**Day2: Lab-Testing private methods in JUnit**

1. **Testing via public method (recommended)**
2. **Testing private method using Reflection (workaround)**

**Loan Eligibility Check**

**Step 1: Original Class (Before Refactoring)**

Here the **private method** does important logic → but it’s buried inside the main class.

public class LoanService {

public boolean approveLoan(String customerId, double income, double existingDebt) {

// private method does the real work

return isEligible(income, existingDebt);

}

// Private method with business logic

private boolean isEligible(double income, double existingDebt) {

double debtToIncomeRatio = existingDebt / income;

return debtToIncomeRatio < 0.4 && income >= 25000;

}

}

Here isEligible() is **private**, so hard to test directly.

**Step 2: Refactor – Extract into Helper Class**

We move the private logic into a **LoanEligibilityUtils** class.

public class LoanEligibilityUtils {

public static boolean isEligible(double income, double existingDebt) {

double debtToIncomeRatio = existingDebt / income;

return debtToIncomeRatio < 0.4 && income >= 25000;

}

}

Now LoanService just calls it:

public class LoanService {

public boolean approveLoan(String customerId, double income, double existingDebt) {

return LoanEligibilityUtils.isEligible(income, existingDebt);

}

}

**Step 3: JUnit Tests**

**A) Test the Utility Class Directly**

import org.junit.jupiter.api.Test;

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

class LoanEligibilityUtilsTest {

@Test

void testEligibleCustomer() {

assertTrue(LoanEligibilityUtils.isEligible(50000, 10000)); // ratio = 0.2, income ok

}

@Test

void testIneligibleHighDebt() {

assertFalse(LoanEligibilityUtils.isEligible(40000, 20000)); // ratio = 0.5

}

@Test

void testIneligibleLowIncome() {

assertFalse(LoanEligibilityUtils.isEligible(20000, 5000)); // income < 25k

}

}

**B) Test LoanService**

import org.junit.jupiter.api.Test;

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

class LoanServiceTest {

@Test

void testApproveLoan() {

LoanService service = new LoanService();

assertTrue(service.approveLoan("CUST001", 60000, 10000)); // eligible

assertFalse(service.approveLoan("CUST002", 20000, 2000)); // income too low

}

}

**Why This Approach is Better**

* **Business logic (isEligible)** is now reusable across the system.
* Both **LoanEligibilityUtils** and **LoanService** can be tested independently.
* No need for **reflection hacks**.

**Final Takeaway**:

* If a private method does **simple one-liners**, don’t test it directly (test via public methods).
* If it does **important business logic**, **extract it** into a helper/service/utility class and test it separately.