Mockito exercises

Exercise 1: Mocking and Stubbing

Goal: Use Mockito to create a mock of an external API and stub its behavior.

Step-by-step:

1. Define an interface ExternalApi:

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

2. Define a service class that depends on ExternalApi:

```
public class MyService {
 private ExternalApi api;

public MyService(ExternalApi api) {
    this.api = api;
}

public String fetchData() {
    return api.getData();
}


}
```

3. Write the Mockito test using mock and when().thenReturn():

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

public class MyServiceTest {

@Test
public void testFetchData_withMockedApi() {
    // Arrange
    ExternalApi mockApi = mock(ExternalApi.class);
    when(mockApi.getData()).thenReturn("Mock Data");

    MyService service = new MyService(mockApi);

    // Act
    String result = service.fetchData();

    // Assert
    assertEquals("Mock Data", result);
}


}
```

Exercise 2: Verifying Interactions

Goal: Use verify() to confirm that a method was called.

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

public class MyServiceInteractionTest {

@Test
public void testVerifyMethodCalled() {
    // Arrange
```

```
    ExternalApi mockApi = mock(ExternalApi.class);
    MyService service = new MyService(mockApi);

    // Act
    service.fetchData();

    // Assert
    verify(mockApi).getData(); // Confirm getData was called once
}


}
```

Exercise 3: Argument Matchers

Goal: Use Mockito argument matchers (like anyString()) to stub methods based on flexible input.

1. Modify ExternalApi with a method that takes arguments:

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

2. Modify MyService:

```
public class MyService {
 private ExternalApi api;

public MyService(ExternalApi api) {
    this.api = api;
}
```

```
public String getGreeting(String name) {
    return api.getPersonalizedData(name);
}


}
```

3. Write test using Mockito's anyString():

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

public class MyServiceMatcherTest {

@Test
public void testGetGreeting_withArgumentMatcher() {
    // Arrange
    ExternalApi mockApi = mock(ExternalApi.class);
    when(mockApi.getPersonalizedData(anyString())).thenReturn("Hello!");

    MyService service = new MyService(mockApi);

    // Act
    String result = service.getGreeting("Alice");

    // Assert
    assertEquals("Hello!", result);
}


}
```