**Exercise 1: Parameterized Tests**

**Java Class:**

public class EvenChecker {

public boolean isEven(int number) {

return number % 2 == 0;

}

}

**JUnit Test with Parameterized Inputs:**

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

import org.junit.jupiter.params.ParameterizedTest;

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

public class EvenCheckerTest {

@ParameterizedTest

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

void testIsEven(int number) {

EvenChecker checker = new EvenChecker();

assertTrue(checker.isEven(number));

}

}

**Exercise 2: Test Suites and Categories**

**JUnit 5 Test Suite:**

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

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

@Suite

@SelectClasses({EvenCheckerTest.class, ExceptionThrowerTest.class})

public class AllTests {

}

**Exercise 3: Test Execution Order**

**Controlling Test Order:**

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

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)

public class OrderedTests {

@Test

@Order(1)

void testFirst() {

System.out.println("Test 1");

}

@Test

@Order(2)

void testSecond() {

System.out.println("Test 2");

}

}

**Exercise 4: Exception Testing**

**Java Class:**

public class ExceptionThrower {

public void throwException() throws IllegalArgumentException {

throw new IllegalArgumentException("Invalid argument");

}

}

**JUnit Test:**

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

import org.junit.jupiter.api.Test;

public class ExceptionThrowerTest {

@Test

void testException() {

ExceptionThrower et = new ExceptionThrower();

assertThrows(IllegalArgumentException.class, et::throwException);

}

}

**Exercise 5: Timeout and Performance Testing**

**Java Class:**

public class PerformanceTester {

public void performTask() throws InterruptedException {

Thread.sleep(100); // simulate time-consuming task

}

}

**JUnit Test with Timeout:**

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

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.Timeout;

import java.util.concurrent.TimeUnit;

public class PerformanceTesterTest {

@Test

@Timeout(value = 500, unit = TimeUnit.MILLISECONDS)

void testPerformTaskWithinTime() throws InterruptedException {

PerformanceTester tester = new PerformanceTester();

tester.performTask();

}

}