Mockito Hands-On Exercises - Solution

# Exercise 1: Mocking and Stubbing

Test for mocking an external API and stubbing return values:

import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
import org.mockito.Mockito;  
  
class ExternalApi {  
 public String getData() {  
 return "Real Data";  
 }  
}  
  
class MyService {  
 private ExternalApi api;  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
 public String fetchData() {  
 return api.getData();  
 }  
}  
  
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);  
 }  
}

Output:  
Test passed: returned 'Mock Data' as expected.

# Exercise 2: Verifying Interactions

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

Output:  
Verified: getData() was called on the mock.

# Exercise 3: Argument Matching

@Test  
public void testArgumentMatching() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 when(mockApi.getDataById(anyInt())).thenReturn("Matched Data");  
 mockApi.getDataById(42);  
 verify(mockApi).getDataById(eq(42));  
}

Output:  
Argument 42 matched and verified.

# Exercise 4: Handling Void Methods

@Test  
public void testVoidMethod() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 doNothing().when(mockApi).logEvent();  
 mockApi.logEvent();  
 verify(mockApi).logEvent();  
}

Output:  
Void method call verified successfully.

# Exercise 5: Multiple Return Values

@Test  
public void testMultipleReturns() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 when(mockApi.getData()).thenReturn("First").thenReturn("Second");  
 assertEquals("First", mockApi.getData());  
 assertEquals("Second", mockApi.getData());  
}

Output:  
Test passed with sequential returns: 'First', then 'Second'.

# Exercise 6: Verifying Interaction Order

@Test  
public void testInteractionOrder() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 mockApi.first();  
 mockApi.second();  
 InOrder inOrder = inOrder(mockApi);  
 inOrder.verify(mockApi).first();  
 inOrder.verify(mockApi).second();  
}

Output:  
Interaction order verified: first() then second().

# Exercise 7: Void Methods with Exceptions

@Test  
public void testVoidMethodThrowsException() {  
 ExternalApi mockApi = Mockito.mock(ExternalApi.class);  
 doThrow(new RuntimeException("Error")).when(mockApi).logEvent();  
 assertThrows(RuntimeException.class, () -> mockApi.logEvent());  
}

Output:  
Exception thrown and test passed as expected.