

КПІ ім. Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Звіт до комп'ютерного практикуму з курсу
“Основи програмування. Частина 2”

Прийняв
Ст. викладач кафедри ІІІ
Вітковська І.І.
“13” травня 2025 р.

Виконав
Студент групи ІІІ-41
Козлов М.М.

Київ 2025

Комп'ютерний практикум №8

Тема: Відношення між класами та об'єктами

Мета – дослідити типи відношень між класами та об'єктами в ООП, навчитися проектувати об'єктно-орієнтовану модель предметної галузі.

Завдання:

1. Вивчити типи відношень між класами в ООП.
2. Спроекувати об'єктно-орієнтовану модель предметної галузі згідно з варіантом, визначивши необхідні для цього класи та їх структуру.
3. Вимоги до проектування:
 - розробити не менше 6 типів даних;
 - застосувати всі базові принципи ООП;
 - застосувати всі види відношень;
 - застосувати обробку виключень, де це є необхідним;
 - дотримуватись єдиної конвенції найменувань та принципів написання "чистого" коду;
 - код дозволяється коментувати лише xml-коментарями.
4. Написати програму, в якій реалізувати попередньо спроектовану об'єктно-орієнтовану модель.
5. Програмний інтерфейс, наприклад, введення\виведення з консолі, реалізовувати окремим проектом. Код програмного інтерфейсу має бути простим (демонструється використання класів предметної галузі шляхом створення об'єктів та їх застосування, відсутня перевірка коректності вводу, введення з консолі мінімальне або відсутнє взагалі).

Завдання за варіантом:

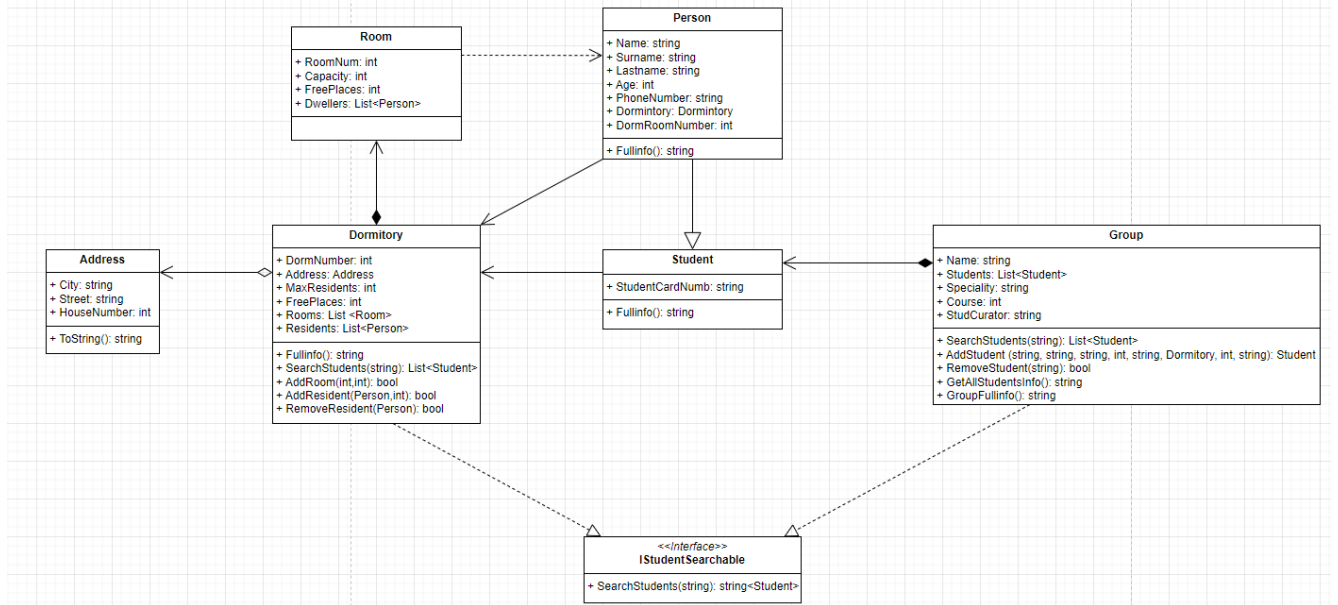
ЕЛЕКТРОННИЙ ДЕКАНАТ: ОБЛІК СТУДЕНТІВ НАВЧАЛЬНОГО
ЗАКЛАДУ

Функціональні вимоги до програмного забезпечення

1. Управління студентами
 - 1.1. Можливість додавати студента
 - 1.2. Можливість видаляти студента
 - 1.3. Можливість змінити дані студента
 - 1.4. Можливість перегляду списку всіх студентів
 - 1.5. Можливість перегляду даних вказаного студента
2. Управління групами
 - 2.1. Можливість додавати групу
 - 2.2. Можливість видаляти групу
 - 2.3. Можливість змінювати дані групи
 - 2.4. Можливість перегляду списку даних групи
 - 2.5. Можливість перегляду даних певної групи
 - 2.6. Можливість додавання студента до існуючої групи
 - 2.7. Можливість видалення студента з існуючої групи
3. Управління поселенням у гуртожиток
 - 3.1. Можливість додавання даних про гуртожиток (номери кімнат, максимальна кількість мешканців тощо)
 - 3.2. Можливість змінення даних про гуртожиток
 - 3.3. Можливість поселення студента у гуртожиток
 - 3.4. Можливість виписки студента з гуртожитку
 - 3.5. Можливість отримання інформації про проживаючих загалом, по кімнатах, вільні місця.
4. Пошук
 - 4.1. Можливість пошуку студента за його даними (прізвище, ім'я)
 - 4.2. Можливість пошуку студентів певної групи
 - 4.3. Можливість пошуку студентів у гуртожитку

Об'єктно-орієнтована модель

Діаграма класів:



Програмна реалізація

Клас Address:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace myClassLibrary
{
    public class Address
    {
        public string City { get; set; }
        public string Street { get; set; }
        public int HouseNumber { get; set; }

        public Address(string city, string street, int houseNumber)
        {
            City = city;
            Street = street;
            HouseNumber = houseNumber;
        }

        public override string ToString()
        {
            return $"{City}, {Street} {HouseNumber}";
        }
    }
}
```

Клас Dormitory:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Linq;

namespace myClassLibrary
{
    public class Dormitory : ISearchable
    {
        public int DormNumber { get; set; }
        public Address Address { get; set; }

        public int MaxResidents => Rooms.Sum(room => room.Capacity);
        public int FreePlaces => Rooms.Sum(room => room.FreePlaces);
        public List<Room> Rooms { get; set; } = new List<Room>();
        public List<Person> Residents { get; set; } = new List<Person>();

        public Dormitory(int dormnumber, Address address)
        {
            DormNumber = dormnumber;
            Address = address;
        }

        public string Fullinfo()
        {
            return $"Dormitory {DormNumber}, Address: {Address}, Rooms: {Rooms.Count}, Residents: {Residents.Count}";
        }
    }
}
```

```

public List<Student> SearchStudents(string keyword)
{
    return Residents
        .OfType<Student>() /// Фільтрує тільки Student
        .Where(s =>
            s.Name.ToLower().Contains(keyword.ToLower()) ||
            s.Surname.ToLower().Contains(keyword.ToLower()) ||
            s.PhoneNumber.ToLower().Contains(keyword.ToLower()) ||
            s.StudentCardNum.ToLower().Contains(keyword.ToLower())
        ).ToList();
}

```

```

public bool AddRoom(int NewRoomNumber, int capacity)
{
    var newRoom = new Room(NewRoomNumber, capacity);
    foreach (var room in Rooms)
    {
        if (room.RoomNum==NewRoomNumber)
        {
            Console.WriteLine("Кімната з таким номером вже існує");
            return false;
        }
    }
    Rooms.Add(newRoom);
    return true;
}

public bool AddResident(Person person, int? dormRoomNumber)
{
    if (!Residents.Contains(person))

```

```

    {
        Residents.Add(person);
        foreach (var room in Rooms)
        {
            if (room.RoomNum == dormRoomNumber && !room.Dwellers.Contains(person) &&
room.FreePlaces>=1)
            {
                room.Dwellers.Add(person);
                person.Dormintory = this;
                person.DormRoomNumber = dormRoomNumber;

                ///Console.WriteLine($"Резидент {person.Surname} {person.Name} успішно доданий
до кімнати {room.RoomNum}");
                return true;
            }
        }
        throw new InvalidOperationException ("Такої кімнати не існує, або у ній немає вільним
місць");
    }
    else
    {
        throw new InvalidOperationException("Ця особа вже проживає в гуртожитку");
    }
}

```

```

public bool RemoveResident(Person person)

```

```

{
    if (Residents.Contains(person))
    {
        Residents.Remove(person);
        foreach (var room in Rooms)
        {
            if (room.Dwellers.Contains(person))

```



```

    {
        room.Dwellers.Remove(person);
        person.Dormintory = null;
        person.DormRoomNumber = null;

        ///Console.WriteLine($"Резидент {person.Surname} {person.Name} успішно
виселений з гуртожитку ");
        return true;
    }
}

/// Якщо не знайдено в жодній кімнаті, все одно очищаємо дані
person.Dormintory = null;
person.DormRoomNumber = null;

throw new InvalidOperationException($"Резидент {person.Surname} {person.Name}
видалений зі списку мешканців, але не був знайдений у жодній кімнаті.");

///return true;

}
else
{
    throw new InvalidOperationException("Ця особа не проживає в гуртожитку");
    ///return false;
}
}
}
}
}

```

Клас Group:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Globalization;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace myClassLibrary
```

```
{
```

```
    public class Group : ISearchable
```

```
    {
```

```
        public string Name { get; set; }
```

```
        public List<Student> Students { get; set; }
```

```
        public string Speciality { get; set; }
```

```
        public int Course { get; set; }
```

```
        public string StudCurator { get; set; }
```

```
        public List<Student> SearchStudents(string keyword)
```

```
        {
```

```
            keyword = keyword.ToLowerInvariant();
```

```
            return Students.Where(s =>
```

```
                s.Name.ToLower().Contains(keyword) ||
```

```
                s.Surname.ToLower().Contains(keyword)
```

```
            ).ToList();
```

```
        }
```

```
        public Student AddStudent(string name, string surname, string lastname, int age, string  
        phoneNumber,
```

```
        Dormitory dormNumber, int? dormRoomNumber, string studentCardNum)
```

```

{

    if (Students.Any(s => s.StudentCardNum == studentCardNum))
    {
        throw new InvalidOperationException("Студент з таким номером білета вже існує.");
    }

    var student = new Student(name, surname, lastname, age, phoneNumber, dormNumber,
dormRoomNumber, studentCardNum);
    Students.Add(student);
    return student;
}

public Group(string name, string speciality, int course, string studCurator)
{
    Name = name;
    Speciality = speciality;
    Course = course;
    StudCurator = studCurator;
    Students = new List<Student>();
}

public bool RemoveStudent(string studentCardNum)
{
    var student = Students.FirstOrDefault(s => s.StudentCardNum == studentCardNum);
    if (student != null)
    {
        Students.Remove(student);
        student.Dormintory?.RemoveResident(student);
        Console.WriteLine($"Студента {student.Surname} {student.Name} успішно видалено.");
        return true;
    }
    else

```

```

    {
        throw new InvalidOperationException("Студента не найдено.");
        ///return false;
    }
}

public string GetAllStudentsInfo()
{
    if (Students.Count == 0)
        return "There are no students in this group.";

    var sb = new StringBuilder();
    sb.AppendLine($"Students of Group {Name}:");

    foreach (var student in Students)
    {
        sb.AppendLine(student.Fullinfo());
    }

    return sb.ToString();
}

public string GroupFullinfo()
{
    return $"Group {Name}, {Speciality}, Course {Course}, Curator: {StudCurator}, Students:
{Students.Count}";
}

}
}

```

Интерфейс ISearchable:

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Text;
```

```
using System.Threading.Tasks;
```

```
namespace myClassLibrary
```

```
{
```

```
    public interface ISearchable
```

```
    {
```

```
        List<Student> SearchStudents(string keyword);
```

```
    }
```

```
}
```

Клас Person:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace myClassLibrary
{
    public class Person
    {
        public string Name { get; set; }
        public string Surname { get; set; }
        public string Lastname { get; set; }
        public int Age { get; set; }
        public string PhoneNumber { get; set; }
        public Dormitory Dormintory { get; set; }
        public int? DormRoomNumber { get; set; }

        public Person(string name, string surname, string lastname, int age, string phoneNumber,
Dormitory dormNumber, int? dormRoomNumb)
        {
            Name = name;
            Surname = surname;
            Lastname = lastname;
            Age = age;
            PhoneNumber = phoneNumber;
            ///bool roomFound = false;
            if (dormNumber != null)
            {
                if (dormNumber.AddResident(this, dormRoomNumb))
                {
```

```

        Dormintory = dormNumber;
        DormRoomNumber = dormRoomNumb;
    }
    else
    {
        throw new InvalidOperationException("Кімната не знайдена або вже зайнята.");
    }
}
}

```

```

public virtual string Fullinfo()

```

```

{
    string namePart = $"{Surname} {Name} {Lastname}".PadRight(40);
    string agePart = $"Age: {Age}".PadRight(10);
    string phonePart = $"Phone number: {PhoneNumber}".PadRight(30);

    if (Dormintory != null && DormRoomNumber != null)
    {
        string dormPart = $"Dorm: {Dormintory.DormNumber}".PadRight(15);
        string roomPart = $"DormNumber: {DormRoomNumber}".PadRight(15);
        return $"{namePart} {agePart} {phonePart} {dormPart} {roomPart}";
    }
    else
    {
        return $"{namePart} {agePart} {phonePart}";
    }
}

```

```

}

```

```

}

```

Клас Room:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace myClassLibrary
{
    public class Room
    {
        public int RoomNum { get; set; }
        public int Capacity { get; set; }
        public int FreePlaces => Capacity - Dwellers.Count;
        public List<Person> Dwellers { get; set; }

        internal Room(int roomNum, int capacity)
        {
            RoomNum = roomNum;
            Capacity = capacity;
            Dwellers = new List<Person>();
        }
    }
}
```


Клас Student:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using myClassLibrary;

namespace myClassLibrary
{
    public class Student : Person
    {
        public string StudentCardNum { get; set; }

        internal Student(string name, string surname, string lastname, int age, string phoneNumber,
            Dormitory dormNumber, int? dormRoomNumb, string studentCardNum)
            : base(name, surname, lastname, age, phoneNumber, dormNumber, dormRoomNumb)
        {
            ///Group = group ?? throw new ArgumentNullException(nameof(group));
            StudentCardNum = studentCardNum;
        }

        public override string Fullinfo()
        {
            string namePart = $"{Surname} {Name} {Lastname}".PadRight(40);
            string agePart = $"Age: {Age}".PadRight(10);
            string phonePart = $"Phone number: {PhoneNumber}".PadRight(30);
            string cardPart = $"Student Card: {StudentCardNum}".PadRight(30);

            string baseInfo = $"{namePart} {agePart} {phonePart} {cardPart}";

            if (Dormintory != null && DormRoomNumber != null)
```

```
{
    string dormPart = $"Dorm: {Dormintory.DormNumber}".PadRight(15);
    string roomPart = $"DormNumber: {DormRoomNumber}".PadRight(15);
    return $" {baseInfo} {dormPart} {roomPart} ";
}
else
{
    return baseInfo;
}
}

}
}
```

Код з демонстрацією використання моделі:

Приклад роботи

```
Microsoft Visual Studio Debug Console

Creating 3 groups:
Group IP-41, Software Engineering, Course 1, Curator: Juliy Polupan, Students: 0
Group IP-42, Software Engineering, Course 1, Curator: Oleh Lisovichenko, Students: 0
Group IP-45, Software Engineering, Course 1, Curator: Volodymyr Clichko, Students: 0

Deleting group IP-45::
Groups after deletion:
Group IP-41, Software Engineering, Course 1, Curator: Juliy Polupan, Students: 0
Group IP-42, Software Engineering, Course 1, Curator: Oleh Lisovichenko, Students: 0
Group IP-41, Software Engineering, Course 2, Curator: Oleh Lisovichenko, Students: 0

Adding 5 students to the first group:

Students of Group IP-41:
Sokurenko Bohdan Vitaliiiovych      Age: 19  Phone number: +380631545159  Student Card: 1R2C3485V6
Ponomarenko Mykola Oleksandrovych    Age: 18  Phone number: +380683449158  Student Card: 3T7K29M1QX      Dorm: 18      DormNumber: 212
Cherednichenko Artem Olehovych       Age: 17  Phone number: +380983453037  Student Card: 9P5D43L8ZN      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY

Студента Cherednichenko Artem успішно видалено.
Students of Group IP-41:
Sokurenko Bohdan Vitaliiiovych      Age: 19  Phone number: +380660544000  Student Card: 1R2C3485V6
Ponomarenko Mykola Oleksandrovych    Age: 18  Phone number: +380683449158  Student Card: 3T7K29M1QX      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY

Students of Group IP-42:
Lazarev Vitalii Vadymovych          Age: 18  Phone number: +380996036304  Student Card: B9R7C2VMY8
Serhiienko Taras Andriiovych         Age: 18  Phone number: +380671583065  Student Card: C8V2B1R9Y6

Dormitory residents data:

Free places in the dormitory: 5
Free places in room212: 2
Free places in room1: 3

Enter student's first or last name to search:
ko
Sokurenko Bohdan Vitaliiiovych      Age: 19  Phone number: +380660544000  Student Card: 1R2C3485V6
Ponomarenko Mykola Oleksandrovych    Age: 18  Phone number: +380683449158  Student Card: 3T7K29M1QX      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR      Dorm: 18      DormNumber: 212
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY
Serhiienko Taras Andriiovych         Age: 18  Phone number: +380671583065  Student Card: C8V2B1R9Y6

Searching for students in group IP-41:
Sokurenko Bohdan Vitaliiiovych      Age: 19  Phone number: +380660544000  Student Card: 1R2C3485V6
Ponomarenko Mykola Oleksandrovych    Age: 18  Phone number: +380683449158  Student Card: 3T7K29M1QX

Searching for students in dormitory:
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR      Dorm: 18      DormNumber: 212

C:\Users\user\source\repos\OOP8\OOP8\bin\Debug\OOP8.exe (process 11744) exited with code 0 (0x0).
Press any key to close this window . . .

Microsoft Visual Studio Debug Console

Cherednichenko Artem Olehovych       Age: 17  Phone number: +380983453037  Student Card: 9P5D43L8ZN      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY

Студента Cherednichenko Artem успішно видалено.
Students of Group IP-41:
Sokurenko Bohdan Vitaliiiovych      Age: 19  Phone number: +380660544000  Student Card: 1R2C3485V6
Ponomarenko Mykola Oleksandrovych    Age: 18  Phone number: +380683449158  Student Card: 3T7K29M1QX      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY

Students of Group IP-42:
Lazarev Vitalii Vadymovych          Age: 18  Phone number: +380996036304  Student Card: B9R7C2VMY8
Serhiienko Taras Andriiovych         Age: 18  Phone number: +380671583065  Student Card: C8V2B1R9Y6

Dormitory residents data:

Free places in the dormitory: 5
Free places in room212: 2
Free places in room1: 3

Enter student's first or last name to search:
kozlov
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY      Dorm: 18      DormNumber: 212

Searching for students in group IP-41:
Sokurenko Bohdan Vitaliiiovych      Age: 19  Phone number: +380660544000  Student Card: 1R2C3485V6
Ponomarenko Mykola Oleksandrovych    Age: 18  Phone number: +380683449158  Student Card: 3T7K29M1QX

Searching for students in dormitory:
Kozlov Mykyta Maksymovych            Age: 18  Phone number: +380667992569  Student Card: 2B9R76F3WY      Dorm: 18      DormNumber: 212
Kolomiets Maksym Oleksiiovych        Age: 18  Phone number: +380660544379  Student Card: 6A1C82V4JR      Dorm: 18      DormNumber: 212

C:\Users\user\source\repos\OOP8\OOP8\bin\Debug\OOP8.exe (process 8080) exited with code 0 (0x0).
Press any key to close this window . . .
```