**Single Responsibility Principle (SRP)**

Create a new console App

Add class 🡺Invoice

Add below functionalities

sendEmail is not exact functionality of the Invoice Class 🡺 Move that functionality to another class (MailSender)

Invoice.cs

internal class Invoice

{

public long InvoiceAmount { get; set; }

public DateTime InvoiceDate { get; set; }

public void AddInvoice()

{

Console.WriteLine("Invoice Added");

MailSender mailSender = new MailSender();

mailSender.SendEmail();

// MailMessage mailMessage = new MailMessage("EmailFrom", "EmailTo", "Subject", "Invoice Added");

}

public void RemoveInvoice()

{

Console.WriteLine("Invoice Removed");

}

//public void SendInvoiceMail(MailMessage mailMessage)

//{

// Console.WriteLine("Email sent");

// }

MailSender.cs

internal class MailSender

{

public string EmailFrom { get; set; }

public int EmailTo { get; set; }

public string Subject { get; set; }

public string Body { get; set; }

public void SendEmail() {

Console.WriteLine("Mail sending ");

}

}

Another Example

**🟢 1. Single Responsibility Principle (SRP)**

**Definition**: A class should have only one reason to change.

**❌ Bad Example**

public class BankAccount

{

public void Deposit(decimal amount) { }

public void Withdraw(decimal amount) { }

public void PrintStatement() { } // Printing is a different responsibility

}

**✅ Refactored**

public class BankAccount

{

public void Deposit(decimal amount) { }

public void Withdraw(decimal amount) { }

}

public class StatementPrinter

{

public void Print(BankAccount account) { }

}

Example for Open Close Principle

Create a new Project 🡺Solid\_Open\_Close\_Demo

{

public enum InvoiceType

{

ProposedInvoice,

FinalInvoice

}

public class Invoice

{

public double GetInvoiceDiscounts(double amount,InvoiceType invoiceType )

{

double finalAmount = 0;

if(invoiceType== InvoiceType.FinalInvoice )

{

finalAmount = amount - 200;

}

if(invoiceType == InvoiceType.ProposedInvoice )

{

finalAmount= amount - 100;

}

return finalAmount;

}

}

}

In the above code if I need to modify the logic .It should create a problem wherever I implement the functionality I need change at all places

We are Going to modify like below

namespace Solid\_OPen\_Close\_Demo

{

public enum InvoiceType

{

ProposedInvoice,

FinalInvoice

}

public class Invoice

{

public virtual double GetInvoiceDiscount(double amount)

{

return amount - 100;

}

}

public class FinalInvoice : Invoice

{

public override double GetInvoiceDiscount(double amount)

{

return base.GetInvoiceDiscount(amount) - 200;

}

}

public class ProposedInvoice : Invoice

{

public override double GetInvoiceDiscount(double amount)

{

return base.GetInvoiceDiscount(amount)-50;

}

}

public class RecurringInvoice : Invoice

{

public override double GetInvoiceDiscount(double amount)

{

return base.GetInvoiceDiscount(amount)-300;

}

}

}