**Day1: Lab - A Loan Eligibility Calculator.**

Using [Fact], [Theory] (with [InlineData], and [MemberData]) to test it.

**Step 1: Business Logic – Loan Eligibility Calculator**

Let’s say the rules are:

1. Customer is **eligible** if:
   * Net income ≥ 25,000
   * No more than 2 existing loans
   * Age between 21 and 60
2. Otherwise → Not eligible.

namespace XUnitDemo

{

public class LoanEligibility

{

public bool IsEligible(int age, decimal netIncome, int existingLoans)

{

if (age < 21 || age > 60) return false;

if (netIncome < 25000) return false;

if (existingLoans > 2) return false;

return true;

}

}

}

**Step 2: [Fact] – Single Fixed Case**

We test one **specific case**.

using Xunit;

namespace XUnitDemo.Tests

{

public class LoanEligibilityFactTests

{

private readonly LoanEligibility \_loanEligibility = new LoanEligibility();

[Fact]

public void IsEligible\_ShouldReturnTrue\_ForValidCustomer()

{

// Arrange

int age = 30;

decimal income = 50000;

int loans = 1;

// Act

bool result = \_loanEligibility.IsEligible(age, income, loans);

// Assert

Assert.True(result);

}

}

}

[Fact] = one fixed scenario.

**Step 3: [Theory] + [InlineData] – Multiple Inputs**

Now, let’s test multiple customers.

using Xunit;

namespace XUnitDemo.Tests

{

public class LoanEligibilityTheoryInlineTests

{

private readonly LoanEligibility \_loanEligibility = new LoanEligibility();

[Theory]

[InlineData(30, 40000, 1, true)] // eligible

[InlineData(20, 40000, 0, false)] // age too low

[InlineData(65, 40000, 0, false)] // age too high

[InlineData(30, 20000, 1, false)] // income too low

[InlineData(30, 40000, 3, false)] // too many loans

public void IsEligible\_ShouldReturnExpectedResult(int age, decimal income, int loans, bool expected)

{

// Act

bool result = \_loanEligibility.IsEligible(age, income, loans);

// Assert

Assert.Equal(expected, result);

}

}

}

[Theory] + InlineData = parameterized test cases.

**Step 4: [MemberData] – Reusable Data**

Now let’s store test cases in a property.

using Xunit;

using System.Collections.Generic;

namespace XUnitDemo.Tests

{

public class LoanEligibilityTheoryMemberTests

{

private readonly LoanEligibility \_loanEligibility = new LoanEligibility();

public static IEnumerable<object[]> EligibilityTestData =>

new List<object[]>

{

new object[] { 25, 30000, 0, true },

new object[] { 22, 28000, 2, true },

new object[] { 19, 50000, 0, false }, // age too low

new object[] { 40, 24000, 0, false }, // income too low

new object[] { 50, 60000, 3, false } // too many loans

};

[Theory]

[MemberData(nameof(EligibilityTestData))]

public void IsEligible\_ShouldReturnExpectedResult\_UsingMemberData(int age, decimal income, int loans, bool expected)

{

bool result = \_loanEligibility.IsEligible(age, income, loans);

Assert.Equal(expected, result);

}

}

}

[MemberData] = reusable test dataset inside the same class.