***Лабораторна робота № 4***

***Тема роботи***: Властивості та методи

***Мета роботи***: Освоїти основні концепції роботи з властивостями, які є важливими складовими класів у мові програмування C#. Здобути навички створення властивостей для забезпечення доступу до даних в класах.

***Завдання***: Удосконалити та розширити програму, яку було створено під час виконання лабораторної роботи №3 наступним чином:

• Для двох полів, що були описані раніше в класі, створити властивості, які дозволять читати та змінювати їх значення.

• Додатково створити дві автоматичні властивості, які описують додаткові характеристики об'єкта на ваш вибір.

• Створити властивість, доступну тільки для читання, яка розраховує певне значення характеристики об'єкта на ваш вибір.

• Внести зміни в конструктори класів, додавши нові властивості та замінивши оператори присвоювання значень полів на оператори присвоювання значень властивостей.

• Розширити програму методами, які мають різні типи передачі параметрів: передача за значенням, передача за посиланням (використовуючи ключове слово ref).

• Перевірити роботу створених методів, додаючи оператори присвоювання значень властивостям.

**Код:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOP\_Lab4

{

class Student

{

private string surname;

private string phone;

private int exam1;

private int exam2;

private int exam3;

public string Surname

{

get { return surname; }

set { surname = value; }

}

public string Phone

{

get { return phone; }

set { phone = value; }

}

public string Group { get; set; }

public int Course { get; set; }

public double Average

{

get { return (exam1 + exam2 + exam3) / 3.0; }

}

public Student()

{

Surname = "Невідомий";

Phone = "0000000000";

exam1 = 0; exam2 = 0; exam3 = 0;

Group = "N/A"; Course = 0;

}

public Student(string s, string p, int e1, int e2, int e3)

{

Surname = s; Phone = p;

exam1 = e1; exam2 = e2; exam3 = e3;

Group = "Default"; Course = 1;

}

public void SetExams(int e1, int e2, int e3)

{

exam1 = e1; exam2 = e2; exam3 = e3;

}

public void ChangePhone(ref string newPhone)

{

Phone = newPhone;

}

public override string ToString()

{

return $"Прізвище: {Surname}, Телефон: {Phone}, Оцінки: {exam1}, {exam2}, {exam3}, " +

$"Група: {Group}, Курс: {Course}, Середній бал: {Average:F2}";

}

}

class Program

{

static void Main()

{

Console.OutputEncoding = UTF8Encoding.UTF8;

Student st = new Student("Купиро", "0501234567", 80, 72, 82);

st.Group = "БІП-1-24";

st.Course = 2;

string newPhone = "0931112233";

st.ChangePhone(ref newPhone);

Console.WriteLine(st);

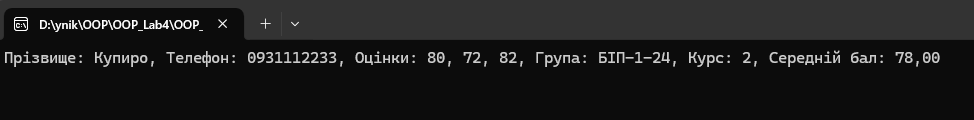
Console.ReadLine();

}

}

}

**Вивід результату:**



**Контрольні запитання:**

**1. Що таке властивість класу?**  
Властивість — це елемент класу, який надає контрольований доступ до приватних полів через аксесори get і set.

**2. Що таке автоматична властивість?**  
Це властивість, для якої не потрібно явно створювати поле — компілятор сам створює приховане поле.

public string Name { get; set; }

**3. Для чого призначені аксесори set і get?**

get — повертає значення властивості.

set — встановлює нове значення (через параметр value).

**4. Які специфікатори доступу можливо використовувати для властивостей?**  
public, private, protected, internal, protected internal.

**5. Чи можна використовувати різні специфікатори доступу для аксесорів set і get?**  
Так. Наприклад, public get; private set; — тоді читати можна ззовні, а змінювати лише всередині класу.

**6. Як відбувається задання та зчитування полів класу за допомогою властивостей?**  
При зверненні до властивості викликаються аксесори: obj.Property = 10; → викликає set, а Console.WriteLine(obj.Property); → викликає get.

**7. Чи може клас містити властивості, які не пов’язані з окремими полями?**  
Так, може. Наприклад, властивість може повертати обчислюване значення (наприклад, середній бал).

**8. Наведіть загальний синтаксис опису властивості.**

[модифікатор] тип Назва

{

get { ... }

set { ... }

}

**9. Навести приклади різних типів передачі параметрів.**

За значенням (звичайний випадок):

void Foo(int x) { x = 5; }

За посиланням (ref):

void Foo(ref int x) { x = 5; }

Вихідний параметр (out):

void Foo(out int x) { x = 5; }

Масив параметрів (params):

void Foo(params int[] numbers) { }

**10. Які існують специфікатори рівня доступу?**

public — доступ без обмежень

private — доступ тільки всередині класу

protected — доступ у класі та його нащадках

internal — доступ лише в межах збірки

protected internal — доступ у нащадках і в межах збірки

**Висновки:** Отже, я освоїв основні концепції роботи з властивостями, які є важливими складовими класів у мові програмування C#. Здобув навички створення властивостей для забезпечення доступу до даних в класах.