ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КПІ імені Ігоря Сікорського”

Кафедра програмного забезпечення комп’ютерних систем

**ЛАБОРАТОРНА РОБОТА №3**

з дисципліни

Програмування

C# .Net. Реалізація основних

принципів ООП мовою C#

Студента 2 курсу ФПМ

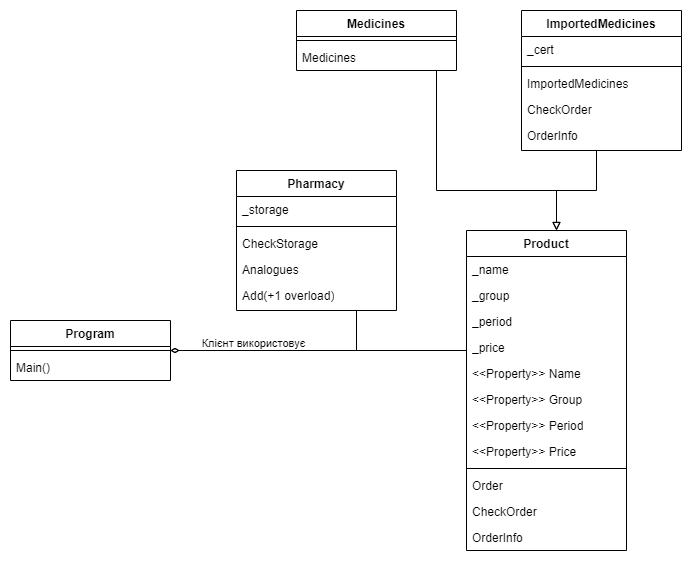
гр. КП-23 Верьовочкіна М. С.

Київ – 2024

**Варіант №6**

1. В аптеці продаються ліки різних виробників. Всі вони мають назву, групу ліків, до якої вони належать (антибіотики, протизапальні, шлункові тощо), ціну, термін зберігання. Однак, імпортні ліки повинні мати додатковий сертифікат про проходження препаратом лабораторного контролю в Україні. Крім того, у випадку замовлення покупцем імпортного препарату, фармацевт (якщо звернеться до бази даних ліків в аптеці) повинен побачити не тільки інформацію про наявність ліків в аптеці та їх ціну, а й дані про вітчизняні аналоги препарату, які як правило, є дешевшими. Розробити структуру класів, які можна використовувати для комп’ютерної обробки даних про ліки (як вітчизняні, так й імпортні) в аптеці.
2. Розробити модуль до програмного забезпечення, яке використовується в міліції. Даний модуль повинен забезпечити автоматичне створення електронної справи затриманого (окрім загальної анкети, де міститься загальна інформація про злочинця, дата та причини його затримання) у разі, якщо його затримано більше, ніж на мінімальний термін ув’язнення.
   1. Враховуючи необхідність створення класу аптеки з базою даних ліків, основного компоненту (ліків) та додаткового компоненту (імпортних ліків), доцільно використати шаблон «декоратор» у вигляді абстрактного класу (продукт) з базовими параметрами, який наслідують та реалізовують конкретний компонент (ліки) та власне декоратор (імпортні ліки). При цьому клієнт взаємодіє (використовує) саме з абстрактним класом. В такому разі немає потреби розбиття декоратора на абстрактний та конкретний.

Преставимо UML-діаграму структури класів:



Текст програми (головна функція визначає один з варіантів виконання програми в якості прикладу):

class Program

{

static void Main()

{

Product p1 = new Medicines("Paracetamol", "antipyretic", "2025 May", 180.30);

Product p2 = new Medicines("Citramon", "analgesics", "2028 June", 109.00);

Product p3 = new ImportedMedicines(new Medicines("Acetylsalicylic acid", "analgesics", "2024 May", 205.45), true);

Product p4 = new Medicines("Analgin", "analgesics", "2024 December", 23.40);

Product p5 = new ImportedMedicines(new Medicines("Ambroxol extra", "cough medicine", "2027 April", 456.88), false);

List<Product> products = new List<Product> { p1, p2, p3, p4, p5 };

Pharmacy.Add(products);

foreach (Product product in products)

product.CheckOrder(new Random().Next(1, 6));

}

}

public static class Pharmacy

{

private static List<Product> \_storage = new List<Product>();

public static Product CheckStorage(string name) => \_storage.Find(x => x.Name == name);

public static List<string> Analogues(string name)

{

List<string> a = new List<string>();

foreach(Product product in \_storage.FindAll(x => x.Group == CheckStorage(name).Group && x.Name != name))

a.Add(product.Name);

return a;

}

public static void Add(Product p) => \_storage.Add(p);

public static void Add(List<Product> lp) => \_storage = new List<Product>(lp);

}

public abstract class Product

{

protected string \_name;

protected string \_group;

protected string \_period;

protected double \_price;

public string Name => \_name;

public string Group => \_group;

public string Period => \_period;

public double Price => \_price;

protected virtual void Order(int count)

{

Console.WriteLine("You ordered the {0} in the amount of {1} pieces. Amount due: {2}\n",

\_name, count, Pharmacy.CheckStorage(\_name).\_price \* count);

}

public virtual void CheckOrder(int count)

{

if (Pharmacy.CheckStorage(\_name) == null) { Console.WriteLine("There is no such the product on the storage.\n"); return; }

OrderInfo();

Order(count);

}

protected virtual void OrderInfo()

{

Console.WriteLine("Name: {0}\nMedicine group: {1}\nPeriod till: {2}\nPrice: {3:0.00}\n", \_name, \_group, \_period, \_price);

}

}

public class Medicines : Product

{

public Medicines(string name, string group, string period, double price)

{

\_name = name;

\_group = group;

\_period = period;

\_price = price;

}

}

public class ImportedMedicines : Product

{

bool \_cert;

public ImportedMedicines(Medicines medicines, bool cert)

{

\_name = medicines.Name;

\_group = medicines.Group;

\_period = medicines.Period;

\_price = medicines.Price;

\_cert = cert;

}

public override void CheckOrder(int count)

{

if (!\_cert) { Console.WriteLine("The product is not certified. Cannot be sold."); return; }

if (Pharmacy.CheckStorage(\_name) == null) { Console.WriteLine("There is no such the product on the storage."); return; }

OrderInfo();

Order(count);

}

protected override void OrderInfo()

{

base.OrderInfo();

Console.WriteLine("The product is certified. Allowed for sale.");

Console.WriteLine("Domestic analogues of the drug: ");

List<string> a = Pharmacy.Analogues(\_name);

for (int i = 0; i < a.Count; i++)

Console.WriteLine(" {4}.\tName: {0}\n\tMedicine group: {1}\n\tPeriod till: {2}\n\tPrice: {3:0.00}\n", Pharmacy.CheckStorage(a[i]).Name,

Pharmacy.CheckStorage(a[i]).Group, Pharmacy.CheckStorage(a[i]).Period, Pharmacy.CheckStorage(a[i]).Price, i + 1);

}

}

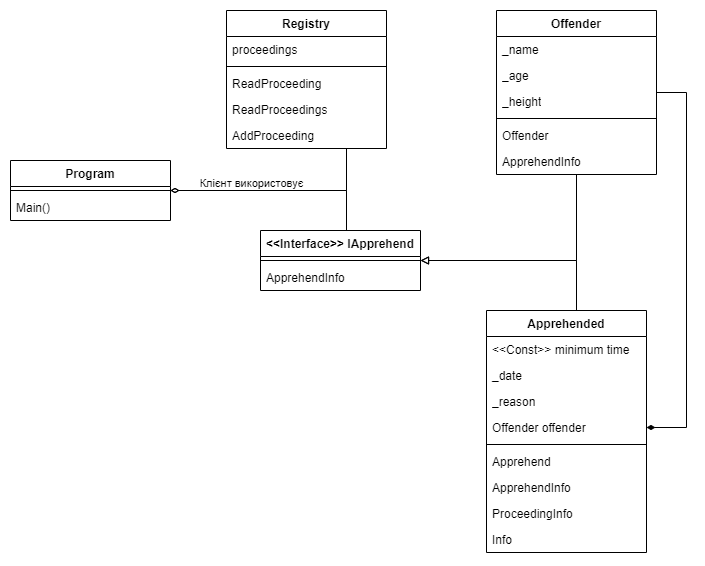
Приклад виконання розробленого коду:

A screenshot of a computer

Description automatically generated

* 1. Для визначення оптимального шаблону в цій задачі можна використати наступну логіку: поліція не може взаємодіяти із правопорушником і заводити на нього справу, не затримавши його, отже, виникає потреба у відкладеній ініціалізації, чого можна досягти за допомогою шаблону «заступник». Окрім цього, цей шаблон дозволяє здійснювати додаткові дії над запитом клієнта на шляху до компонента. Недоліком цього методу може бути його неуніверсальність у площині відповідності використаної логіки реальному законодавству, але цей недолік незначний у контексті умовності задачі.

Преставимо UML-діаграму структури класів:



Текст програми:

class Program

{

static void Main()

{

IApprehend ap = new Apprehended();

ap.Apprehend(3);

ap.Apprehend(5);

ap.Apprehend(6);

ap.Apprehend(3);

ap.Apprehend(3);

ap.Apprehend(18);

ap = new Apprehended();

ap.Apprehend(3);

ap.Apprehend(3);

ap.Apprehend(18);

Console.Write("Enter the offender's name to show its proceedings: ");

foreach (string proceeding in Registry.ReadProceedings(Console.ReadLine()))

Console.WriteLine("\n" + proceeding);

}

}

public static class Registry

{

private static List<string> proceedings = new List<string>();

public static string ReadProceeding(string name) => proceedings.Find(x => x.Contains(name));

public static List<string> ReadProceedings(string name) => proceedings.FindAll(x => x.Contains("Name: " + name));

public static void AddProceeding(string proceeding) => proceedings.Add(proceeding);

}

public interface IApprehend

{

string ApprehendInfo(int time);

void Apprehend(int time)

{

Console.WriteLine("\nApprehending information:\n{0}-------------------------------------", ApprehendInfo(time));

}

}

public class Offender : IApprehend

{

private string \_name;

private short \_age;

private short \_height;

public Offender()

{

Console.WriteLine("New offender was apprehended. Fill its case...");

Console.Write("Offender's name: ");

\_name = Console.ReadLine();

Console.Write("Offender's age: ");

\_age = short.Parse(Console.ReadLine());

Console.Write("Offender's height: ");

\_height = short.Parse(Console.ReadLine());

}

public string ApprehendInfo(int time)

{

return String.Format("Name: {0}\nAge: {1}\nHeight {2}\nDuration of apprehending: {3}\n", \_name, \_age, \_height, time);

}

}

public class Apprehended : IApprehend

{

const int minimum\_time = 3;

private string \_date;

private string \_reason;

private Offender offender;

public string ApprehendInfo(int time)

{

if (offender == null)

offender = new Offender();

return time > minimum\_time ? ProceedingInfo(time) : offender.ApprehendInfo(time);

}

private string ProceedingInfo(int time)

{

Info();

string proceeding = offender.ApprehendInfo(time) +

String.Format("Date of apprehending: {0}\nReason of apprehending: {1}\n", \_date, \_reason);

Registry.AddProceeding(proceeding);

return proceeding;

}

private void Info()

{

Console.WriteLine("New apprehending of the offender. Fill apprehendings details...");

Console.Write("Date of apprehending: ");

\_date = Console.ReadLine();

Console.Write("Reason of apprehending: ");

\_reason = Console.ReadLine();

}

}

Приклад виконання розробленого коду:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Як бачимо, програма коректно виводить інформацію з електронного реєстру, яку перед цим автоматично реєструє згідно з умовами задачі.

**Висновки:** в ході виконання лабораторної роботи було опановано роботу з такими особливостями мови C#, як структурні шаблони, вивчено відмінності між ними та способи доцільного підбору. Були продемонстровані різні види UML-діаграм для обраних шаблонів. Безпосередньо в роботі були використані шаблони «декоратор» та «заступник».