**Day2 Lab2: Boundary value and Edge case testing – Loan EMI Calculator**

A **Banking Loan EMI Calculator** example with **boundary value & edge case testing** in xUnit.

**Scenario: EMI Calculator**

A bank has these rules for loan calculation:

1. **Principal Amount** must be between **₹50,000 and ₹10,00,000**.
2. **Interest Rate** must be between **5% and 20%**.
3. **Tenure (Months)** must be between **12 and 360 months**.

**Class to Test**

using System;

public class LoanEmiCalculator

{

public decimal CalculateEmi(decimal principal, decimal annualInterestRate, int tenureMonths)

{

if (principal < 50000 || principal > 1000000)

throw new ArgumentOutOfRangeException(nameof(principal), "Principal out of range");

if (annualInterestRate < 5 || annualInterestRate > 20)

throw new ArgumentOutOfRangeException(nameof(annualInterestRate), "Interest rate out of range");

if (tenureMonths < 12 || tenureMonths > 360)

throw new ArgumentOutOfRangeException(nameof(tenureMonths), "Tenure out of range");

decimal monthlyRate = (annualInterestRate / 12) / 100;

double pow = Math.Pow(1 + (double)monthlyRate, tenureMonths);

decimal emi = principal \* monthlyRate \* (decimal)pow / ((decimal)pow - 1);

return Math.Round(emi, 2);

}

}

**Exercise 1: Boundary Value Tests (xUnit)**

using Xunit;

public class LoanEmiCalculator\_BoundaryTests

{

private readonly LoanEmiCalculator \_calculator = new LoanEmiCalculator();

[Theory]

[InlineData(50000, 5, 12)] // Lower boundary principal, rate, tenure

[InlineData(1000000, 20, 360)] // Upper boundary principal, rate, tenure

[InlineData(50001, 5.1, 13)] // Just above lower boundary

[InlineData(999999, 19.9, 359)]// Just below upper boundary

public void BoundaryValues\_ShouldNotThrow(decimal principal, decimal rate, int months)

{

var emi = \_calculator.CalculateEmi(principal, rate, months);

Assert.True(emi > 0); // EMI should always be positive

}

[Theory]

[InlineData(49999, 10, 120)] // Below principal limit

[InlineData(1000001, 10, 120)] // Above principal limit

[InlineData(200000, 4.9, 120)] // Below interest rate limit

[InlineData(200000, 20.1, 120)]// Above interest rate limit

[InlineData(200000, 10, 11)] // Below tenure limit

[InlineData(200000, 10, 361)] // Above tenure limit

public void OutOfBoundary\_ShouldThrow(decimal principal, decimal rate, int months)

{

Assert.Throws<ArgumentOutOfRangeException>(() => \_calculator.CalculateEmi(principal, rate, months));

}

}

**Exercise 2: Edge Case Tests (xUnit)**

using Xunit;

public class LoanEmiCalculator\_EdgeCaseTests

{

private readonly LoanEmiCalculator \_calculator = new LoanEmiCalculator();

[Fact]

public void VeryHighPrincipalAndLowRate\_ShouldReturnReasonableEmi()

{

var emi = \_calculator.CalculateEmi(1000000, 5, 360);

Assert.True(emi > 0 && emi < 100000); // EMI shouldn’t exceed absurd limits

}

[Fact]

public void VeryLowPrincipalAndHighRate\_ShouldReturnHighEmi()

{

var emi = \_calculator.CalculateEmi(50000, 20, 12);

Assert.True(emi > 4000); // EMI should be higher than principal/tenure

}

[Fact]

public void PrincipalAtMidRangeWithAverageRateAndTenure\_ShouldWork()

{

var emi = \_calculator.CalculateEmi(500000, 10, 120);

Assert.InRange(emi, 6000, 7000); // Check approximate expected EMI range

}

[Fact]

public void MinimumTenure\_ShouldReturnAlmostPrincipal()

{

var emi = \_calculator.CalculateEmi(100000, 5, 12);

Assert.True(emi > 8000); // Close to principal/12

}

}

**Explanation**

* **Boundary Testing**: Tested exact limits (50k, 10L, 5%, 20%, 12, 360). Also just inside & outside boundaries.
* **Edge Cases**:
  + Very high loan with low interest.
  + Very small loan with max interest.
  + Middle-of-the-range realistic case.
  + One-year minimum tenure.