WEEK 2:- PROGRAMMING LANGUAGES

Test driven development and Logging framework ---

Junit Testing Exercises ---

Exercise 1 --- src\test\java\com\example\AppTest.java

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>
```

3. Create a new test class in your project.

Exercise 3 --- src\test\java\com\example\AssertionsTest.java

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.

Solution Code:

```java
public class AssertionsTest {
  @Test
  public void testAssertions() {
    // Assert equals
    assertEquals(5, 2 + 3);

    // Assert true
    assertTrue(5 > 3);

    // Assert false
    assertFalse(5 < 3);

    // Assert null
    assertNull(null);

    // Assert not null
    assertNotNull(new Object());
  }
}
```

Exercise 4 --- src\test\java\com\example\LifecycleTest.java

Exercise 4: Arrange-Act-Assert (AAA) Pattern, Test Fixtures, Setup and Teardown Methods in JUnit

Scenario:

You need to organize your tests using the Arrange-Act-Assert (AAA) pattern and use setup and teardown methods.

Steps:

1. Write tests using the AAA pattern.

2. Use @Before and @After annotations for setup and teardown methods.

Mockito Exercises ---

Exercise 1 --- src\test\java\com\example\MyServiceTest.java

Exercise 1: Mocking and Stubbing

Scenario:
You need to test a service that depends on an external API. Use Mockito to mock the external API and stub its methods.

Steps:
1. Create a mock object for the external API.
2. Stub the methods to return predefined values.
3. Write a test case that uses the mock object.

Solution Code:
```
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 --- src\test\java\com\example\MyServiceTest2.java

Exercise 2: Verifying Interactions

Scenario:

You need to ensure that a method is called with specific arguments.

Steps:

1. Create a mock object.
2. Call the method with specific arguments.
3. Verify the interaction.

Solution Code:

```
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();
  }
}
```

SLF4J Logging Exercises ---

Exercise 1 --- src\main\java\com\example\ LoggingExample.java

Exercise 1: Logging Error Messages and Warning Levels

Task: Write a Java application that demonstrates logging error messages and warning levels using SLF4J.

Step-by-Step Solution:

1. Add SLF4J and Logback dependencies to your `pom.xml` file:

```
<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>
```

2. Create a Java class that uses SLF4J for logging:

```java
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");
   }
}
```