

Exercise 2: Verifying Interactions

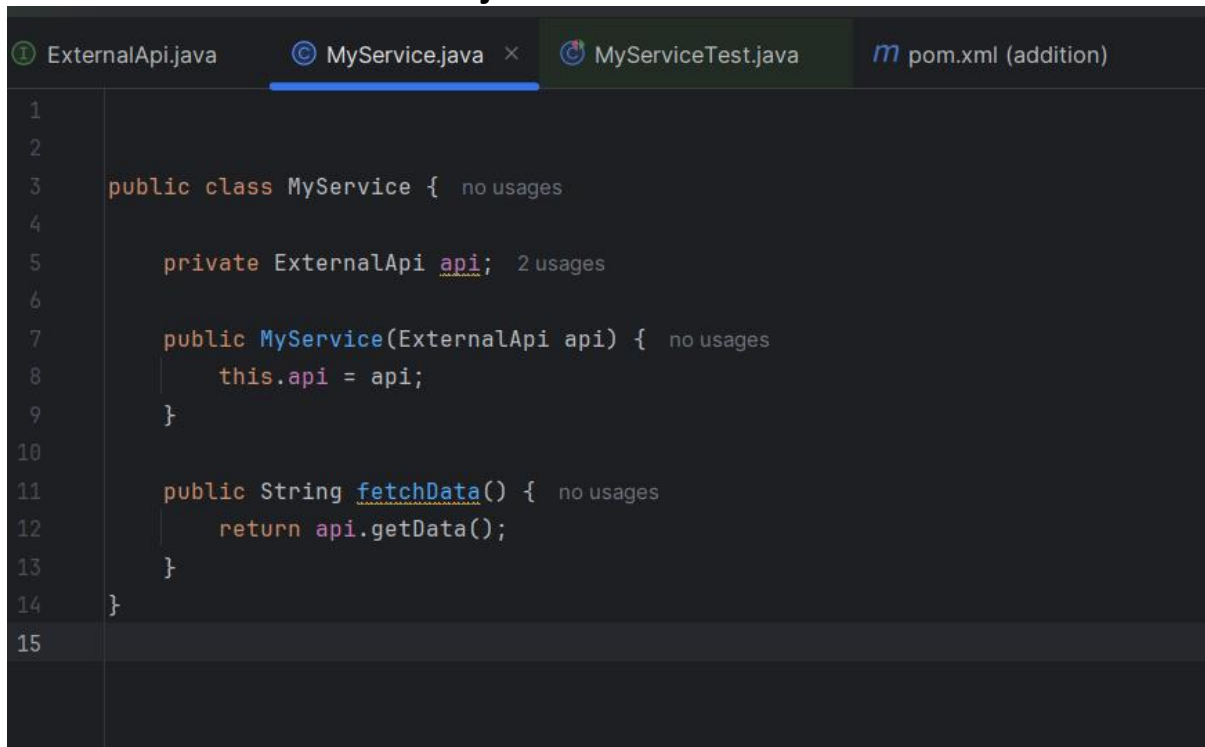
FROM FILE : Mockito Exercises

MyService.java : class

Code:

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

Practice screen from IntelliJ:



The screenshot shows the IntelliJ IDEA IDE with the 'MyService.java' file open. The code is syntax-highlighted and includes usage statistics for each line. The tabs at the top are 'ExternalApi.java', 'MyService.java' (active), 'MyServiceTest.java', and 'pom.xml (addition)'. The code is as follows:

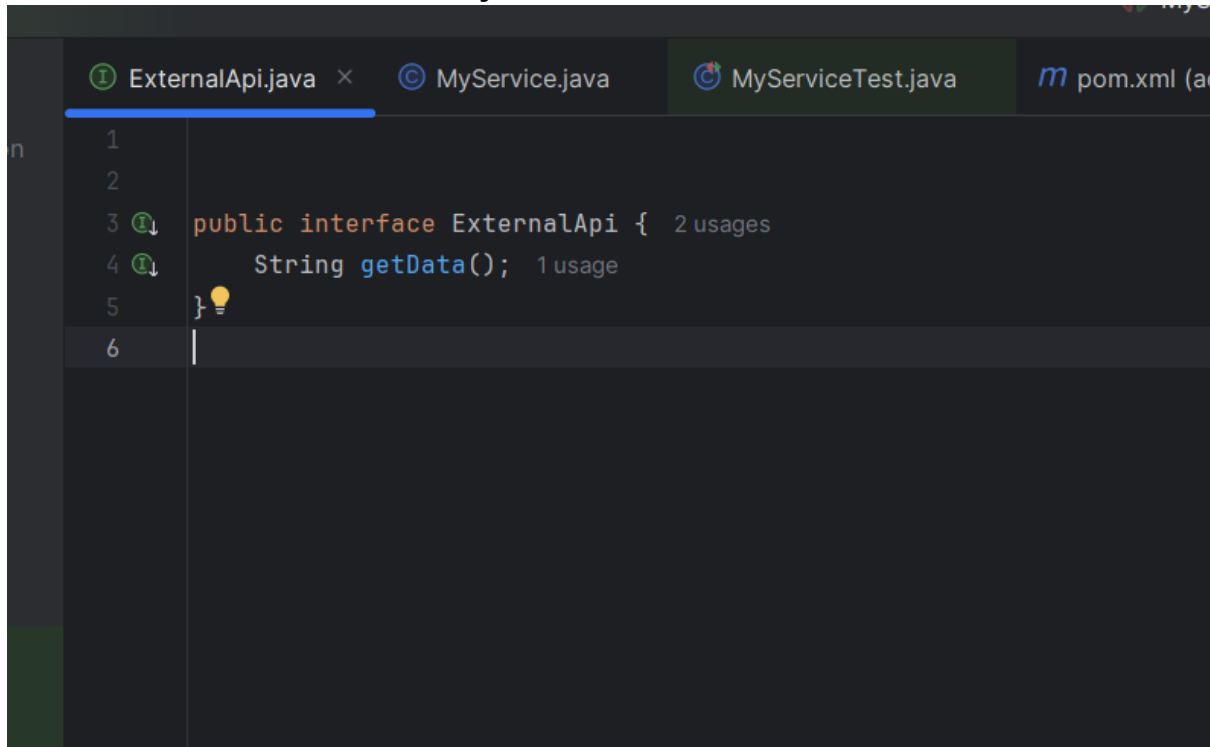
```
1  
2  
3 public class MyService { no usages  
4  
5     private ExternalApi api; 2 usages  
6  
7     public MyService(ExternalApi api) { no usages  
8         this.api = api;  
9     }  
10  
11     public String fetchData() { no usages  
12         return api.getData();  
13     }  
14 }  
15
```

ExternalApi.java class:

code:

```
public interface ExternalApi {  
    String getData();  
}
```

Practice screen from IntelliJ:



ServiceTest.java class:

Code:

```
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
import static org.mockito.Mockito.*;
```

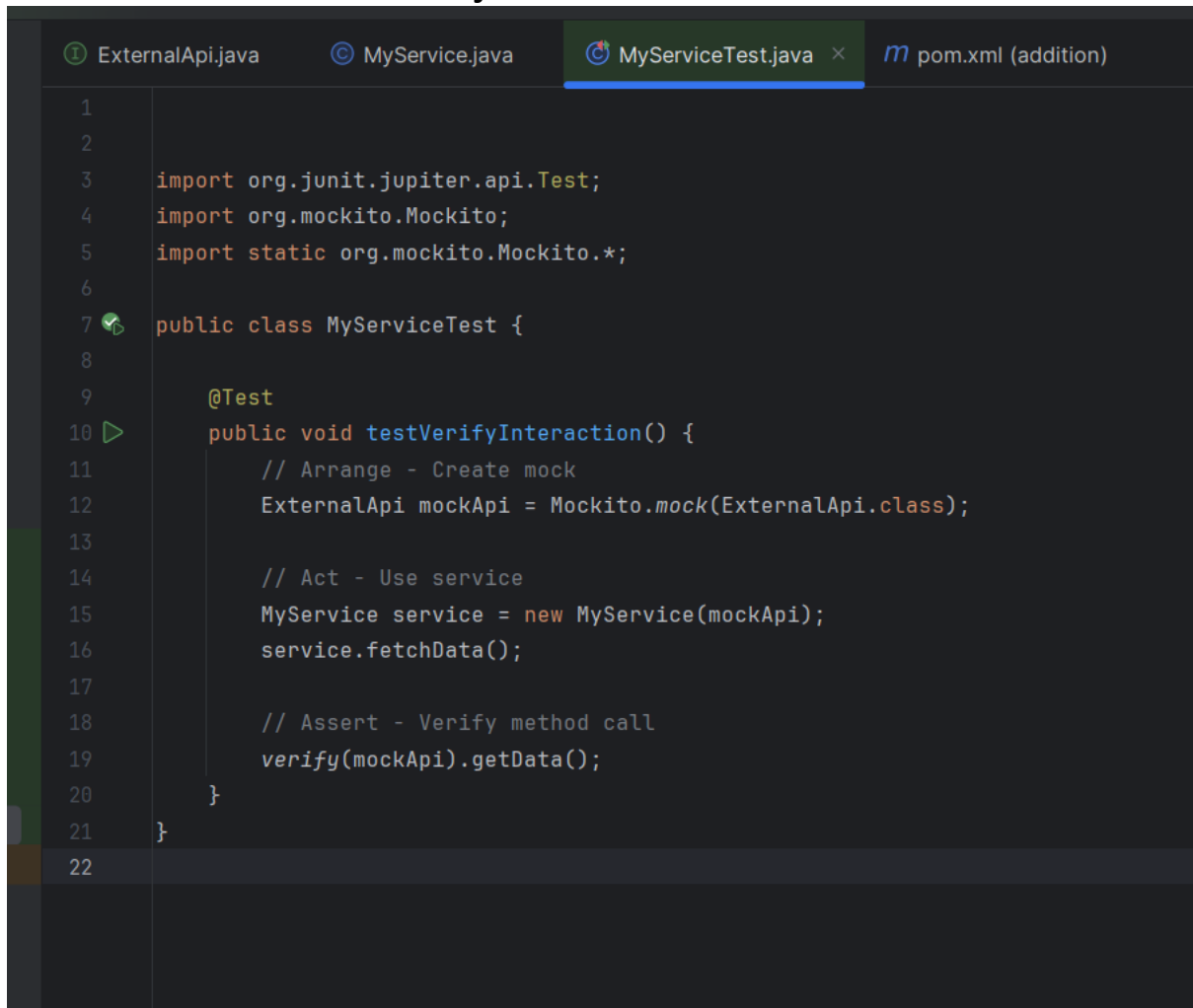
```
public class MyServiceTest {  
    @Test  
    public void testVerifyInteraction() {  
        ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
        MyService service = new MyService(mockApi);
```

```

        service.fetchData();
        verify(mockApi).getData();
    }
}

```

Practice screen from IntelliJ:



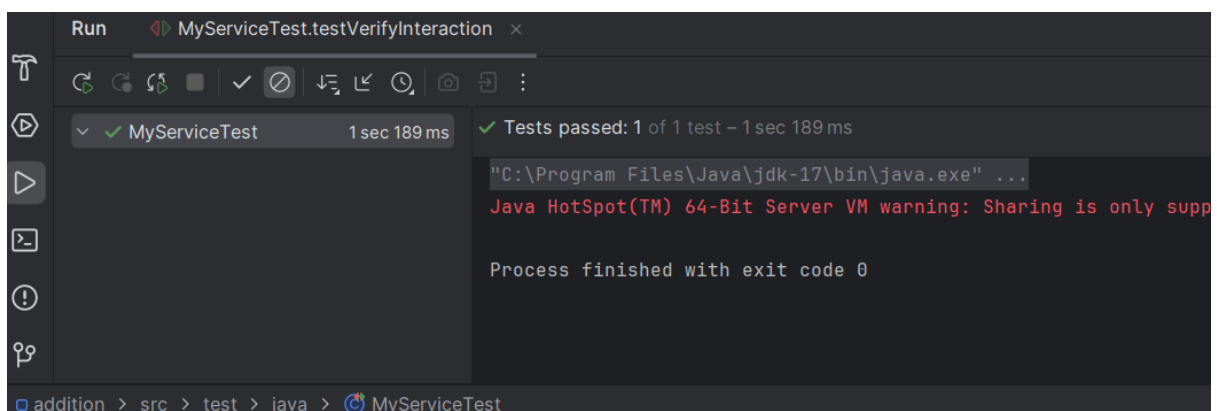
The screenshot shows the IntelliJ IDEA code editor with the file `MyServiceTest.java` open. The code is as follows:

```

1
2
3  import org.junit.jupiter.api.Test;
4  import org.mockito.Mockito;
5  import static org.mockito.Mockito.*;
6
7  public class MyServiceTest {
8
9      @Test
10     public void testVerifyInteraction() {
11         // Arrange - Create mock
12         ExternalApi mockApi = Mockito.mock(ExternalApi.class);
13
14         // Act - Use service
15         MyService service = new MyService(mockApi);
16         service.fetchData();
17
18         // Assert - Verify method call
19         verify(mockApi).getData();
20     }
21 }
22

```

OUTPUT Screen: Process finished with exit code 0



The screenshot shows the IntelliJ IDEA Run window for the test `MyServiceTest.testVerifyInteraction`. The test passed successfully. The output text is as follows:

```

Run MyServiceTest.testVerifyInteraction x
✓ MyServiceTest 1 sec 189 ms ✓ Tests passed: 1 of 1 test – 1 sec 189 ms
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
Java HotSpot(TM) 64-Bit Server VM warning: Sharing is only supp
Process finished with exit code 0

```

The breadcrumb at the bottom indicates the location: `addition > src > test > java > MyServiceTest`.