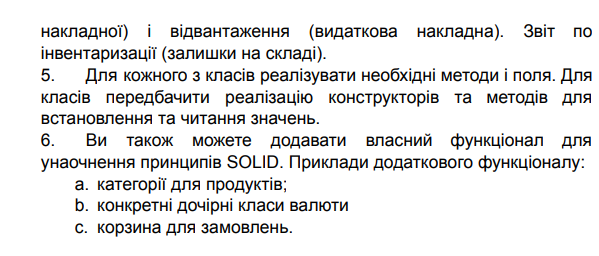
**Лабораторна робота № 2**

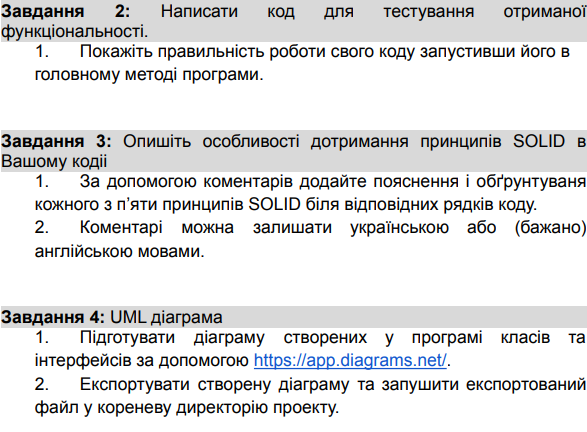
**ПРИНЦИПИ ПРОГРАМУВАННЯ. SOLID**

**Мета:** навчитися дотримуватися принципів SOLID та обґрунтовувати їх. Навчитися описувати дизайн програми за допомогою UML діаграм.

**Хід роботи:**

****

****



**Завдання 1:** Виконати завдання з дотриманням принципів SOLID

1. Програмування класу **Money**

# Лістинг коду класу Mоney:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Text.Json.Serialization;

using System.Threading.Tasks;

using WareHouseApp.Interfaces;

namespace WareHouseApp.Classes

{

internal abstract class Money

{

protected int Entire;

protected int Coin;

public bool SetMoney(int entire, int coin)

{

if(entire>=0&&coin>=0&&coin<=99)

{

Entire = entire;

Coin = coin;

Console.WriteLine("Successfully");

return true;

}

Console.WriteLine("Fail, check you input!");

return false;

}

public abstract char Sign { get; }

public abstract string ShortName { get; }

public string PrintMoney()

{

return $"{Entire}.{Coin} {Sign}";

}

}

}

//Принцип SOLID: Single Responsibility Principle (SRP)

//Клас Money відповідає за представлення грошей і маніпулювання ними.

//Має окремі поля для зберігання цілої та дробової частини грошей.

1. Створення класу **Product**

# Лістинг коду класу Product:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Serialization;

using System.Text;

using System.Threading.Tasks;

namespace WareHouseApp.Classes

{

internal class Product

{

public int Id { get; set; }

public string? Name { get; set; }

public Category? Category { get; set; }

public Money? Price { get; set; }

public string? Unit { get; set; }

public DateTime? Arrivas { get; set; }

}

}

//Принцип SOLID: Open-Closed Principle (OCP)

//Клас Product відкритий для розширення (може мати нові похідні класи), але закритий для модифікації.

//Надає метод зменшення ціни на вказану суму.

1. Реалізація класу **Warehouse**

# Лістинг коду класу Warehouse:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WareHouseApp.Interfaces;

using WareHouseApp.Managers;

using WareHouseApp.Stores;

namespace WareHouseApp.Classes

{

internal class WareHouse

{

public int Id { get; set; }

public string Name { get; set; } = string.Empty;

public DateTime LastUpdate { get; set; }

public List<Product> Products { get; set; } = new List<Product>();

public IWareHouseManager? Manager { get; set; }

}

}

//Принцип SOLID: Liskov Substitution Principle (LSP)

//Клас Warehouse представляє склад, де зберігаються продукти.

//Він має властивості зберігати інформацію про продукти та їх кількість.

1. Реалізація класу **Reporting**

# Лістинг коду класу Reporting:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WareHouseApp.Interfaces;

using WareHouseApp.Stores;

namespace WareHouseApp.Classes

{

internal class Reporting : IReportingService

{

private Store \_store;

public Reporting(Store store)

{

\_store = store;

}

public void CreateReport(IReport report)

{

\_store.reports.Add(report);

}

public void GetProfitableInvoice()

{

Console.WriteLine("\n-----------Profitable Invoice----------\n");

var profInvoce = \_store.reports.FindAll(r => r.IsDeleted == false);

foreach(IReport prof in profInvoce)

{

prof.PrintReport();

}

}

public void GetExpenseInvoice()

{

Console.WriteLine("\n-----------Expense Invoice----------\n");

var profInvoce = \_store.reports.FindAll(r => r.IsDeleted == true);

foreach (var prof in profInvoce)

{

prof.PrintReport();

}

}

public void GetProductCount()

{

var profInvoce = \_store.reports.FindAll(r => r.IsDeleted == false);

Console.WriteLine($"Count of products {profInvoce.Count}");

}

public void GetReports()

{

foreach (Report report in \_store.reports)

{

report.PrintReport();

}

}

}

}

# Результат виконання програми:

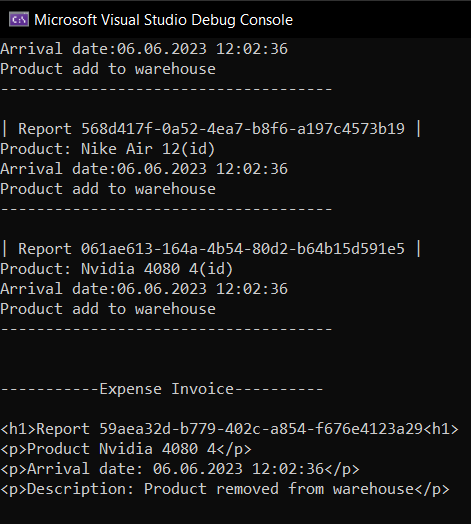


Рис.2.1. Реалізація завдання.

# 

Рис.2.2. UML діаграма.

**Висновок:** на лабораторній роботі я навчився дотримуватися принципів SOLID та обґрунтовувати їх. Навчиіся описувати дизайн програми за допомогою UML діаграм.