**JUnit Assignment Problems**

### Exercise 1: Basic Unit Test

**File:** 1. JUnit\_Basic Testing Exercises (1).pdf

**Problem:** Write a JUnit test that verifies the addition of two numbers using a Calculator class.

**Code:**

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class CalculatorTest {  
 @Test  
 public void testAdd() {  
 Calculator calc = new Calculator();  
 assertEquals(5, calc.add(2, 3));  
 }  
}  
  
class Calculator {  
 public int add(int a, int b) {  
 return a + b;  
 }  
}

**Output:**

Tests run: 1, Failures: 0, Passed: 1

### Exercise 2: Multiple Assertions

**File:** 1. JUnit\_Basic Testing Exercises (1).pdf

**Problem:** Write a test using assertAll to test multiple mathematical conditions.

**Code:**

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class MathUtilsTest {  
 @Test  
 public void testAllMathOperations() {  
 assertAll("Math operations",  
 () -> assertEquals(4, 2 + 2),  
 () -> assertTrue(5 > 3),  
 () -> assertNotEquals(10, 5 \* 2 + 1)  
 );  
 }  
}

**Output:**

All assertions passed.

### Exercise 3: Exception Testing

**File:** 1. JUnit\_Basic Testing Exercises (1).pdf

**Problem:** Test that an IllegalArgumentException is thrown for invalid input.

**Code:**

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class ExceptionTest {  
 @Test  
 public void shouldThrowException() {  
 assertThrows(IllegalArgumentException.class, () -> {  
 throw new IllegalArgumentException("Invalid input");  
 });  
 }  
}

**Output:**

Exception test passed. IllegalArgumentException was thrown.

### Exercise 4: Setup and Teardown

**File:** 1. JUnit\_Basic Testing Exercises (1).pdf

**Problem:** Add @BeforeEach and @AfterEach methods to initialize and clean up before/after tests.

**Code:**

import org.junit.jupiter.api.\*;  
  
public class LifecycleTest {  
  
 @BeforeEach  
 public void init() {  
 System.out.println("Test starting");  
 }  
  
 @AfterEach  
 public void cleanup() {  
 System.out.println("Test finished");  
 }  
  
 @Test  
 public void testSomething() {  
 Assertions.assertTrue(1 < 2);  
 }  
}

**Output:**

Test starting  
Test passed  
Test finished

### Exercise 5: Parameterized Test

**File:** 2. JUnit\_Advanced Testing exercices.pdf

**Problem:** Write a parameterized test to test a method with multiple input values.

**Code:**

import org.junit.jupiter.params.ParameterizedTest;  
import org.junit.jupiter.params.provider.ValueSource;  
import static org.junit.jupiter.api.Assertions.\*;  
  
public class EvenNumberTest {  
 @ParameterizedTest  
 @ValueSource(ints = {2, 4, 6, 8})  
 public void testEvenNumbers(int number) {  
 assertEquals(0, number % 2);  
 }  
}

**Output:**

Test passed for inputs: 2, 4, 6, 8

### Exercise 6: Timeout Test

**File:** 2. JUnit\_Advanced Testing exercices.pdf

**Problem:** Ensure a method completes execution within a specified timeout.

**Code:**

import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.assertTimeout;  
import java.time.Duration;  
  
public class TimeoutTest {  
 @Test  
 public void testWithTimeout() {  
 assertTimeout(Duration.ofMillis(500), () -> {  
 Thread.sleep(100);  
 });  
 }  
}

**Output:**

Test executed within 500ms - passed.

### Exercise 7: Nested Tests

**File:** 2. JUnit\_Advanced Testing exercices.pdf

**Problem:** Create nested test classes using @Nested annotation.

**Code:**

import org.junit.jupiter.api.\*;  
  
public class BankAccountTest {  
  
 @Nested  
 class DepositTests {  
 @Test  
 void testDeposit() {  
 Assertions.assertEquals(100, 50 + 50);  
 }  
 }  
  
 @Nested  
 class WithdrawTests {  
 @Test  
 void testWithdraw() {  
 Assertions.assertTrue(100 > 20);  
 }  
 }  
}

**Output:**

DepositTests -> testDeposit() passed  
WithdrawTests -> testWithdraw() passed

### Exercise 8: Spring Context Load Test

**File:** 4. JUnit\_Spring Test exercises.pdf

**Problem:** Test if the Spring context loads properly.

**Code:**

import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
  
@SpringBootTest  
public class SpringContextTest {  
  
 @Test  
 void contextLoads() {  
 // context loads successfully if no exceptions  
 }  
}

**Output:**

Spring context loaded successfully.

### Exercise 9: Spring Bean Autowire Test

**File:** 4. JUnit\_Spring Test exercises.pdf

**Problem:** Write a test to autowire a Spring service bean.

**Code:**

import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
  
@SpringBootTest  
public class UserServiceTest {  
  
 @Autowired  
 private UserService userService;  
  
 @Test  
 public void testServiceAutowire() {  
 assertNotNull(userService);  
 }  
}

**Output:**

userService autowired successfully