Java Testing & Logging Exercises

# Exercise 1: Setting Up JUnit

Scenario: You need to set up JUnit in your Java project to start writing unit tests.  
  
Steps:  
1. Create a new Java project in your IDE (e.g., IntelliJ IDEA, Eclipse).  
2. Add JUnit dependency to your project. If you are using Maven, add the following to your pom.xml:

<dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
</dependency>

# Exercise 3: Assertions in JUnit

Scenario: You need to use different assertions in JUnit to validate your test results.  
  
Steps:  
1. Write tests using various JUnit assertions.

public class AssertionsTest {  
 @Test  
 public void testAssertions() {  
 assertEquals(5, 2 + 3);  
 assertTrue(5 > 3);  
 assertFalse(5 < 3);  
 assertNull(null);  
 assertNotNull(new Object());  
 }  
}

# Exercise 4: AAA Pattern and Test Fixtures

Scenario: Organize your tests using Arrange-Act-Assert and use setup/teardown methods.  
  
Steps:  
1. Write tests using AAA pattern.  
2. Use @Before and @After for setup and teardown.

public class SampleTest {  
 private List<String> list;  
  
 @Before  
 public void setUp() {  
 list = new ArrayList<>();  
 }  
  
 @Test  
 public void testAdd() {  
 list.add("test");  
 assertEquals(1, list.size());  
 }  
  
 @After  
 public void tearDown() {  
 list.clear();  
 }  
}

# Exercise 1: Mocking and Stubbing (Mockito)

Scenario: Test a service that depends on an external API. Use Mockito to mock it.  
  
Steps:  
1. Create a mock object.  
2. Stub methods.  
3. Write test.

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

Scenario: Ensure method is called with specific arguments.  
  
Steps:  
1. Create mock.  
2. Call method.  
3. Verify interaction.

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();  
 }  
}

# Task: Logging with SLF4J

Scenario: Demonstrate logging error and warning messages.  
  
Steps:  
1. Add SLF4J and Logback dependencies.  
2. Create a logging class.

<dependency>  
 <groupId>org.slf4j</groupId>  
 <artifactId>slf4j-api</artifactId>  
 <version>1.7.30</version>  
</dependency>  
<dependency>  
 <groupId>ch.qos.logback</groupId>  
 <artifactId>logback-classic</artifactId>  
 <version>1.2.3</version>  
</dependency>  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
public class LoggingExample {  
 private static final Logger logger = LoggerFactory.getLogger(LoggingExample.class);  
  
 public static void main(String[] args) {  
 logger.error("This is an error message");  
 logger.warn("This is a warning message");  
 }  
}