Лабораторна робота №4  
Богуславський Альберт КН-922В

Варіант 1

Завдання 1

Розробити клас згідно до свого варіанта. Включити до класу методи **set (…), get (…), show (…)** та ін., використовуючи принцип інкапсуляції. Окрім гетерів, сетерів конструкторів та інших стандартних методів розробити не менше двох додаткових методів, які реалізують власний функціонал об’єкту відповідної предметної галузі. Написати програму, яка створює список, масив чи колекцію, об’єктів на основі відповідного класу. Під час створення використати випадковий підхід для генерації числової та текстової інформації об’єктів списку. Передбачити можливість додання, редагування та видалення об’єктів зі списку за певними критеріями. В програмі реалізувати функціонал за своїм варіантом.

1. **Student**: Прізвище, Ім'я, По-батькові, Дата народження, Адреса, Телефон, Факультет, Курс. Створити масив об'єктів. Вивести:

• перелік студентів заданого факультету;

• списки студентів для кожного факультету та курсу;

• список студентів, які народилися після цього року.

Код програми:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task\_1

{

class Student

{

// Приватні поля для інкапсуляції

private string lastName;

private string firstName;

private string middleName;

private int birthDate;

private string address;

private string phoneNumber;

private string faculty;

private int course;

// Конструктор

public Student(string lastName, string firstName, string middleName, int birthDate, string address, string phoneNumber, string faculty, int course)

{

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

this.address = address;

this.phoneNumber = phoneNumber;

this.faculty = faculty;

this.course = course;

}

// Характеристики (гетери та сетери)

public string LastName { get => lastName; set => lastName = value; }

public string FirstName { get => firstName; set => firstName = value; }

public string MiddleName { get => middleName; set => middleName = value; }

public int BirthDate { get => birthDate; set => birthDate = value; }

public string Address { get => address; set => address = value; }

public string PhoneNumber { get => phoneNumber; set => phoneNumber = value; }

public string Faculty { get => faculty; set => faculty = value; }

public int Course { get => course; set => course = value; }

//Метод для виводу інформації про студентів

public void Show()

{

Console.WriteLine($"{LastName} {FirstName} {MiddleName}, {birthDate}, {Address}, {PhoneNumber}, Факультет: {Faculty}, Курс: {Course}");

}

// Метод для визначення факультета студента

public bool IsFromFaculty(string faculty)

{

return this.faculty.Equals(faculty, StringComparison.OrdinalIgnoreCase);

}

}

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.InputEncoding = System.Text.Encoding.Unicode;

int a = 0, b = 0, c = 0;

List<Student> students = new List<Student>();

// Генерація випадкової інформації про студентів

Random random = new Random();

string[] faculties = { "Інженерія", "Фізика", "Хімія", "КН" };

for (int i = 0; i < 20; i++)

{

students.Add(new Student(

$"Прізвище\_{i}",

$"Ім'я\_{i}",

$"По-батькові\_{i}",

random.Next(2000, 2030),

$"Адреса\_{i}",

$"123-456-{i:D4}",

faculties[random.Next(faculties.Length)],

random.Next(1, 5)

));

}

// Додати нового студента

Console.WriteLine("\nХочете додати нового студента? (1/0):");

a = Convert.ToInt32( Console.ReadLine() );

if (a == 1)

{

Console.WriteLine("Введіть інформацію для нового студента:");

Console.Write("Прізвище: ");

string lastName = Console.ReadLine();

Console.Write("Ім'я: ");

string firstName = Console.ReadLine();

Console.Write("По-батькові: ");

string middleName = Console.ReadLine();

Console.Write("Рік народження: ");

int birthDate = Convert.ToInt32(Console.ReadLine());

Console.Write("Адреса: ");

string address = Console.ReadLine();

Console.Write("Телефонний номер: ");

string phoneNumber = Console.ReadLine();

Console.Write("Факультет: ");

string faculty = Console.ReadLine();

Console.Write("Курс: ");

int course = Convert.ToInt32(Console.ReadLine());

// Додаємо нового студента до списку

students.Add(new Student(lastName, firstName, middleName, birthDate, address, phoneNumber, faculty, course));

Console.WriteLine("\nНового студента додано!");

}

// Вивід всіх студентів

Console.WriteLine("Всі студенти:");

students.ForEach(s => s.Show());

// Редагування студента за номером у списку

Console.WriteLine("\nХочете редагувати студента? (1/0):");

b = Convert.ToInt32(Console.ReadLine());

if (b == 1)

{

Console.WriteLine("Введіть номер студента (від 0 до {0}):", students.Count - 1);

int studentIndex = Convert.ToInt32(Console.ReadLine());

// Перевірка на правильність введеного індексу

if (studentIndex >= 0 && studentIndex < students.Count)

{

// Показуємо поточні дані студента

Student studentToEdit = students[studentIndex];

Console.WriteLine("\nПоточні дані студента:");

studentToEdit.Show();

Console.WriteLine("\nРедагуйте інформацію:");

Console.Write("Прізвище ({0}): ", studentToEdit.LastName);

string newLastName = Console.ReadLine();

if (!string.IsNullOrEmpty(newLastName)) studentToEdit.LastName = newLastName;

Console.Write("Ім'я ({0}): ", studentToEdit.FirstName);

string newFirstName = Console.ReadLine();

if (!string.IsNullOrEmpty(newFirstName)) studentToEdit.FirstName = newFirstName;

Console.Write("По-батькові ({0}): ", studentToEdit.MiddleName);

string newMiddleName = Console.ReadLine();

if (!string.IsNullOrEmpty(newMiddleName)) studentToEdit.MiddleName = newMiddleName;

Console.Write("Рік народження ({0}): ", studentToEdit.BirthDate);

string newBirthDate = Console.ReadLine();

if (!string.IsNullOrEmpty(newBirthDate)) studentToEdit.BirthDate = Convert.ToInt32(newBirthDate);

Console.Write("Адреса ({0}): ", studentToEdit.Address);

string newAddress = Console.ReadLine();

if (!string.IsNullOrEmpty(newAddress)) studentToEdit.Address = newAddress;

Console.Write("Телефонний номер ({0}): ", studentToEdit.PhoneNumber);

string newPhoneNumber = Console.ReadLine();

if (!string.IsNullOrEmpty(newPhoneNumber)) studentToEdit.PhoneNumber = newPhoneNumber;

Console.Write("Факультет ({0}): ", studentToEdit.Faculty);

string newFaculty = Console.ReadLine();

if (!string.IsNullOrEmpty(newFaculty)) studentToEdit.Faculty = newFaculty;

Console.Write("Курс ({0}): ", studentToEdit.Course);

string newCourse = Console.ReadLine();

if (!string.IsNullOrEmpty(newCourse)) studentToEdit.Course = Convert.ToInt32(newCourse);

Console.WriteLine("\nДані студента оновлені!");

// Показуємо оновлені дані студента

studentToEdit.Show();

}

else

{

Console.WriteLine("Невірний номер студента.");

}

}

// Вивід всіх студентів

Console.WriteLine("Всі студенти:");

students.ForEach(s => s.Show());

// Видалення студентів за роком народження

Console.WriteLine("\nХочете видалити студентів за роком народження? (1/0):");

c = Convert.ToInt32(Console.ReadLine());

if (c == 1)

{

Console.WriteLine("Введіть рік народження студентів для видалення:");

int yearToDelete = Convert.ToInt32(Console.ReadLine());

// Видаляємо студентів, рік народження яких дорівнює введеному

int initialCount = students.Count;

students.RemoveAll(s => s.BirthDate == yearToDelete);

int deletedCount = initialCount - students.Count;

if (deletedCount > 0)

{

Console.WriteLine($"\nВидалено {deletedCount} студентів, народжених в {yearToDelete} році.");

}

else

{

Console.WriteLine("Не знайдено студентів з таким роком народження.");

}

}

// Вивід всіх студентів

Console.WriteLine("Всі студенти:");

students.ForEach(s => s.Show());

// Пошук по факультету

Console.WriteLine("\nВведіть факультет:");

string inputFaculty = Console.ReadLine();

var facultyStudents = students.Where(s => s.IsFromFaculty(inputFaculty)).ToList();

if (facultyStudents.Any())

{

Console.WriteLine($"\nСтуденти факультету {inputFaculty}:");

facultyStudents.ForEach(s => s.Show());

}

else

{

Console.WriteLine($"Студентів не знайдено {inputFaculty}.");

}

// Пошук по факультету та курсу

Console.WriteLine("\nВведіть факультет:");

string inputFacultyForCourse = Console.ReadLine();

Console.WriteLine("Введіть курс:");

int inputCourse = Convert.ToInt32(Console.ReadLine());

var facultyAndCourseStudents = students.Where(s => s.IsFromFaculty(inputFacultyForCourse) && s.Course == inputCourse).ToList();

if (facultyAndCourseStudents.Any())

{

Console.WriteLine($"\nСтуденти факультету {inputFacultyForCourse} та курсу {inputCourse}:");

facultyAndCourseStudents.ForEach(s => s.Show());

}

else

{

Console.WriteLine($"Студентів не знайдено на факультеті {inputFacultyForCourse} та {inputCourse} курсі.");

}

// Народилися після 2024 року

int year = 2024;

var bornAfterYear = students.Where(s => s.BirthDate > year).ToList();

if (bornAfterYear.Any())

{

Console.WriteLine($"\nСтуденти народжені після {year} року:");

bornAfterYear.ForEach(s => s.Show());

}

else

{

Console.WriteLine($"Немає студентів народжених після {year} року.");

}

Console.ReadLine();

}

}

}

Короткий опис програми

Додавання нового студента – користувач може ввести дані нового студента (прізвище, ім'я, по-батькові, рік народження, адреса, телефон, факультет, курс).

Редагування даних студента – користувач може змінювати інформацію про студента, вибравши його з списку за індексом.

Видалення студентів – можна видалити студентів, народжених у певному році.

Пошук студентів:

За факультетом.

За факультетом і курсом.

За роком народження.

Усі операції здійснюються через консольний інтерфейс, де користувач вводить необхідні дані. Програма також генерує випадкових студентів для демонстрації функцій.

Виконання:

Хочете додати нового студента? (1/0):

1

Введіть інформацію для нового студента:

Прізвище: Іванов

Ім'я: Іван

По-батькові: Іванович

Рік народження: 2009

Адреса: Вул.Галицька 10

Телефонний номер: 380991111111

Факультет: Буівельництво

Курс: 3

Нового студента додано!

Всі студенти:

Прізвище\_0 Ім'я\_0 По-батькові\_0, 2024, Адреса\_0, 123-456-0000, Факультет: КН, Курс: 4

Прізвище\_1 Ім'я\_1 По-батькові\_1, 2019, Адреса\_1, 123-456-0001, Факультет: Хімія, Курс: 2

Прізвище\_2 Ім'я\_2 По-батькові\_2, 2013, Адреса\_2, 123-456-0002, Факультет: Хімія, Курс: 2

Прізвище\_3 Ім'я\_3 По-батькові\_3, 2007, Адреса\_3, 123-456-0003, Факультет: Інженерія, Курс: 3

Прізвище\_4 Ім'я\_4 По-батькові\_4, 2028, Адреса\_4, 123-456-0004, Факультет: КН, Курс: 3

Прізвище\_5 Ім'я\_5 По-батькові\_5, 2011, Адреса\_5, 123-456-0005, Факультет: КН, Курс: 3

Прізвище\_6 Ім'я\_6 По-батькові\_6, 2020, Адреса\_6, 123-456-0006, Факультет: КН, Курс: 1

Прізвище\_7 Ім'я\_7 По-батькові\_7, 2003, Адреса\_7, 123-456-0007, Факультет: КН, Курс: 4

Прізвище\_8 Ім'я\_8 По-батькові\_8, 2011, Адреса\_8, 123-456-0008, Факультет: Інженерія, Курс: 1

Прізвище\_9 Ім'я\_9 По-батькові\_9, 2017, Адреса\_9, 123-456-0009, Факультет: КН, Курс: 4

Прізвище\_10 Ім'я\_10 По-батькові\_10, 2024, Адреса\_10, 123-456-0010, Факультет: Фізика, Курс: 4

Прізвище\_11 Ім'я\_11 По-батькові\_11, 2007, Адреса\_11, 123-456-0011, Факультет: Інженерія, Курс: 1

Прізвище\_12 Ім'я\_12 По-батькові\_12, 2026, Адреса\_12, 123-456-0012, Факультет: Хімія, Курс: 4

Прізвище\_13 Ім'я\_13 По-батькові\_13, 2003, Адреса\_13, 123-456-0013, Факультет: Фізика, Курс: 2

Прізвище\_14 Ім'я\_14 По-батькові\_14, 2011, Адреса\_14, 123-456-0014, Факультет: Інженерія, Курс: 2

Прізвище\_15 Ім'я\_15 По-батькові\_15, 2023, Адреса\_15, 123-456-0015, Факультет: КН, Курс: 2

Прізвище\_16 Ім'я\_16 По-батькові\_16, 2019, Адреса\_16, 123-456-0016, Факультет: Фізика, Курс: 1

Прізвище\_17 Ім'я\_17 По-батькові\_17, 2019, Адреса\_17, 123-456-0017, Факультет: Хімія, Курс: 2

Прізвище\_18 Ім'я\_18 По-батькові\_18, 2002, Адреса\_18, 123-456-0018, Факультет: Хімія, Курс: 1

Прізвище\_19 Ім'я\_19 По-батькові\_19, 2001, Адреса\_19, 123-456-0019, Факультет: Фізика, Курс: 3

Іванов Іван Іванович, 2009, Вул.Галицька 10, 380991111111, Факультет: Буівельництво, Курс: 3

Хочете редагувати студента? (1/0):

1

Введіть номер студента (від 0 до 20):

1

Поточні дані студента:

Прізвище\_1 Ім'я\_1 По-батькові\_1, 2019, Адреса\_1, 123-456-0001, Факультет: Хімія, Курс: 2

Редагуйте інформацію:

Прізвище (Прізвище\_1): Аедрій

Ім'я (Ім'я\_1): Галицький

По-батькові (По-батькові\_1): Андрійович

Рік народження (2019): 2007

Адреса (Адреса\_1): Вул.Заміщанська 34

Телефонний номер (123-456-0001): 380992222222

Факультет (Хімія): Маркетинг

Курс (2): 3

Дані студента оновлені!

Аедрій Галицький Андрійович, 2007, Вул.Заміщанська 34, 380992222222, Факультет: Маркетинг, Курс: 3

Всі студенти:

Прізвище\_0 Ім'я\_0 По-батькові\_0, 2024, Адреса\_0, 123-456-0000, Факультет: КН, Курс: 4

Аедрій Галицький Андрійович, 2007, Вул.Заміщанська 34, 380992222222, Факультет: Маркетинг, Курс: 3

Прізвище\_2 Ім'я\_2 По-батькові\_2, 2013, Адреса\_2, 123-456-0002, Факультет: Хімія, Курс: 2

Прізвище\_3 Ім'я\_3 По-батькові\_3, 2007, Адреса\_3, 123-456-0003, Факультет: Інженерія, Курс: 3

Прізвище\_4 Ім'я\_4 По-батькові\_4, 2028, Адреса\_4, 123-456-0004, Факультет: КН, Курс: 3

Прізвище\_5 Ім'я\_5 По-батькові\_5, 2011, Адреса\_5, 123-456-0005, Факультет: КН, Курс: 3

Прізвище\_6 Ім'я\_6 По-батькові\_6, 2020, Адреса\_6, 123-456-0006, Факультет: КН, Курс: 1

Прізвище\_7 Ім'я\_7 По-батькові\_7, 2003, Адреса\_7, 123-456-0007, Факультет: КН, Курс: 4

Прізвище\_8 Ім'я\_8 По-батькові\_8, 2011, Адреса\_8, 123-456-0008, Факультет: Інженерія, Курс: 1

Прізвище\_9 Ім'я\_9 По-батькові\_9, 2017, Адреса\_9, 123-456-0009, Факультет: КН, Курс: 4

Прізвище\_10 Ім'я\_10 По-батькові\_10, 2024, Адреса\_10, 123-456-0010, Факультет: Фізика, Курс: 4

Прізвище\_11 Ім'я\_11 По-батькові\_11, 2007, Адреса\_11, 123-456-0011, Факультет: Інженерія, Курс: 1

Прізвище\_12 Ім'я\_12 По-батькові\_12, 2026, Адреса\_12, 123-456-0012, Факультет: Хімія, Курс: 4

Прізвище\_13 Ім'я\_13 По-батькові\_13, 2003, Адреса\_13, 123-456-0013, Факультет: Фізика, Курс: 2

Прізвище\_14 Ім'я\_14 По-батькові\_14, 2011, Адреса\_14, 123-456-0014, Факультет: Інженерія, Курс: 2

Прізвище\_15 Ім'я\_15 По-батькові\_15, 2023, Адреса\_15, 123-456-0015, Факультет: КН, Курс: 2

Прізвище\_16 Ім'я\_16 По-батькові\_16, 2019, Адреса\_16, 123-456-0016, Факультет: Фізика, Курс: 1

Прізвище\_17 Ім'я\_17 По-батькові\_17, 2019, Адреса\_17, 123-456-0017, Факультет: Хімія, Курс: 2

Прізвище\_18 Ім'я\_18 По-батькові\_18, 2002, Адреса\_18, 123-456-0018, Факультет: Хімія, Курс: 1

Прізвище\_19 Ім'я\_19 По-батькові\_19, 2001, Адреса\_19, 123-456-0019, Факультет: Фізика, Курс: 3

Іванов Іван Іванович, 2009, Вул.Галицька 10, 380991111111, Факультет: Буівельництво, Курс: 3

Хочете видалити студентів за роком народження? (1/0):

1

Введіть рік народження студентів для видалення:

2013

Видалено 1 студентів, народжених в 2013 році.

Всі студенти:

Прізвище\_0 Ім'я\_0 По-батькові\_0, 2024, Адреса\_0, 123-456-0000, Факультет: КН, Курс: 4

Аедрій Галицький Андрійович, 2007, Вул.Заміщанська 34, 380992222222, Факультет: Маркетинг, Курс: 3

Прізвище\_3 Ім'я\_3 По-батькові\_3, 2007, Адреса\_3, 123-456-0003, Факультет: Інженерія, Курс: 3

Прізвище\_4 Ім'я\_4 По-батькові\_4, 2028, Адреса\_4, 123-456-0004, Факультет: КН, Курс: 3

Прізвище\_5 Ім'я\_5 По-батькові\_5, 2011, Адреса\_5, 123-456-0005, Факультет: КН, Курс: 3

Прізвище\_6 Ім'я\_6 По-батькові\_6, 2020, Адреса\_6, 123-456-0006, Факультет: КН, Курс: 1

Прізвище\_7 Ім'я\_7 По-батькові\_7, 2003, Адреса\_7, 123-456-0007, Факультет: КН, Курс: 4

Прізвище\_8 Ім'я\_8 По-батькові\_8, 2011, Адреса\_8, 123-456-0008, Факультет: Інженерія, Курс: 1

Прізвище\_9 Ім'я\_9 По-батькові\_9, 2017, Адреса\_9, 123-456-0009, Факультет: КН, Курс: 4

Прізвище\_10 Ім'я\_10 По-батькові\_10, 2024, Адреса\_10, 123-456-0010, Факультет: Фізика, Курс: 4

Прізвище\_11 Ім'я\_11 По-батькові\_11, 2007, Адреса\_11, 123-456-0011, Факультет: Інженерія, Курс: 1

Прізвище\_12 Ім'я\_12 По-батькові\_12, 2026, Адреса\_12, 123-456-0012, Факультет: Хімія, Курс: 4

Прізвище\_13 Ім'я\_13 По-батькові\_13, 2003, Адреса\_13, 123-456-0013, Факультет: Фізика, Курс: 2

Прізвище\_14 Ім'я\_14 По-батькові\_14, 2011, Адреса\_14, 123-456-0014, Факультет: Інженерія, Курс: 2

Прізвище\_15 Ім'я\_15 По-батькові\_15, 2023, Адреса\_15, 123-456-0015, Факультет: КН, Курс: 2

Прізвище\_16 Ім'я\_16 По-батькові\_16, 2019, Адреса\_16, 123-456-0016, Факультет: Фізика, Курс: 1

Прізвище\_17 Ім'я\_17 По-батькові\_17, 2019, Адреса\_17, 123-456-0017, Факультет: Хімія, Курс: 2

Прізвище\_18 Ім'я\_18 По-батькові\_18, 2002, Адреса\_18, 123-456-0018, Факультет: Хімія, Курс: 1

Прізвище\_19 Ім'я\_19 По-батькові\_19, 2001, Адреса\_19, 123-456-0019, Факультет: Фізика, Курс: 3

Іванов Іван Іванович, 2009, Вул.Галицька 10, 380991111111, Факультет: Буівельництво, Курс: 3

Введіть факультет:

КН

Студенти факультету КН:

Прізвище\_0 Ім'я\_0 По-батькові\_0, 2024, Адреса\_0, 123-456-0000, Факультет: КН, Курс: 4

Прізвище\_4 Ім'я\_4 По-батькові\_4, 2028, Адреса\_4, 123-456-0004, Факультет: КН, Курс: 3

Прізвище\_5 Ім'я\_5 По-батькові\_5, 2011, Адреса\_5, 123-456-0005, Факультет: КН, Курс: 3

Прізвище\_6 Ім'я\_6 По-батькові\_6, 2020, Адреса\_6, 123-456-0006, Факультет: КН, Курс: 1

Прізвище\_7 Ім'я\_7 По-батькові\_7, 2003, Адреса\_7, 123-456-0007, Факультет: КН, Курс: 4

Прізвище\_9 Ім'я\_9 По-батькові\_9, 2017, Адреса\_9, 123-456-0009, Факультет: КН, Курс: 4

Прізвище\_15 Ім'я\_15 По-батькові\_15, 2023, Адреса\_15, 123-456-0015, Факультет: КН, Курс: 2

Введіть факультет:

КН

Введіть курс:

3

Студенти факультету КН та курсу 3:

Прізвище\_4 Ім'я\_4 По-батькові\_4, 2028, Адреса\_4, 123-456-0004, Факультет: КН, Курс: 3

Прізвище\_5 Ім'я\_5 По-батькові\_5, 2011, Адреса\_5, 123-456-0005, Факультет: КН, Курс: 3

Студенти народжені після 2024 року:

Прізвище\_4 Ім'я\_4 По-батькові\_4, 2028, Адреса\_4, 123-456-0004, Факультет: КН, Курс: 3

Прізвище\_12 Ім'я\_12 По-батькові\_12, 2026, Адреса\_12, 123-456-0012, Факультет: Хімія, Курс: 4

Завдання 2

Для класу з минулого завдання, ретельно занурившись у предметну область власного варіанта, створити абстрактний клас чи інтерфейс, який має стати предком, зв’язати ці класи відносинами спадкування.

Окрім вашого класу додати до абстрактного класу чи інтерфейсу, створеному у попередньому пункті, ще два класи-спадкоємця першого рівня.

Для одного з доданих у попередньому завданні класів створити не менше двох спадкоємців другого рівня спадкування.

Створити масив чи колекцію об’єктів усіх рівнів спадкування. Виконати відповідні функції над поліморфним списком із завдання власного варіанта.

Короткий опис програми

Програма підтримує три типи стипендій:

Стипендія (Scholarship): Загальний клас для стипендій.

Соціальна стипендія (SocialScholarship): Клас для соціальної стипендії.

Академічна стипендія (AcademicScholarship): Клас для академічної стипендії.

Клас AVM додає інформацію про середній бал студента. Усі класи підтримують поліморфізм, дозволяючи виводити інформацію за допомогою методу Show.

Програма створює список студентів з різними типами стипендій і виводить їх інформацію, розділяючи студентів на групи залежно від типу стипендії (соціальна або академічна).

Код програми:

using System;

using System.Collections.Generic;

using System.Linq;

namespace Task\_2

{

// Абстрактний клас "Національність"

abstract class Nationality

{

// Властивість для зберігання національності

public abstract string Nation { get; set; }

// Абстрактний метод для виведення інформації про національність

public abstract void ShowNationality();

}

class Student : Nationality

{

// Приватні поля для інкапсуляції

private string lastName;

private string firstName;

private string middleName;

private int birthDate;

private string address;

private string phoneNumber;

private string faculty;

private int course;

// Властивість для зберігання національності

public override string Nation { get; set; }

// Конструктор

public Student(string lastName, string firstName, string middleName, int birthDate, string address, string phoneNumber, string faculty, int course, string nation)

{

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.birthDate = birthDate;

this.address = address;

this.phoneNumber = phoneNumber;

this.faculty = faculty;

this.course = course;

this.Nation = nation;

}

// Гетери та сетери

public string LastName { get => lastName; set => lastName = value; }

public string FirstName { get => firstName; set => firstName = value; }

public string MiddleName { get => middleName; set => middleName = value; }

public int BirthDate { get => birthDate; set => birthDate = value; }

public string Address { get => address; set => address = value; }

public string PhoneNumber { get => phoneNumber; set => phoneNumber = value; }

public string Faculty { get => faculty; set => faculty = value; }

public int Course { get => course; set => course = value; }

// Реалізація методу ShowNationality

public override void ShowNationality()

{

Console.WriteLine($"Національність: {Nation}");

}

// Метод для виводу інформації про студентів

public virtual void Show()

{

Console.WriteLine($"{LastName} {FirstName} {MiddleName}, {birthDate}, {Address}, {PhoneNumber}, Факультет: {Faculty}, Курс: {Course}");

ShowNationality();

}

// Метод для визначення факультета студента

public bool IsFromFaculty(string faculty)

{

return this.faculty.Equals(faculty, StringComparison.OrdinalIgnoreCase);

}

}

class AVM : Student

{

public double AvM { get; set; }

public AVM(string lastName, string firstName, string middleName, int birthDate, string address, string phoneNumber, string faculty, int course, string nation, double avm)

: base(lastName, firstName, middleName, birthDate, address, phoneNumber, faculty, course, nation)

{

AvM = avm;

}

public override void Show()

{

base.Show();

Console.WriteLine($"Середній бал: {AvM:F2}");

}

}

// Клас "Стипендія"

class Scholarship : Student

{

public double Amount { get; set; }

public Scholarship(string lastName, string firstName, string middleName, int birthDate, string address, string phoneNumber, string faculty, int course, string nation, double amount)

: base(lastName, firstName, middleName, birthDate, address, phoneNumber, faculty, course, nation)

{

Amount = amount;

}

public void ShowScholarship()

{

Console.WriteLine($"Стипендія: {Amount} грн.");

}

public override void Show()

{

base.Show();

ShowScholarship();

}

}

// Клас "Соціальна стипендія"

class SocialScholarship : Scholarship

{

public SocialScholarship(string lastName, string firstName, string middleName, int birthDate, string address, string phoneNumber, string faculty, int course, string nation, double amount)

: base(lastName, firstName, middleName, birthDate, address, phoneNumber, faculty, course, nation, amount)

{ }

public void SocScholarship()

{

Console.WriteLine($"Соціальна стипендія: {Amount} грн.");

}

public override void Show()

{

base.Show();

SocScholarship();

}

}

// Клас "Академічна стипендія"

class AcademicScholarship : Scholarship

{

public AcademicScholarship(string lastName, string firstName, string middleName, int birthDate, string address, string phoneNumber, string faculty, int course, string nation, double amount)

: base(lastName, firstName, middleName, birthDate, address, phoneNumber, faculty, course, nation, amount)

{ }

public void AcademScholarship()

{

Console.WriteLine($"Академічна стипендія: {Amount} грн.");

}

public override void Show()

{

base.Show();

AcademScholarship();

}

}

internal class Program

{

static void Main(string[] args)

{

List<Student> students = new List<Student>();

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.InputEncoding = System.Text.Encoding.Unicode;

// Студенти з різними типами стипендій

students.Add(new AVM("Іванов", "Іван", "Іванович", 2000, "Адреса 1", "123-456-0001", "Інженерія", 2, "Українець", 4.5));

students.Add(new SocialScholarship("Петренко", "Петро", "Петрович", 1999, "Адреса 2", "123-456-0002", "Фізика", 3, "Поляк", 1250)); // Соціальна стипендія

students.Add(new AcademicScholarship("Сидоренко", "Сидір", "Сидорович", 2001, "Адреса 3", "123-456-0003", "Хімія", 4, "Німець", 2000)); // Академічна стипендія

students.Add(new SocialScholarship("Коваленко", "Марія", "Олександрівна", 2002, "Адреса 4", "123-456-0004", "КН", 1, "Француз", 1250)); // Соціальна стипендія

students.Add(new AcademicScholarship("Бондаренко", "Тарас", "Володимирович", 2003, "Адреса 5", "123-456-0005", "Фізика", 2, "Українець", 2000)); // Академічна стипендія

// Поліморфний виклик

Console.WriteLine("Інформація про студентів:");

foreach (var student in students)

{

student.Show();

Console.WriteLine();

}

// Масиви стипендій

var socialScholarships = students.OfType<SocialScholarship>().ToList();

var academicScholarships = students.OfType<AcademicScholarship>().ToList();

// Виведення студентів з соціальною стипендією

Console.WriteLine("Студенти з соціальною стипендією:");

foreach (var student in socialScholarships)

{

student.Show();

}

// Виведення студентів з академічною стипендією

Console.WriteLine("\nСтуденти з академічною стипендією:");

foreach (var student in academicScholarships)

{

student.Show();

}

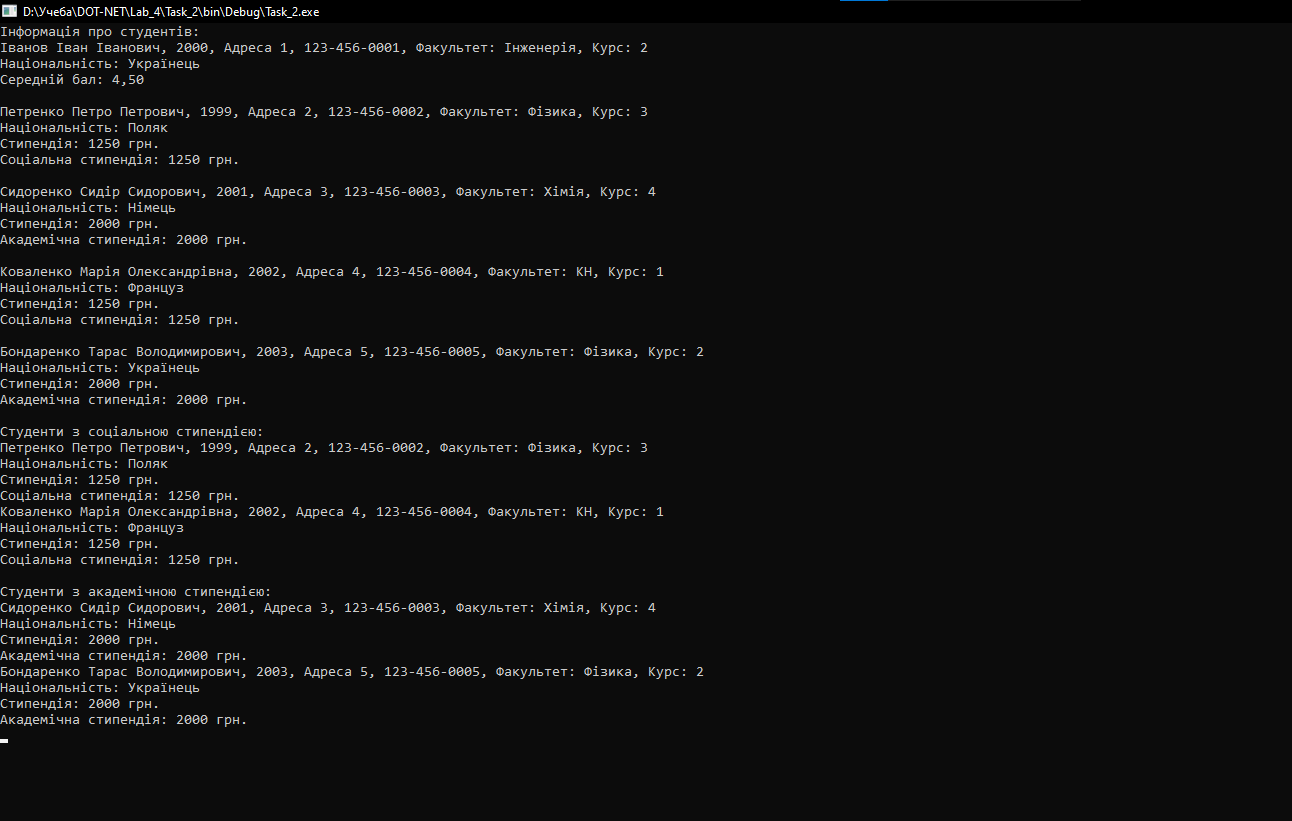
Console.ReadLine();

}

}

}

Виконання



Завдання 3

Розробити клас та перевантажити оператори згідно до свого варіанта. В головній функції програми протестувати роботу створених класів на прикладі використання окремих об’єктів та масивів чи колекцій цих об’єктів. Під час створення об’єктів застосувати випадковий підхід чи зчитування інформації з файлів.

"Комплексне число" - **Complex**. Клас повинен містити кілька конструкторів та оператори для складання, віднімання, множення, поділу, присвоювання. Створити два вектори довільної розмірності з комплексних координат. Передати їх у функцію, яка виконує складання комплексних векторів.

Короткий опис роботи   
Клас Complex:

Має два поля: r (реальна частина) і i (уявна частина) для зберігання комплексного числа.

Має кілька конструкторів: за замовчуванням (0 + 0i), з передачею значень для обох частин, і конструктор копії.

Реалізує методи для арифметичних операцій: додавання, віднімання, множення та ділення (з перевіркою на ділення на нуль).

Перевизначений оператор для кожної з операцій (+, -, \*, /).

Метод ToString() для відображення комплексного числа у форматі (a + bi).

Функція AddVectors:

Приймає два списки комплексних чисел (вектори), перевіряє, чи мають вони однакову довжину, і додає відповідні елементи обох векторів.

Основна програма:

Створює два вектори випадкових комплексних чисел (довжина векторів — 2).

Виводить вектори та результат їх додавання на екран.

Код програми

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Task\_3;

namespace Task\_3

{

class Complex

{

public double r, i;

public Complex()

{

this.r = 0.0;

this.i = 0.0;

}

public Complex(double real, double imag)

{

this.r = real;

this.i = imag;

}

public static Complex Sum(Complex a, Complex b)

{

Complex res = new Complex();

res.r = a.r + b.r;

res.i = a.i + b.i;

return res;

}

public static Complex Multiplication(Complex a, Complex b)

{

Complex res = new Complex();

res.r = a.r \* b.r - a.i \* b.i;

res.i = a.i \* b.r + a.r \* b.i;

return res;

}

public static Complex Subtract(Complex a, Complex b)

{

Complex res = new Complex();

res.r = a.r - b.r;

res.i = a.i - b.i;

return res;

}

public static Complex Division(Complex a, Complex b)

{

Complex res = new Complex();

double denominator = b.r \* b.r + b.i \* b.i;

if (denominator == 0)

{

throw new DivideByZeroException("Ділення на 0");

}

res.r = (a.r \* b.r + a.i \* b.i) / denominator;

res.i = (a.i \* b.r - a.r \* b.i) / denominator;

return res;

}

public static Complex operator +(Complex a, Complex b)

{

return Complex.Sum(a, b);

}

public static Complex operator -(Complex a, Complex b)

{

return Complex.Subtract(a, b);

}

public static Complex operator \*(Complex a, Complex b)

{

return Complex.Multiplication(a, b);

}

public static Complex operator /(Complex a, Complex b)

{

return Complex.Division(a, b);

}

public Complex Clone()

{

return new Complex { r = this.r, i = this.i };

}

public Complex(Complex other)

{

this.r = other.r;

this.i = other.i;

}

public override string ToString()

{

return $"({r:F3} + {i:F3}i)";

}

}

internal class Program

{

// Функція для додавання двох векторів комплексних чисел

static List<Complex> AddVectors(List<Complex> vector1, List<Complex> vector2)

{

if (vector1.Count != vector2.Count)

{

throw new ArgumentException("Вектори мають різну довжину");

}

List<Complex> result = new List<Complex>();

for (int i = 0; i < vector1.Count; i++)

{

result.Add(vector1[i] + vector2[i]);

}

return result;

}

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.InputEncoding = System.Text.Encoding.Unicode;

Random random = new Random();

// Створення двох векторів з випадковими комплексними числами

int vectorSize = 2;

List<Complex> vector1 = new List<Complex>();

List<Complex> vector2 = new List<Complex>();

for (int i = 0; i < vectorSize; i++)

{

vector1.Add(new Complex(random.NextDouble() \* 10, random.NextDouble() \* 10));

vector2.Add(new Complex(random.NextDouble() \* 10, random.NextDouble() \* 10));

}

Console.WriteLine("Вектор 1:");

vector1.ForEach(c => Console.WriteLine(c));

Console.WriteLine("\nВектор 2:");

vector2.ForEach(c => Console.WriteLine(c));

// Додавання векторів

var resultVector = AddVectors(vector1, vector2);

Console.WriteLine("\nРезультат додавання векторів:");

resultVector.ForEach(c => Console.WriteLine(c));

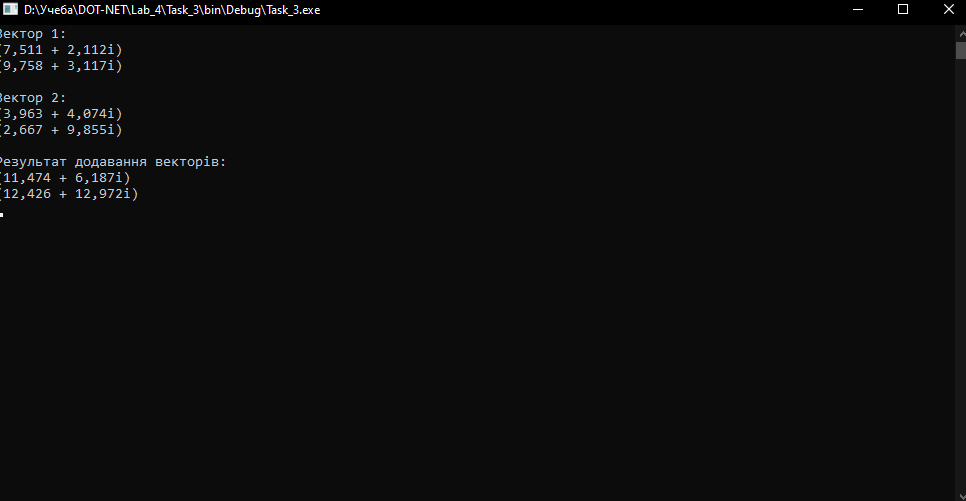
Console.ReadLine();

}

}

}

Виконання



Завдання 4.1

Для класу першого згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини **агрегації**. Продемонструвати роботу відносин, створивши декілька об’єктів

відповідних класів та позбавляючи чи додаючи властивості до цих об’єктів.

Короткий опис роботи:

Клас Faculty буде головним класом і міститиме колекцію студентів.

Клас Student залишатиметься підпорядкованим класом.

Клас Faculty буде мати можливість додавати або видаляти студентів, демонструючи агрегацію.

Клас Student: містить інформацію про студентів (ім'я, прізвище, рік народження) і метод Show(), щоб вивести інформацію про студента.

Клас Faculty: містить ім'я факультету та список студентів, яких факультет агрегує. Методи AddStudent і RemoveStudent дозволяють додавати або видаляти студентів із факультету. Метод ShowFacultyStudents виводить студентів, що належать до факультету.

В Main: створюються факультет і студенти, додаються студенти до факультету, після чого демонструється робота з агрегацією через додавання і видалення студентів із факультету.

Агрегація полягає в тому, що факультет (головний клас) має студентів (підпорядковані класи), але студенти можуть існувати незалежно від факультету.

Студент може бути доданий до одного факультету і видалений з нього, демонструючи відносини агрегації, де факультет не створює студентів, але управляє їхнім списком.

Код програми:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task\_4.\_1

{

class Student

{

public string LastName { get; set; }

public string FirstName { get; set; }

public int BirthDate { get; set; }

public Student(string lastName, string firstName, int birthDate)

{

LastName = lastName;

FirstName = firstName;

BirthDate = birthDate;

}

public void Show()

{

Console.WriteLine($"{LastName} {FirstName}, {BirthDate}");

}

}

class Faculty

{

public string Name { get; set; }

private List<Student> students;

// Конструктор факультету

public Faculty(string name)

{

Name = name;

students = new List<Student>();

}

// Додавання студента до факультету

public void AddStudent(Student student)

{

students.Add(student);

Console.WriteLine($"Студент {student.FirstName} {student.LastName} доданий до факультету {Name}.");

}

// Видалення студента з факультету

public void RemoveStudent(Student student)

{

if (students.Contains(student))

{

students.Remove(student);

Console.WriteLine($"Студент {student.FirstName} {student.LastName} видалений з факультету {Name}.");

}

else

{

Console.WriteLine("Студент не знайдений на цьому факультеті.");

}

}

// Виведення всіх студентів факультету

public void ShowFacultyStudents()

{

Console.WriteLine($"\nСтуденти факультету {Name}:");

if (students.Count > 0)

{

students.ForEach(s => s.Show());

}

else

{

Console.WriteLine("Немає студентів на цьому факультеті.");

}

}

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.InputEncoding = System.Text.Encoding.Unicode;

// Створюємо факультет

Faculty CSFaculty = new Faculty("КН");

// Створюємо студентів

Student student1 = new Student("Іванов", "Іван", 2000);

Student student2 = new Student("Петренко", "Петро", 2001);

Student student3 = new Student("Коваленко", "Іван", 2003);

Student student4 = new Student("Сидоров", "Олександр", 2002);

Student student5 = new Student("Шевченко", "Тарас", 1999);

Student student6 = new Student("Бондаренко", "Катерина", 2004);

// Додаємо студентів на факультет

CSFaculty.AddStudent(student1);

CSFaculty.AddStudent(student2);

CSFaculty.AddStudent(student3);

CSFaculty.AddStudent(student4);

CSFaculty.AddStudent(student5);

CSFaculty.AddStudent(student6);

// Виводимо студентів факультету

CSFaculty.ShowFacultyStudents();

// Видаляємо двох студентів

CSFaculty.RemoveStudent(student1);

CSFaculty.RemoveStudent(student3);

// Виводимо студентів після видалення

CSFaculty.ShowFacultyStudents();

Console.ReadLine();

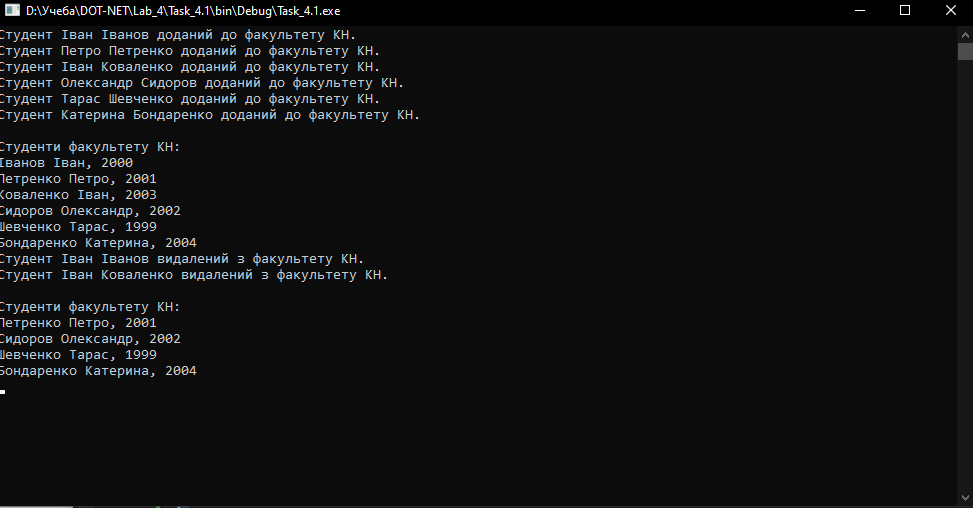
}

}

}

}

Виконання:



Завдання 4.2

Для класу першого завдання згідно до свого варіанту створити головний чи підпорядкований клас із яким організувати відносини **композиції**. Продемонструвати роботу відносин, створивши декілька об’єктів відповідних класів та позбавляючи чи додаючи властивості до цих об’єктів

Композиція

Головний клас

Студент

Підпорядкований клас

Оцінки з різних дисциплін  
Короткий опис роботи:

Grade: цей клас описує оцінку студента з певної дисципліни. Має два властивості: Subject (назва дисципліни) та GradeValue (сама оцінка). Крім того, є метод ShowGrade, який виводить інформацію про оцінку.

Student: цей клас описує студента з його особистими даними (ім'я, прізвище, рік народження) та списком оцінок (тип композиції). Методами цього класу є:

AddGrade, що дозволяє додавати оцінки для студента;

ShowGrades, що виводить всі оцінки студента;

Show, що виводить основну інформацію про студента (ім'я, прізвище та рік народження).

У методі Main створюються шість студентів, кожному з яких додаються оцінки по різних дисциплінах. Після цього інформація про кожного студента та їхні оцінки виводиться на екран.

Програма демонструє основи об'єктно-орієнтованого програмування в C#, зокрема композицію (студент має список оцінок) та інкапсуляцію (інформація про студента та оцінки знаходяться в окремих класах).

Код програми:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

namespace Task\_4.\_2

{

class Grade

{

public string Subject { get; set; }

public int GradeValue { get; set; }

// Конструктор

public Grade(string subject, int gradeValue)

{

Subject = subject;

GradeValue = gradeValue;

}

// Метод для виведення оцінки

public void ShowGrade()

{

Console.WriteLine($"Дисципліна: {Subject}, Оцінка: {GradeValue}");

}

}

class Student

{

private string firstName;

private string lastName;

private int birthDate;

// Список оцінок (композиція)

private List<Grade> grades;

public Student(string firstName, string lastName, int birthDate)

{

this.firstName = firstName;

this.lastName = lastName;

this.birthDate = birthDate;

grades = new List<Grade>(); // ініціалізація списку оцінок

}

// Методи для роботи з оцінками

public void AddGrade(string subject, int grade)

{

grades.Add(new Grade(subject, grade));

}

public void ShowGrades()

{

Console.WriteLine("\nОцінки студента:");

foreach (var grade in grades)

{

grade.ShowGrade();

}

}

public void Show()

{

Console.WriteLine($"{lastName} {firstName}, {birthDate}");

}

}

internal class Program

{

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.Unicode;

Console.InputEncoding = System.Text.Encoding.Unicode;

Student student1 = new Student("Іванов", "Іван", 2000);

Student student2 = new Student("Петренко", "Петро", 2001);

Student student3 = new Student("Коваленко", "Іван", 2003);

Student student4 = new Student("Сидоров", "Олександр", 2002);

Student student5 = new Student("Шевченко", "Тарас", 1999);

Student student6 = new Student("Бондаренко", "Катерина", 2004);

// Додавання оцінок

student1.AddGrade("Математика", 5);

student1.AddGrade("Фізика", 4);

student2.AddGrade("Математика", 3);

student2.AddGrade("Біологія", 4);

student3.AddGrade("Математика", 5);

student3.AddGrade("Фізика", 4);

student4.AddGrade("Фізика", 3);

student4.AddGrade("Інформатика", 4);

student5.AddGrade("Хімія", 5);

student5.AddGrade("Біологія", 5);

student6.AddGrade("Інформатика", 4);

student6.AddGrade("Математика", 3);

// Виведення інформації про студентів і їх оцінки

student1.Show();

student1.ShowGrades();

Console.WriteLine();

student2.Show();

student2.ShowGrades();

Console.WriteLine();

student3.Show();

student3.ShowGrades();

Console.WriteLine();

student4.Show();

student4.ShowGrades();

Console.WriteLine();

student5.Show();

student5.ShowGrades();

Console.WriteLine();

student6.Show();

student6.ShowGrades();

Console.WriteLine();

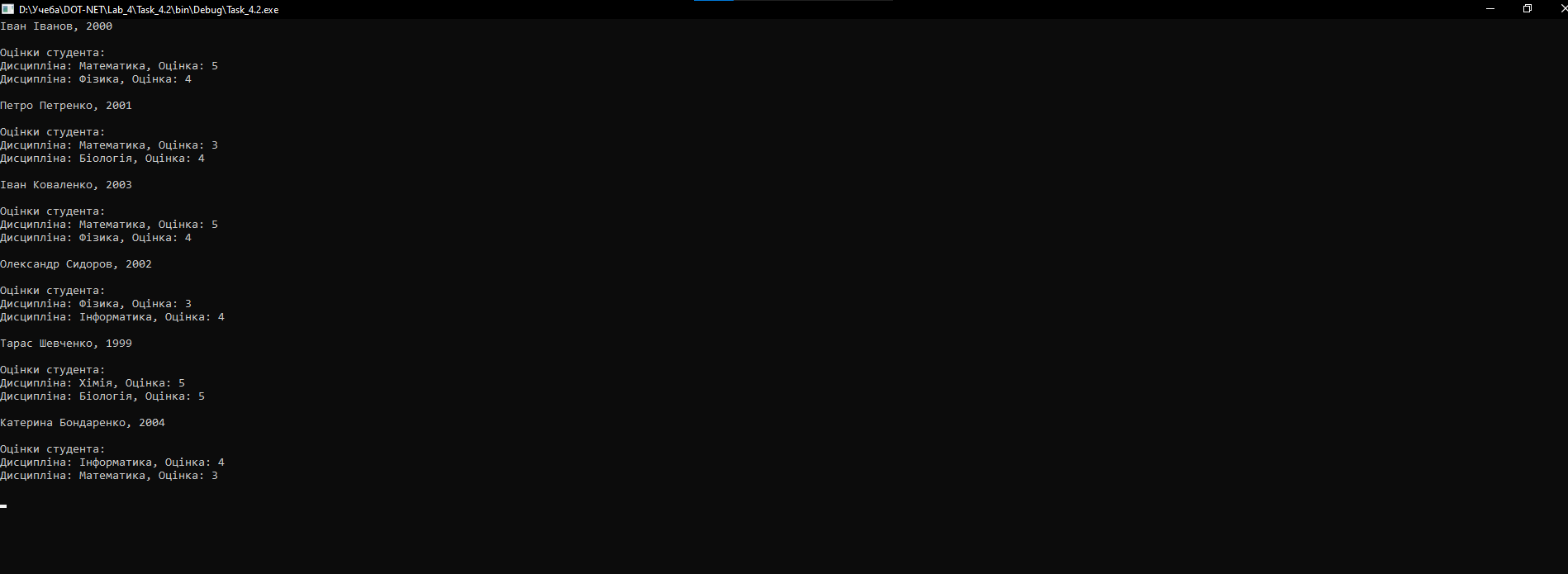
Console.ReadLine();

}

}

}

Виконання:



Висновок: набули навички в написанні програм із застосуванням принципів та відносин об’єктно-орієнтованого програмування (ООП).