**Day2 Lab2: Boundary Value Analysis** and **Edge Case Testing** with **JUnit 5**

**Exercise 1: Bank Withdrawal (Boundary + Edge Cases)**

System rules:

* Balance must be **≥ 100** after withdrawal.
* Cannot withdraw **negative** amount.

import org.junit.jupiter.api.Test;

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

public class BankTest {

boolean canWithdraw(int balance, int amount) {

if (amount <= 0) return false;

return (balance - amount) >= 100;

}

@Test

void testNegativeAmount() {

assertFalse(canWithdraw(1000, -50), "Negative withdrawal should be invalid");

}

@Test

void testZeroAmount() {

assertFalse(canWithdraw(1000, 0), "Zero withdrawal should be invalid");

}

@Test

void testBoundaryWithdrawal() {

assertTrue(canWithdraw(500, 400), "Should allow leaving exactly 100");

assertFalse(canWithdraw(500, 401), "Should not allow leaving <100");

}

@Test

void testEdgeBalance() {

assertFalse(canWithdraw(100, 1), "Cannot withdraw if balance is only 100");

}

}

**Tested negative amount, zero, exact boundary, just below boundary.**

**Exercise 2: with @ParameterizedTest (Boundary Values)**

import org.junit.jupiter.params.ParameterizedTest;

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

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

public class TemperatureTest {

boolean isNormalTemp(int temp) {

return temp >= 36 && temp <= 37;

}

@ParameterizedTest

@ValueSource(ints = {35, 36, 37, 38})

void testTemperatureBoundaries(int temp) {

if (temp == 36 || temp == 37) {

assertTrue(isNormalTemp(temp), temp + " should be normal");

} else {

assertFalse(isNormalTemp(temp), temp + " should be abnormal");

}

}

}

**Tests just below, exact limits, and just above body temperature range.**

So in summary:

* **Boundary values:** min, min-1, max, max+1.
* **Edge cases:** null, empty, negative, zero, large input, special chars.