**Week\_2\_JUnit\_Adavanced\_Testing**

**Exercise 1: Parameterized Tests:**

**EvenChecker.java:**

public class EvenChecker {

public boolean isEven(int num) {

return num % 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 {

private final EvenChecker checker = new EvenChecker();

@ParameterizedTest

@ValueSource(ints = {2, 4, 6, 8, 10, 12})

void testIsEven\_WithEvenNumbers(int num) {

assertTrue(checker.isEven(num));

}

@ParameterizedTest

@ValueSource(ints = {1, 3, 5, 7, 9, 11})

void testIsEven\_WithOddNumbers(int num) {

assertFalse(checker.isEven(num));

}

}

**Output:**

**A screenshot of a computer error message

AI-generated content may be incorrect.**

**Exercise 2: Test Suites and Categories:**

**AdditionTest.java:**

a import org.junit.jupiter.api.Test;

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

public class AdditionTest {

@Test

void testAddPositive() {

assertEquals(8, 4 + 4);

}

@Test

void testAddNegative() {

assertEquals(-8, -4 + -4);

}

}

**MultiplicationTest.java:**

a import org.junit.jupiter.api.Test;

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

public class MultiplicationTest {

@Test

void testMultiplyPositive() {

assertEquals(16, 4 \* 4);

}

@Test

void testMultiplyByZero() {

assertEquals(0, 8 \* 0);

}

}

**AllTests.java:**

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

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

public class AllTests {

}

@Suite

@SelectClasses({

AdditionTest.class,

MultiplicationTest.class

})

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 3: Test Execution Order:**

**OrderedTests.java:**

import org.junit.jupiter.api.MethodOrderer.OrderAnnotation;

import org.junit.jupiter.api.Order;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.TestMethodOrder;

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

@TestMethodOrder(OrderAnnotation.class)

public class OrderedTests {

@Test

@Order(3)

void testC() {

System.out.println("Running testC()");

assertTrue(3 < 4);

}

@Test

@Order(1)

void testA() {

System.out.println("Running testA()");

assertEquals(2 + 2, 4);

}

@Test

@Order(2)

void testB() {

System.out.println("Running testB()");

assertFalse(5 < 3);

}

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 4: Exception Testing:**

**ExceptionThrower.java:**

public class ExceptionThrower {

public void throwException() {

throw new IllegalArgumentException("This is an expected exception.");

}

}

**ExceptionThrowerTest.java:**

import org.junit.jupiter.api.Test;

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

public class ExceptionThrowerTest {

@Test

void testThrowException() {

ExceptionThrower exThrower = new ExceptionThrower();

IllegalArgumentException exception = assertThrows(

IllegalArgumentException.class,

() -> exThrower.throwException()

);

assertEquals("This is an expected exception.", exception.getMessage());

}

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Exercise 5: Timeout and Performance Testing:**

**PerformanceTester.java:**

public class PerformanceTester {

public void performTask() {

try {

Thread.sleep(300);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

**PerformanceTesterTest.java:**

import org.junit.jupiter.api.Test;

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

import java.time.Duration;

public class PerformanceTesterTest {

@Test

void testPerformTaskCompletesWithinTimeout() {

PerformanceTester perfTester = new PerformanceTester();

assertTimeout(

Duration.ofMillis(500),

perfTester::performTask

);

}

}

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.