**Exercise 1: Parameterized Tests**

**EvenChecker.java**

public class EvenChecker {

public boolean isEven(int number) {

return number % 2 == 0;

}

}

**EvenCheckerTest.java**

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;

import static org.junit.jupiter.api.Assertions.\*;

public class EvenCheckerTest {

EvenChecker checker = new EvenChecker();

@ParameterizedTest

@ValueSource(ints = {2, 4, 6, 8, 10})

void testEvenNumbers(int number) {

assertTrue(checker.isEven(number), number + " should be even");

}

@ParameterizedTest

@ValueSource(ints = {1, 3, 5, 7, 9})

void testOddNumbers(int number) {

assertFalse(checker.isEven(number), number + " should be odd");

}

}

**Exercise 2: Test Suites and Categories**

**AllTests.java**

import org.junit.platform.suite.api.SelectClasses;

import org.junit.platform.suite.api.Suite;

@Suite

@SelectClasses({

EvenCheckerTest.class,

OrderedTests.class,

ExceptionThrowerTest.class,

PerformanceTesterTest.class

})

public class AllTests {

// This class is a suite. No implementation needed.

}

**Exercise 3: Test Execution Order**

**OrderedTests.java**

import org.junit.jupiter.api.\*;

import static org.junit.jupiter.api.Assertions.\*;

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)

public class OrderedTests {

@Test

@Order(1)

void testA() {

System.out.println("Test A executed");

assertTrue(true);

}

@Test

@Order(2)

void testB() {

System.out.println("Test B executed");

assertTrue(true);

}

@Test

@Order(3)

void testC() {

System.out.println("Test C executed");

assertTrue(true);

}

}

**Exercise 4: Exception Testing**

**ExceptionThrower.java**

public class ExceptionThrower {

public void throwException(boolean shouldThrow) {

if (shouldThrow) {

throw new IllegalArgumentException("Exception was thrown");

}

}

}

**ExceptionThrowerTest.java**

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.\*;

public class ExceptionThrowerTest {

ExceptionThrower thrower = new ExceptionThrower();

@Test

void testExceptionThrown() {

assertThrows(IllegalArgumentException.class, () -> thrower.throwException(true));

}

@Test

void testNoExceptionThrown() {

assertDoesNotThrow(() -> thrower.throwException(false));

}

}

**Exercise 5: Timeout and Performance Testing**

**PerformanceTester.java**

public class PerformanceTester {

public void performTask() {

try {

Thread.sleep(100); // 100 milliseconds

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

**PerformanceTesterTest.java**

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertTimeout;

import java.time.Duration;

public class PerformanceTesterTest {

PerformanceTester tester = new PerformanceTester();

@Test

void testPerformTaskCompletesInTime() {

assertTimeout(Duration.ofMillis(200), () -> tester.performTask());

}

}