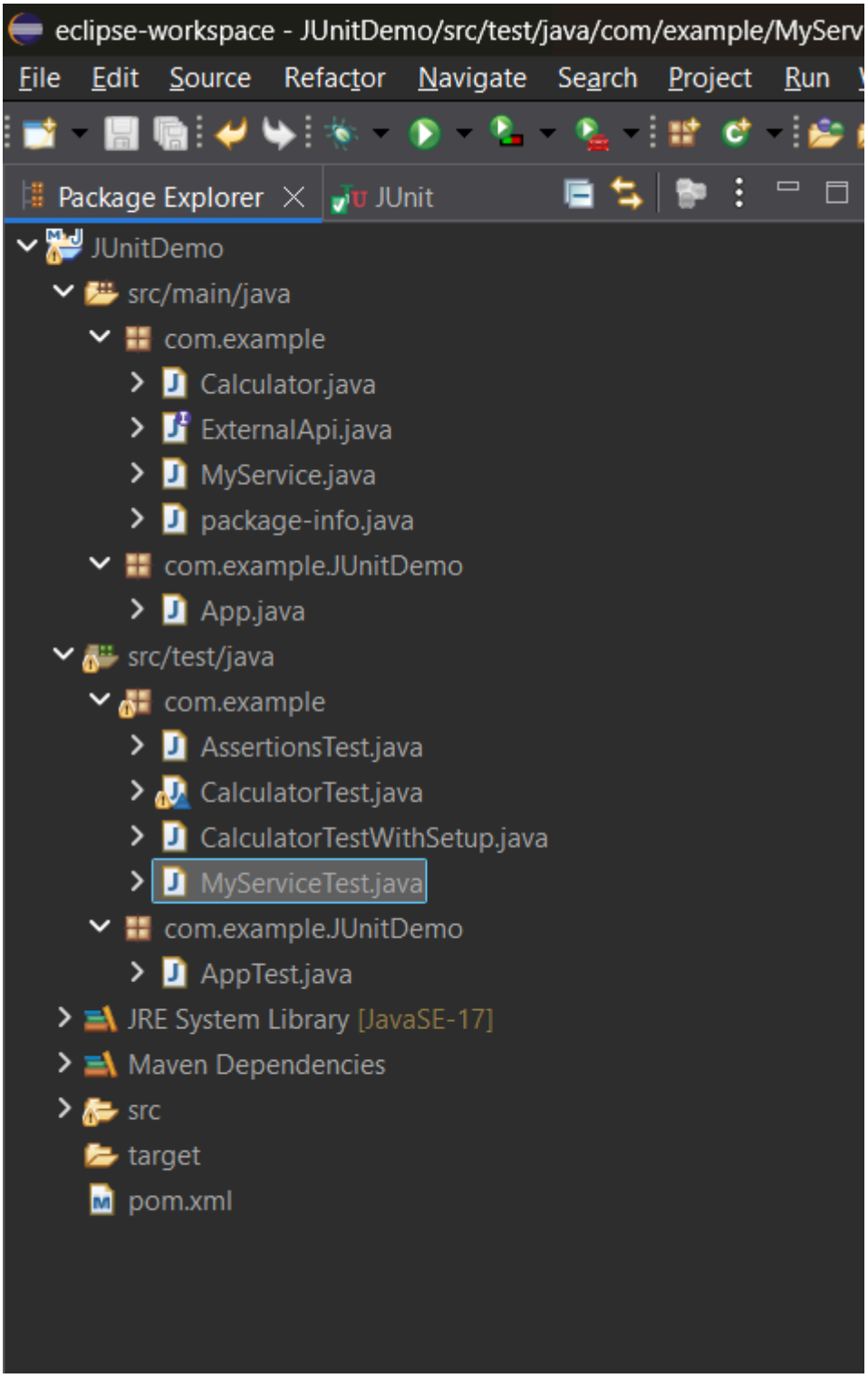
Mockito Hands-On Exercises

Exercise 1: Mocking and Stubbing

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

FOLDER STRUCTURE



ExternalApi.java

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

MyService.java

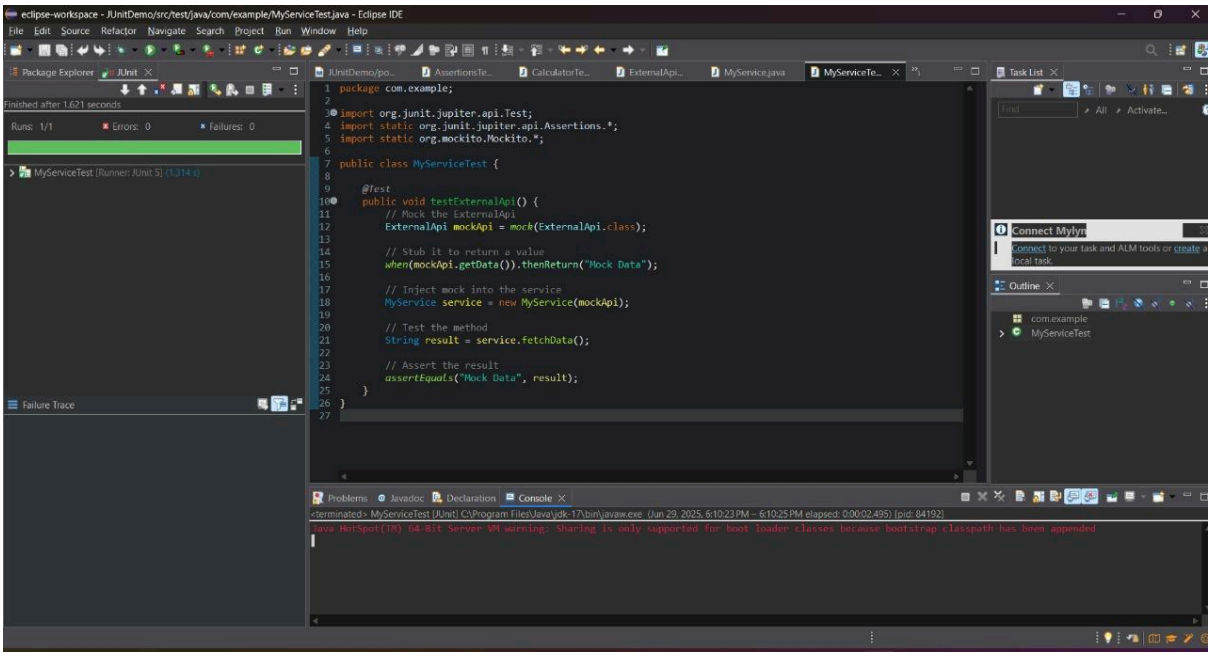
```
package com.example;
public class MyService {
   private ExternalApi api;
public MyService(ExternalApi api) {
      this.api = api;
   }
public String fetchData() {
      return api.getData();  // calls the external API
   }
}
```

MyServiceTest.java

```
package com.example;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.Mockito.*;
public class MyServiceTest {
   @Test
   public void testExternalApi() {
      // Mock the ExternalApi
      ExternalApi mockApi = mock(ExternalApi.class);
```

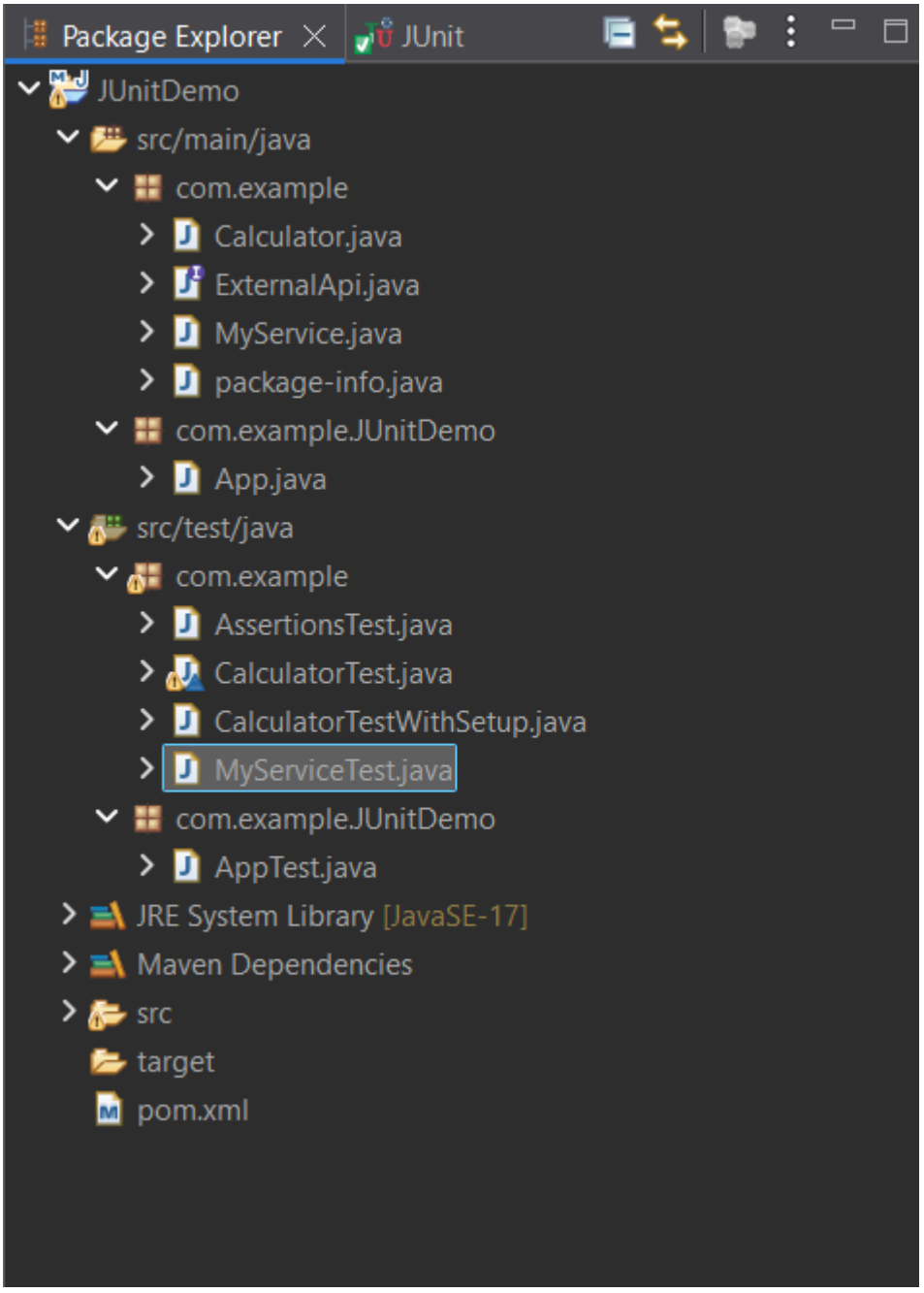
```
        when(mockApi.getData()).thenReturn("Mock Data");
        MyService service = new MyService(mockApi);
         String result = service.fetchData();

       assertEquals("Mock Data", result);
    }
}
```

OUTPUT



Exercise 2: Verifying Interactions

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



MyServiceTest.java

```
package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.*;
```

```
public class MyServiceTest {
 @Test
   public void testVerifyInteraction() {
 ExternalApi mockApi = mock(ExternalApi.class);
 MyService service = new MyService(mockApi);
 service.fetchData();
 verify(mockApi).getData();
   }
}
```

OUTPUT