Mockito Exercises - Solutions

# Exercise 1: Mocking and Stubbing

Test a service that depends on an external API by mocking and stubbing.

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
import static org.junit.jupiter.api.Assertions.\*;  
  
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);  
 }  
}

# Exercise 2: Verifying Interactions

Ensure a method is called with specific arguments.

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
import 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();  
 }  
}

# Exercise 3: Argument Matching

import static org.mockito.Mockito.\*;  
import static org.mockito.ArgumentMatchers.\*;  
import org.junit.jupiter.api.Test;  
  
public class ArgumentMatchingTest {  
 @Test  
 public void testArgumentMatching() {  
 ExternalApi mockApi = mock(ExternalApi.class);  
 mockApi.send("Hello");  
  
 verify(mockApi).send(anyString());  
 }  
}

# Exercise 4: Handling Void Methods

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
  
public class VoidMethodTest {  
 @Test  
 public void testVoidMethod() {  
 Logger mockLogger = mock(Logger.class);  
 doNothing().when(mockLogger).log(anyString());  
  
 mockLogger.log("Test");  
  
 verify(mockLogger).log("Test");  
 }  
}

# Exercise 5: Mocking and Stubbing with Multiple Returns

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class MultiReturnTest {  
 @Test  
 public void testMultipleReturns() {  
 ExternalApi mockApi = mock(ExternalApi.class);  
 when(mockApi.getData()).thenReturn("First", "Second", "Third");  
  
 assertEquals("First", mockApi.getData());  
 assertEquals("Second", mockApi.getData());  
 assertEquals("Third", mockApi.getData());  
 }  
}

# Exercise 6: Verifying Interaction Order

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
import org.mockito.InOrder;  
  
public class OrderVerificationTest {  
 @Test  
 public void testInteractionOrder() {  
 ExternalApi mockApi = mock(ExternalApi.class);  
  
 mockApi.start();  
 mockApi.getData();  
 mockApi.end();  
  
 InOrder inOrder = inOrder(mockApi);  
 inOrder.verify(mockApi).start();  
 inOrder.verify(mockApi).getData();  
 inOrder.verify(mockApi).end();  
 }  
}

# Exercise 7: Handling Void Methods with Exceptions

import static org.mockito.Mockito.\*;  
import org.junit.jupiter.api.Test;  
  
public class VoidExceptionTest {  
 @Test  
 public void testVoidMethodException() {  
 Logger mockLogger = mock(Logger.class);  
 doThrow(new RuntimeException("Log failed")).when(mockLogger).log("fail");  
  
 try {  
 mockLogger.log("fail");  
 } catch (RuntimeException e) {  
 System.out.println(e.getMessage());  
 }  
  
 verify(mockLogger).log("fail");  
 }  
}