**TDD using JUnit5 and Mockito**

**JUnit\_Basic Testing 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: junit junit 4.13.2 test**

**3. Create a new test class in your project.**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>junit\_mandatory\_basic\_1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 4 dependency -->  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Hamcrest for assertions -->  
 <dependency>  
 <groupId>org.hamcrest</groupId>  
 <artifactId>hamcrest</artifactId>  
 <version>2.2</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>

</project>

**NumberUtilsjava:**

package org.example;  
public class NumberUtils{  
 public boolean isPalindrome(int number){  
 if (number<0) return false;  
 int original=number;  
 int reversed=0;  
 while(number!=0) {  
 int digit=number%10;  
 reversed=reversed\*10+digit;  
 number/=10;  
 }  
 return original==reversed;  
 }  
}

**NumberUtilsTest**

package org.example;  
import org.junit.Test;  
import static org.junit.Assert.\*;  
public class NumberUtilsTest{  
 @Test  
 public void testIsPalindrome\_withPalindromeNumbers(){  
 NumberUtils utils=new NumberUtils();  
 *assertTrue*(utils.isPalindrome(121));  
 *assertTrue*(utils.isPalindrome(1221));  
 *assertTrue*(utils.isPalindrome(1));  
 *assertTrue*(utils.isPalindrome(0));  
 }  
 @Test  
 public void testIsPalindrome\_withNonPalindromeNumbers(){  
 NumberUtils utils=new NumberUtils();  
 *assertFalse*(utils.isPalindrome(123));  
 *assertFalse*(utils.isPalindrome(10));  
 *assertFalse*(utils.isPalindrome(-121));  
 }  
}

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**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:**

**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());**

**}**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>junit\_mandatory\_basic\_1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit Jupiter API -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <version>5.9.3</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- JUnit Jupiter Engine (required to run tests) -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.9.3</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.22.2</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**AssertionsTest.java:**

package org.example;  
import static org.junit.jupiter.api.Assertions.\*;  
import org.junit.jupiter.api.Test;  
  
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());  
 }  
}

**OUTPUT:**

A screenshot of a computer program

AI-generated content may be incorrect.

**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.**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>junit\_mandatory\_basic\_1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit Jupiter API -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <version>5.9.3</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- JUnit Jupiter Engine (required to run tests) -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>5.9.3</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.22.2</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**PalindromeChecker.java:**

package org.example;  
public class PalindromeChecker {  
 public boolean isPalindrome(int number){  
 int original=number;  
 int reversed=0;  
 while (number>0){  
 int digit=number%10;  
 reversed=reversed\*10+digit;  
 number=number/10;  
 }  
 return original==reversed;  
 }  
 public void reset(){  
 // No-op: For demo teardown purposes  
 }  
}

**PalindromeCheckerTest.java:**

package org.example;  
import org.junit.jupiter.api.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class PalindromeCheckerTest{  
 private PalindromeChecker checker;  
 @BeforeEach  
 public void setUp(){  
 checker=new PalindromeChecker(); // Arrange  
 System.*out*.println("Setup done");  
 }  
 @AfterEach  
 public void tearDown(){  
 checker.reset(); // Simulate cleanup  
 System.*out*.println("Teardown done");  
 }  
 @Test  
 public void testPalindromeNumber(){  
 // Act  
 boolean result = checker.isPalindrome(121);  
 // Assert  
 *assertTrue*(result,"121 should be a palindrome");  
 }  
 @Test  
 public void testNonPalindromeNumber(){  
 // Act  
 boolean result=checker.isPalindrome(123);  
 // Assert  
 *assertFalse*(result,"123 is not a palindrome");  
 }  
 @Test  
 public void testSingleDigitNumber(){  
 // Act  
 boolean result = checker.isPalindrome(7);  
 // Assert  
 *assertTrue*(result, "Single-digit numbers are palindromes");  
 }  
}

**OUTPUT:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Mockito exercises**

**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.**

**Pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>junit\_mandatory\_basic\_1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 5 -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.9.3</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Mockito -->  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>5.11.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.22.2</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**ExternalApi.java:**

package org.example;  
  
public interface ExternalApi {  
 String getData();  
}

**MyService.java:**

package org.example;  
  
public class MyService {  
 private ExternalApi api;  
  
 public MyService(ExternalApi api) {  
 this.api = api;  
 }  
  
 public String fetchData() {  
 return api.getData();  
 }  
}

**MyServiceTest.java:**

package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
  
public class MyServiceTest {  
  
 @Test  
 public void testExternalApi() {  
 // Step 1: Create mock  
 ExternalApi mockApi = *mock*(ExternalApi.class);  
  
 // Step 2: Stub the method  
 *when*(mockApi.getData()).thenReturn("Mock Data");  
  
 // Step 3: Inject mock into service and test  
 MyService service = new MyService(mockApi);  
 String result = service.fetchData();  
  
 // Step 4: Assert the result  
 *assertEquals*("Mock Data", result);  
 }  
}

**OUTPUT:**

A screenshot of a computer program

AI-generated content may be incorrect.

**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.**

**Pom.XML:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>junit\_mandatory\_basic\_1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- JUnit 5 -->  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter</artifactId>  
 <version>5.9.3</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- Mockito -->  
 <dependency>  
 <groupId>org.mockito</groupId>  
 <artifactId>mockito-core</artifactId>  
 <version>5.11.0</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-surefire-plugin</artifactId>  
 <version>2.22.2</version>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**ExternalApi.java:**

package org.example;  
  
public interface ExternalApi{  
 String getData();  
}

**MyService.java:**

package org.example;  
public class MyService{  
 private ExternalApi api;  
 public MyService(ExternalApi api){  
 this.api = api;  
 }  
 public String fetchData(){  
 return api.getData(); // delegate to external API  
 }  
}

**MyServiceTest.java:**

package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
public class MyServiceTest{  
 @Test  
 public void testVerifyInteraction(){  
 // Step 1: Create mock  
 ExternalApi mockApi=*mock*(ExternalApi.class);  
 // Step 2: Inject into service and call method  
 MyService service=new MyService(mockApi);  
 service.fetchData(); // should call mockApi.getData()  
 // Step 3: Verify interaction  
 *verify*(mockApi).getData(); // verify that getData() was called  
 }  
}

**OUTPUT:**

A screenshot of a computer program

AI-generated content may be incorrect.