Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

# КУРСОВА РОБОТА

з дисципліни «Інженерія програмного забезпечення»

на тему «Електронний деканат: облік студентів навчального закладу»

Виконав:

Маковецький Д.О.

Київ – 2024

# АНОТАЦІЯ

Основні відомості:

Кількість ілюстрацій – 4
Кількість додатків – 6
Кількість використаних джерел – 4

Ключові слова: технології, програмне забезпечення.

Текст реферату:

У даній курсовій роботі представлено основні аспекти розробки програмного забезпечення на тему "Електронний деканат: облік студентів навчального закладу". Метою курсової роботи є створення електронного деканату, який дає змогу вести облік студентів на електронних пристроях. Також пояснюється використання різних технологій та методів у цьому процесі . Використання сучасних технологій є дуже важливим процесом для адаптації до швидких змін у сучасному світі.

# ЗМІСТ

# ВСТУП

На сьогодні існує багато способів збереження інформації, як застарілі, так і більш сучасні. До застарілих можна віднести усі фізичні носії даних, такі як папір, книжки тощо. Такі носії легко втрачати та вони мають властивість реагувати на навколишнє середовище.

Метою цієї курсової роботи є створення електронного деканату, який дає змогу вести облік студентів навчального закладу зі збереженням даних у локальних файлах. Він робить можливим перейти з фізичних носіїв інформації на цифрові. Такі дані можна зберігати у хмарному сховищі, де вони знаходяться у безпеці.

# Розділ 1 - Огляд MVC. Постановка задачі

## 1.1 Огляд MVC

MVC(Модель–представлення–контролер) - це архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення. Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

- Модель є центральним компонентом шаблону MVC і відображає поведінку застосунку, незалежну від інтерфейсу користувача. Модель стосується прямого керування даними, логікою та правилами застосунку.
- Вигляд може являти собою будь-яке представлення інформації, одержуване на виході, наприклад графік чи діаграму. Одночасно можуть співіснувати кілька виглядів (представлень) однієї й тієї ж інформації, наприклад гістограма для керівництва компанії й таблиці для бухгалтерії.
- Контролер одержує вхідні дані й перетворює їх на команди для моделі чи вигляду.

## 1.2 Постановка задачі

Метою курсової роботи є створення програмного забезпечення для електронного деканату. Основні завдання:

- Створення графічного інтерфейсу користувача (GUI)
- Реалізація зберігання та обробки даних
- Реалізація основних функцій програми

Функціональні вимоги до програмного забезпечення:

1. Управління студентами
    1.1. Можливість додавати студента
    1.2. Можливість видаляти студента
    1.3. Можливість змінити даних 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

# Розділ 2 - Проектування ПЗ

## 2.1 Мова програмування

Для виконання цієї курсової роботи я використав мову програмування Python. Мій вибір був зумовлений кількома важливими факторами. По-перше, Python є потужною і гнучкою мовою програмування, яка підтримує об'єктно-орієнтований підхід та має багатий набір стандартних бібліотек, що дозволяє ефективно вирішувати широке коло задач. По-друге, Python забезпечує високу швидкість розробки завдяки простому і зрозумілому синтаксису, що дозволяє зосередитися на логіці програми, а не на деталях реалізації.

## 2.2 Збереження даних

Для збереження даних були створені декілька CSV файлів з роздільником у вигляді крапки з комою для студентів, груп, кімнат та гуртожитку.

## 2.3 Графічний інтерфейс користувача

Для створення графічного інтерфейсу користувача я обрав вбудовану бібліотеку Tkinter у Python. Ця бібліотека була обрана через декілька факторів. По-перше, tkinter є частиною стандартної бібліотеки Python, що означає, що вона доступна за замовчуванням при встановленні Python. По-друге, GUI, створені за допомогою tkinter, можуть працювати на різних платформах (Windows, macOS, Linux) без необхідності внесення змін у код.

## 2.4 Інтегроване середовище розробки

Для виконання цієї курсової роботи було обрано інтегроване середовище розробки PyCharm. PyCharm надає широкий спектр функціональності, що полегшує роботу розробника, такі як автодоповнення коду, перевірка синтаксису, відладка, керування версіями за допомогою систем контролю версій, інтеграція з віртуальними середовищами Python тощо. Крім того, PyCharm має дружній інтерфейс, який допомагає зосередитися на розробці програм, забезпечуючи зручні інструменти для організації проектів та взаємодії з кодом.

## 2.5 Можливості користувача

Користувач зможе вибирати необхідний розділ(Управління студентами, Управління групами, Управління поселенням у гуртожиток, Пошук) у якому буде весь необхідний список функцій.

## Розділ 3 - Розробка ПЗ

### 3.1 Шаблони проектування

Для виконання курсової роботи було вирішено використати поведінковий шаблон проектування Observer.

Шаблон Observer – один із поведінкових паттернів проектування, який використовується для створення механізму підписки, в якому об'єкти, названі "спостерігачами" (observers), автоматично отримують оновлення від об'єкта, який називається "суб'єктом" (subject), коли той змінює свій стан. Головна мета – забезпечити спрощений та розширюваний спосіб взаємодії між об'єктами у системі, дозволяючи одному об'єкту повідомляти інші про свої зміни стану, при цьому зберігаючи низьку зв'язність між ними.

Його використання було зумовлено потребою видаляти студента з групи та кімнати, коли він видаляється з основного списку студентів.

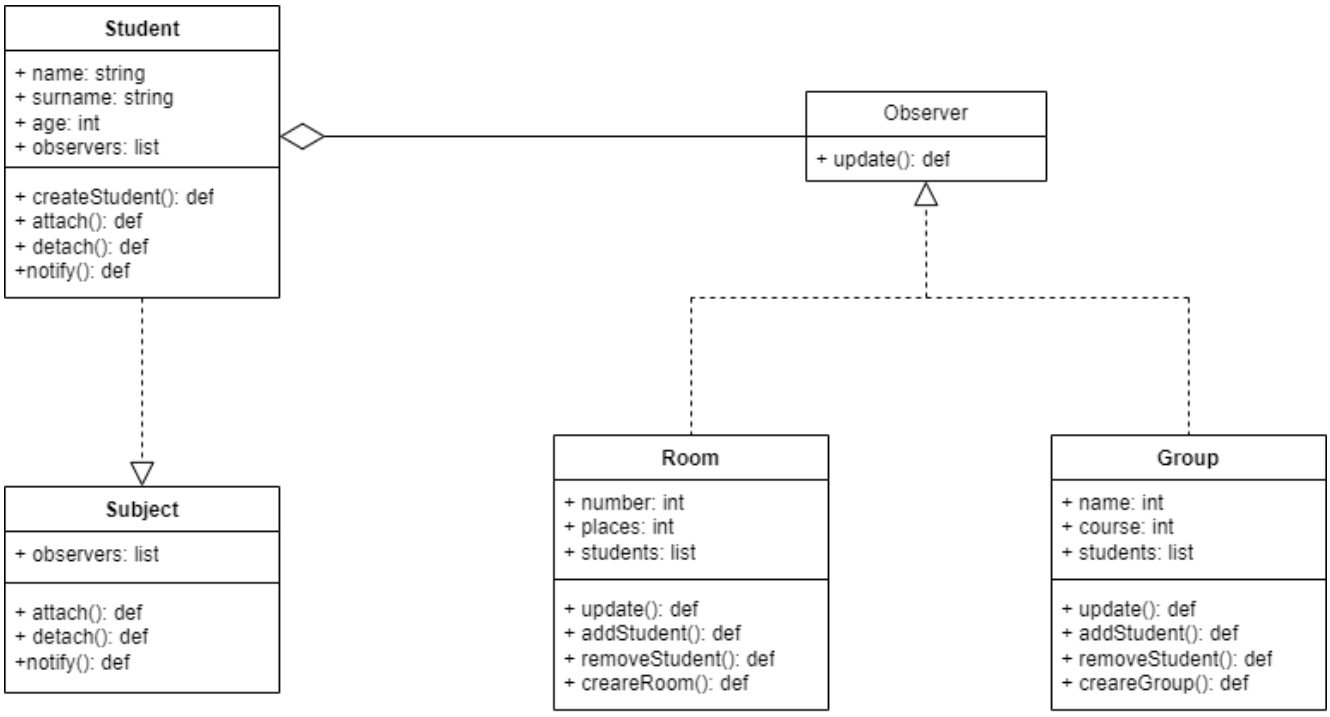


Рисунок 1 – діаграма класів шаблону Observer

### 3.2 Класи

- Observer – це абстрактний клас (далі буду називати його інтерфейс) спостерігача. Потрібен для використання поведінкового шаблону Observer.

Він має лише метод update(оновлення даних). Класи які реалізують цей інтерфейс є спостерігачами за класами який реалізує інтерфейс Subject.

- Subject – це абстрактний клас (далі буду називати його інтерфейс) об’єкту за яким спостерігають класи, які реалізують інтерфейс Observer. Потрібен для використання поведінкового шаблону Observer. Він має декілька методів: attach(додавання спостерігачів), detach(видалення спостерігачів), notify(повідомлення всіх спостерігачів).
- Student – клас студенту, який реалізує інтерфейс Subject. Має поля: ім’я, прізвище, вік, id та список спостерігачів; методи: createStudent(для ініціалізації полів) та всі методи інтерфейсу. Поле id є статичним унікальним ідентифікатором, тобто користувач не може його змінити.
- Group – клас групи, який реалізує інтерфейс Observer. Має поля: назва, курс та список студентів; методи: createGroup(для ініціалізації полів), addStudent(для додавання студента до групи), removeStudent(для видалення студента з групи) та всі методи інтерфейсу.
- Room – клас кімнати гуртожитку, який реалізує інтерфейс Observer. Має поля: номер, максимальна кількість студентів які можуть бути заселені та список студентів; методи: createRoom(для ініціалізації полів), addStudent(для поселення студента до кімнати), removeStudent(для виписки студента з кімнати) та всі методи інтерфейсу. Поле номер є статичним унікальним ідентифікатором, тобто користувач не може його змінити.
- Dormitory – клас гуртожитку. Має поля: максимальна кількість студентів можуть жити у гуртожитку, кількість кімнат та список кімнат; методи: addRoom(для додавання кімнат). Поле кількість кімнат є статичним, тобто користувач не може його змінити.

### 3.3 Збереження та завантаження даних

Як вже було зазначено у розділі 2, для збереження даних були створені декілька CSV файлів з роздільником у вигляді крапки з комою. При запуску програми, у файлах Student.py, Group.py, Room.py та Dormitory.py виконуються функції base(). Ці функції, залежно від файлу, починають створювати об’єкти відповідних класів звертаючись до відповідних CSV файлів, а у випадку з Student та Group ще створюються списки, куди записуються об’єкти цих класів. Створення об’єктів дає змогу не звертатися кожен раз до файлу, а звертатися до відповідних об’єктів.

Коли користувач вирішить закрити програму, то перед цим треба буде натиснути кнопку “Зберегти дані”. Це зроблено для того, щоб у випадку коли користувач, наприклад, випадково видалив користувача, то щоб у нього була можливість перезапустити програму та відновити дані.

Отже програма звертається до файлів при запуску та при закінченні роботи.

## Розділ 4 - Тестування ПЗ

При запуску програми користувач бачить перед собою розділ “Інформація” у якому вказується назва програми та кнопка для збереження даних.


Рисунок 2 – розділ “Інформація”

На вибір у користувача є декілька інших розділів: “Студенти”, “Групи”, “Гуртожиток”, “Пошук”. Кожен з них відповідає за виконання функцій, які пов’язані з кожним розділом. Наприклад розділ студент має функції:

- Новий студент
- Видалити студента
- Змінити дані студента
- Всі студенти

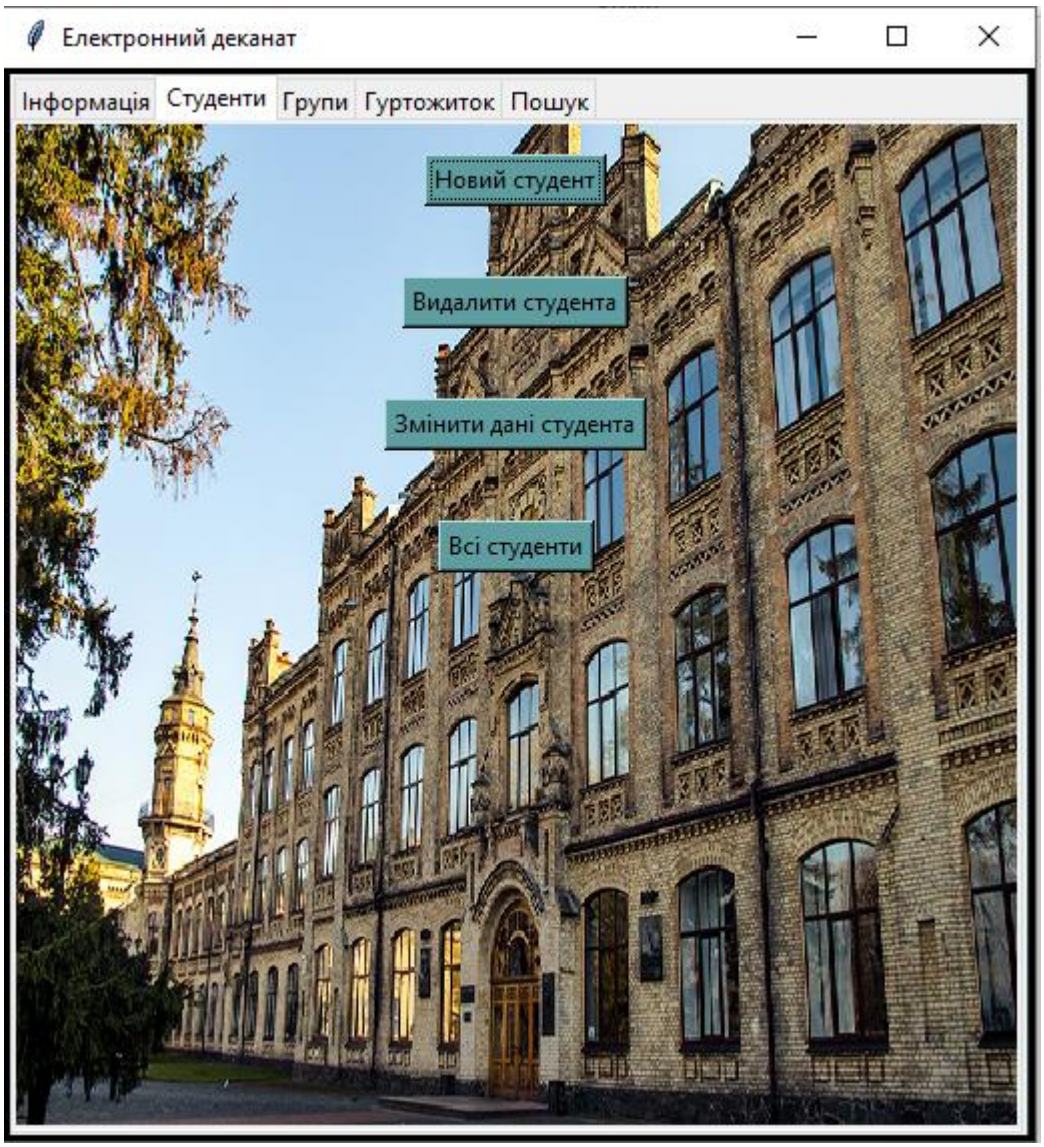


Рисунок 3 – розділ “Студенти”


Рисунок 4 – Окреме вікно для виконання функції “Зміна інформації про студента”

# ВИСНОВОК

При розробці програмного забезпечення “Електронний деканат: облік студентів навчального закладу” було створено зрозумілий інтерфейс користувача, можливість редагування та перегляду інформації про студентів, групи, гуртожиток. Було використано мову програмування Python, бібліотеку tkinter для створення користувацького інтерфейсу та інтегроване середовище розробки PyCharm. Також був використаний поведінковий шаблон проектування Observer.

# СПИСОК ВИКОРИСНИХ ДЖЕРЕЛ

Література:

Design Patterns: elements of reusable object-oriented software / Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Indianapolis: - Addison-Wesley, 1994. -417 p. ISBN: 0201633612

Tkinter:
https://docs.python.org/uk/3/library/tkinter.html
https://www.tcl.tk/man/tcl8.6/TkCmd/contents.htm

Python:
https://docs.python.org/3.9/

# ДОДАТКИ

Main.py:

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from PIL import Image, ImageTk
import Win_Student
import Win_Group
import Win_Dormitory
import Win_Search
import Student
import Group
import Dormitory

def save_data():
    students = Student.get_list_st()
    with open("students", 'w') as file:
        for i in students:
            n = i.name
            s = i.surname
            a = i.age
            id = i.id
            file.write(f'{n};{s};{a};{id};\n')
    file.close()
    groups = Group.get_list_gr()
    with open("groups", 'r+') as file:
        for i in groups:
            n = i.name
            c = i.course
            s = i.students
            id_list = []
            for i in s:
                id_list.append(str(i.id))
            id_list = ",".join(id_list)
            file.write(f'{n};{c};{id list};\n')
    file.close()
    dormitory = Dormitory.get_dormitory()
    with open("rooms", 'w') as file:
        ids = 0
        list_ids = []
        list_write = []
        for i in dormitory.rooms:
            if i.students is not [] and i != dormitory.rooms[len(dormitory.rooms)-
1]:
                n = i.number
                p = i.places
                s = i.students
                id_list = []
                for j in s:
                    id_list.append(str(j.id))
                    ids += 1
                list_write.append(f'{n};{p};{",".join(id_list)};\n')
                list_ids.append(','.join(id_list))
            elif i.students is not [] and i ==
dormitory.rooms[len(dormitory.rooms)-1]:
                n = i.number
                p = i.places
                s = i.students
                id_list = []
                for j in s:
```

```
                    id_list.append(str(j.id))
                    ids += 1
                list_write.append(f'{n};{p};{",".join(id_list)};\n')
                list_ids.append(','.join(id_list))
                if ids > dormitory.max_places:
                    messagebox.showerror("Помилка", "Переповення студентів у
гуртожитку")
                    break
                else:
                    for i in list_write:
                        file.write(i)
    file.close()
    with open("dormitory", 'w') as file:
        file.write(f'{dormitory.max_places}')

root = Tk()
root.title("Електронний деканат")
root.geometry("515x536")
root.config(bg='#2F4F4F', highlightcolor='black', highlightthickness=3)

notebook = ttk.Notebook(root)
notebook.grid(row=0, column=0, sticky="nsew")

image = Image.open('background.png')
image = image.resize((500, 500), Image.Resampling.NEAREST)
image = ImageTk.PhotoImage(image)

frame1 = Frame(notebook)
frame1.config(bg='#4169E1')
notebook.add(frame1, text='Інформація')
Label(frame1, image=image).grid(row=0, rowspan=21)
Label(frame1, text = 'Електронний деканат').grid(row=0, padx=10, pady=10)
Label(frame1, text = 'Перед закриттям програми натисніть кнопку "Зберегти
дані"').grid(row=1, padx=10, pady=10)
Button(frame1, text = 'Зберегти дані', command=save_data,
bg="#5F9EA0").grid(row=2, padx=10, pady=10)

frame2 = Frame(notebook)
frame2.config(bg='#4169E1')
notebook.add(frame2, text='Студенти')
Label(frame2, image=image).grid(row=0, rowspan=21)
Button(frame2, text="Новий студент", command=Win_Student.win_st_cr,
bg="#5F9EA0").grid(row=0, padx=10, pady=10)
Button(frame2, text="Видалити студента", command=Win_Student.win_st_dl,
bg="#5F9EA0").grid(row=1, padx=10, pady=10)
Button(frame2, text="Змінити дані студента", command=Win_Student.win_st_ch,
bg="#5F9EA0").grid(row=2, padx=10, pady=10)
Button(frame2, text="Всі студенти", command=Win_Student.win_show_st,
bg="#5F9EA0").grid(row=3, padx=10, pady=10)

frame3 = Frame(notebook)
frame3.config(bg='#4169E1')
notebook.add(frame3, text='Групи')
Label(frame3, image=image).grid(row=0, rowspan=21)
Button(frame3, text="Нова група", command=Win_Group.win_gr_cr,
bg="#5F9EA0").grid(row=0, padx=10, pady=10)
Button(frame3, text="Видалити групу", command=Win_Group.win_gr_dl,
bg="#5F9EA0").grid(row=1, padx=10, pady=10)
Button(frame3, text="Змінити дані групи", command=Win_Group.win_gr_ch,
bg="#5F9EA0").grid(row=2, padx=10, pady=10)
Button(frame3, text="Дані окремої групи", command=Win_Group.win_show_gr,
bg="#5F9EA0").grid(row=3, padx=10, pady=10)
Button(frame3, text="Дані всіх груп", command=Win_Group.win_show_grs,
```

```
bg="#5F9EA0").grid(row=4, padx=10, pady=10)
Button(frame3, text="Додавання/Видалення студентів", command=Win_Group.win_gr_st,
bg="#5F9EA0").grid(row=5, padx=10, pady=10)

frame4 = Frame(notebook)
frame4.config(bg='#4169E1')
notebook.add(frame4, text='Гуртожиток')
Label(frame4, image=image).grid(row=0, rowspan=21)
Button(frame4, text="Додавання/Видалення студентів",
command=Win_Dormitory.win_dm_st, bg="#5F9EA0").grid(row=0, padx=10, pady=10)
Button(frame4, text="Змінити дані кімнати", command=Win_Dormitory.win_ch_rm,
bg="#5F9EA0").grid(row=1, padx=10, pady=10)
Button(frame4, text="Змінити дані гурожитку", command=Win_Dormitory.win_ch_dm,
bg="#5F9EA0").grid(row=2, padx=10, pady=10)
Button(frame4, text="Подивитися дані про гуртожиток",
command=Win_Dormitory.win_sh_st, bg="#5F9EA0").grid(row=3, padx=10, pady=10)

frame5 = Frame(notebook)
frame5.config(bg='#4169E1')
notebook.add(frame5, text='Пошук')
Label(frame5, image=image).grid(row=0, rowspan=21)
Button(frame5, text="Пошук за ім'ям та прізвищем", command=Win_Search.win_sr_st,
bg="#5F9EA0").grid(row=0, padx=10, pady=10)
Button(frame5, text="Пошук у певній групі", command=Win_Search.win_sr_gr,
bg="#5F9EA0").grid(row=1, padx=10, pady=10)
Button(frame5, text="Пошук у гуртожитку", command=Win_Search.win_sr_dm,
bg="#5F9EA0").grid(row=2, padx=10, pady=10)

root.mainloop()
```

Student.py:

```
from Subject import Subject


class Student(Subject):
    def __init__(self, name="", surname="", age=0, id=0):
        super().__init__()
        self.name = name
        self.surname = surname
        self.age = age
        self.id = id
        self.observers = []

    def createStudent(self, name, surname, age, id):
        self.name = name
        self.surname = surname
        self.age = age
        self.id = id

    def attach(self, observer):
        self.observers.append(observer)

    def detach(self, observer):
        self.observers.remove(observer)

    def notify(self, operation):
        for observer in self.observers:
            observer.update(self, operation)
```

```
def base():
    global list_st
    list_st = []
    with open("students", 'r') as file:
        lines = file.readlines()
        for j in range(len(lines)):
            lines[j] = lines[j].split(";")
            student = Student()
            student.createStudent(lines[j][0], lines[j][1], lines[j][2],
lines[j][3])
            list_st.append(student)
    file.close()


base()


def get_list_st():
    return list_st


def findStudent(id):
    for i in list_st:
        if i.id == id:
            return i


def minId():
    id_list = []
    for i in list_st:
        id_list.append(int(i.id))
    max_id = max(id_list)
    for i in range(1, int(max_id) + 2):
        if i not in id_list:
            return i
```

Group.py:

```
import Student
from Observer import Observer


class Group(Observer):
    def __init__(self, name="", course=0):
        self.name = name
        self.course = course
        self.students = []

    def createGroup(self, name_g, course):
        self.name = name_g
        self.course = course

    def addStudent(self, student):
        self.students.append(student)
        student.attach(self)

    def removeStudent(self, student):
        self.students.remove(student)
        student.detach(self)

    def update(self, student, operation):
```

```
        if operation == "delete":
            for stud in self.students:
                if student.id == stud.id:
                    self.removeStudent(stud)

    def del_group(self):
        for i in self.students:
            i.detach(self)


def base():
    global list_gr
    list_gr = []
    with open("groups", 'r') as file:
        lines = file.readlines()
        for j in range(len(lines)):
            lines[j] = lines[j].split(";")
            group = Group()
            group.createGroup(lines[j][0], lines[j][1])
            students_id = lines[j][2].split(",")
            for k in students_id:
                student = Student.findStudent(k)
                if students_id != ['']:
                    group.addStudent(student)
            list_gr.append(group)
    file.close()


base()


def get_list_gr():
    return list_gr


def findGroup(name):
    for i in list_gr:
        if i.name == name:
            return i


def rep_check(student):
    for i in list_gr:
        for stud in i.students:
            if stud == student:
                return True
    return False
```

Room.py:

```
from Observer import Observer

class Room(Observer):
    def __init__(self, number=0, places=0):
        self.number = number
        self.places = places
        self.students = []

    def addStudent(self, student):
        self.students.append(student)
        student.attach(self)
```

```
    def createRoom(self, number, places):
        self.places = places
        self.number = number

    def removeStudent(self, student):
        self.students.remove(student)
        student.detach(self)

    def update(self, student, operation):
        if operation == "delete":
            for stud in self.students:
                if student.id == stud.id:
                    self.removeStudent(stud)
```

Dormitory.py:

```
from Room import Room
import Student


class Dormitory:
    def __init__(self, max_rooms=20, max_places=0):
        self.rooms = []
        self.max_rooms = max_rooms
        self.max_places = max_places

    def addRoom(self, room):
        self.rooms.append(room)


def base():
    global list_rm, dormitory
    list_rm = []
    with open("rooms", 'r') as file:
        lines = file.readlines()
        dormitory = Dormitory()
        for j in range(dormitory.max_rooms):
            room = Room()
            room.number = j + 1
            dormitory.addRoom(room)
            if j in range(len(lines)):
                lines[j] = lines[j].split(";")
        for i in range(len(lines)):
            for j in range(dormitory.max_rooms):
                if j + 1 == int(lines[i][0]):
                    room1 = Room()
                    students_id = lines[i][2].split(",")
                    room1.createRoom(lines[i][0], lines[i][1])
                    for k in students_id:
                        student = Student.findStudent(k)
                        room1.addStudent(student)
                    dormitory.rooms[j] = room1
    file.close()
    with open("dormitory", 'r') as file:
        lines = file.readlines()
        dormitory.max_places = int(lines[0])
```

```
base()


def findRoom(number):
    for i in dormitory.rooms:
        if i.number == number:
            return i


def list_st():
    list_st = []
    for i in dormitory.rooms:
        for j in dormitory.rooms[i].students:
            list_st.append(j)
    return list_st


def get_dormitory():
    return dormitory


def rep_check(student):
    for i in dormitory.rooms:
        for stud in i.students:
            if stud == student:
                return True
    return False
```

Observer.py:

```
from abc import ABC, abstractmethod

class Observer(ABC):
    @abstractmethod
    def update(self, operation, student):
        pass
```

Subject.py:

```
from abc import ABC, abstractmethod

class Subject(ABC):
    @abstractmethod
    def __init__(self):
        self.observers = []

    @abstractmethod
    def attach(self, observer):
        self.observers.append(observer)

    @abstractmethod
    def detach(self, observer):
        self.observers.remove(observer)

    @abstractmethod
    def notify(self, operation):
        for observer in self.observers:
            observer.update(self, operation)
```

Win_Student.py:

```
from tkinter import *
from tkinter import messagebox
import Student
import Group


def win st dl():
    def deleteS():
        i = st_id1.get()
        student = Student.findStudent(i)
        student.notify("delete")
        Student.get_list_st().remove(student)

    win1 = Tk()
    win1.title("Видалення студента")
    win1.geometry("200x200")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Введіть id студента", bg='#4169E1').grid(row=0, padx=5,
pady=5, sticky="nsew")
    st_id1 = Entry(win1, bg="#DEB887")
    st_id1.grid(row=1, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Видалити", command=deleteS, bg="#5F9EA0").grid(row=8,
padx=5, pady=5, sticky="nsew")


def win_st_cr():
    def createS():
        n = st_name.get()
        s = st_surname.get()
        a = st_age.get()
        i = Student.minId()
        if n and s is not None and a.isdigit() is True:
            student = Student.Student()
            student.createStudent(n, s, a, str(i))
            Student.get_list_st().append(student)
        else:
            messagebox.showerror("Помилка", "Дані введено неправильно")

    win1 = Tk()
    win1.title("Створення студента")
    win1.geometry("200x250")
    win1.config(bg='#4169E1')
    Label(win1, text="Ім'я", bg='#4169E1').grid(row=0, padx=5, pady=5,
sticky="nsew")
    st_name = Entry(win1, bg="#DEB887")
    st name.grid(row=1, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Прізвище", bg='#4169E1').grid(row=2, padx=5, pady=5,
sticky="nsew")
    st_surname = Entry(win1, bg="#DEB887")
    st_surname.grid(row=3, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Вік", bg='#4169E1').grid(row=4, padx=5, pady=5,
sticky="nsew")
    st_age = Entry(win1, bg="#DEB887")
    st_age.grid(row=5, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Створити", command=createS, bg="#5F9EA0").grid(row=8,
padx=5, pady=5, sticky="nsew")


def win_st_ch():
    choice = IntVar()
```

```
    def changeS():
        student = Student.findStudent(st_id.get())
        ch = st_change.get()
        if student is not None:
            if choice.get() == 1:
                if ch != '':
                    student.name = ch
                elif ch == '':
                    messagebox.showerror("Помилка", "Дані введено неправильно")
            elif choice.get() == 2:
                if ch != '':
                    student.surname = ch
                elif ch == '':
                    messagebox.showerror("Помилка", "Дані введено неправильно")
            elif choice.get() == 3:
                if ch is int and ch != '':
                    student.age = ch
                elif ch is not int:
                    messagebox.showerror("Помилка", "Дані введено неправильно")
            student.notify("change")
        elif student is None:
            messagebox.showerror("Помилка", "Студента не існує")

    def radio1():
        choice.set(1)

    def radio2():
        choice.set(2)

    def radio3():
        choice.set(3)

    win1 = Tk()
    win1.title("Зміна інформації про студента")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="ID студента", bg='#4169E1').grid(row=0, column=1, padx=5,
pady=5, sticky="nsew")
    st_id = Entry(win1, bg="#DEB887")
    st_id.grid(row=1, column=1, padx=5, pady=5, sticky="nsew")
    rad1 = Radiobutton(win1, text="Змінити ім'я", variable=choice, value=1,
command=radio1, bg='#4169E1')
    rad1.grid(row=2, column=0, padx=5, pady=5, sticky="nsew")
    rad2 = Radiobutton(win1, text="Змінити прізвище", variable=choice, value=2,
command=radio2, bg='#4169E1')
    rad2.grid(row=2, column=1, padx=5, pady=5, sticky="nsew")
    rad3 = Radiobutton(win1, text="Змінити вік", variable=choice, value=3,
command=radio3, bg='#4169E1')
    rad3.grid(row=2, column=2, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Введіть зміни:", bg='#4169E1').grid(row=3, column=1,
padx=10, pady=10, sticky="nsew")
    st_change = Entry(win1, bg="#DEB887")
    st_change.grid(row=4, column=1, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Змінити", command=changeS, bg="#5F9EA0").grid(row=5,
column=1, padx=5, pady=5, sticky="nsew")
    choice.set(1)
    rad1.select()


def win_show_st():
    def show_st():
        list_stud = []
```

```
        students = Student.get_list_st()
        for i in students:
            local_list = f"ID: {i.id}, {i.name} {i.surname}"
            for j in i.observers:
                if type(j) == Group.Group:
                    local_list += f', Група: {j.name}'
            local_list += '\n'
            list_stud.append(local_list)
        list_st = "".join(list_stud)
        text_data.delete(1.0, END)
        text_data.insert(1.0, ''.join(list_st))

    win1 = Tk()
    win1.title("Список всіх студентів")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    text_data = Text(win1, width=50, height=20, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=0, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Показати", command=show_st, bg="#5F9EA0").grid(row=1,
padx=5, pady=5, sticky="nsew")
```

Win_Group.py:

```
from tkinter import *
from tkinter import messagebox
import Group
import Student


def win_gr_cr():
    def createG():
        n = gr_name.get()
        c = gr_course.get()
        groups = Group.get_list_gr()
        isExists = 0
        for i in groups:
            if i.name == n:
                isExists = 1
        if isExists == 1:
            messagebox.showerror("Помилка", "Така група вже існує")
        elif n and c is not None and c.isdigit() is True:
            group = Group.Group()
            group.createGroup(n, c)
            groups.append(group)
        else:
            messagebox.showerror("Помилка", "Дані введено неправильно")

    win1 = Tk()
    win1.title("Створення групи")
    win1.geometry("200x200")
    win1.config(bg='#4169E1')
    Label(win1, text="Назва", bg='#4169E1').grid(row=0, padx=5, pady=5,
sticky="nsew")
    gr_name = Entry(win1, bg="#DEB887")
    gr_name.grid(row=1, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Курс", bg='#4169E1').grid(row=2, padx=5, pady=5,
sticky="nsew")
    gr_course = Entry(win1, bg="#DEB887")
    gr_course.grid(row=3, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Створити", command=createG, bg="#5F9EA0").grid(row=4,
```

```
padx=5, pady=5, sticky="nsew")


def win_gr_dl():
    def deleteG():
        i = gr_name.get()
        groups = Group.get_list_gr()
        for j in groups:
            if j.name == i:
                j.del_group()
                groups.remove(j)
                return
        messagebox.showerror("Помилка", "Групи не знайдено")

    win1 = Tk()
    win1.title("Видалення групи")
    win1.geometry("200x200")
    win1.config(bg='#4169E1')
    Label(win1, text="Введіть назву групи", bg='#4169E1').grid(row=0, padx=5,
pady=5, sticky="nsew")
    gr_name = Entry(win1, bg="#DEB887")
    gr_name.grid(row=1, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Видалити", command=deleteG, bg="#5F9EA0").grid(row=2,
padx=5, pady=5, sticky="nsew")


def win_gr_ch():
    choice = IntVar()

    def changeG():
        line = Group.findGroup(gr_name.get())
        ch = st_change.get()
        if line is not None:
            if choice.get() == 1:
                groups = Group.get_list_gr()
                isExists = 0
                for i in groups:
                    if i.name == ch:
                        isExists = 1
                if isExists == 1:
                    messagebox.showerror("Помилка", "Така група вже існує")
                else:
                    if ch != '':
                        line.name = ch
                    else:
                        messagebox.showerror("Помилка", "Дані введено
неправильно")
            elif choice.get() == 2:
                if ch.isdigit() is True:
                    line.course = ch
                else:
                    messagebox.showerror("Помилка", "Дані введено неправильно")
        else:
            messagebox.showerror("Помилка", "Групи не існує")

    def radio1():
        choice.set(1)

    def radio2():
        choice.set(2)

    win1 = Tk()
    win1.title("Зміна інформації про групу")
```

```
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Назва групи", bg='#4169E1').grid(row=0, columnspan=2,
padx=5, pady=5, sticky="nsew")
    gr_name = Entry(win1, bg="#DEB887")
    gr_name.grid(row=1, columnspan=2, padx=5, pady=5, sticky="nsew")
    rad1 = Radiobutton(win1, text="Змінити назву", variable=choice, value=1,
command=radio1, bg='#4169E1')
    rad1.grid(row=2, column=0, padx=5, pady=5, sticky="nsew")
    rad2 = Radiobutton(win1, text="Змінити курс", variable=choice, value=2,
command=radio2, bg='#4169E1')
    rad2.grid(row=2, column=1, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Введіть зміни:", bg='#4169E1').grid(row=3, columnspan=2,
padx=5, pady=5, sticky="nsew")
    st_change = Entry(win1, bg="#DEB887")
    st_change.grid(row=4, columnspan=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Змінити", command=changeG, bg="#5F9EA0").grid(row=5,
columnspan=2, padx=5, pady=5, sticky="nsew")
    rad1.select()
    choice.set(1)


def win_show_gr():
    def show_gr():
        gr_name = ent_group.get()
        list_group = []
        group = Group.findGroup(gr_name)
        if group is not None:
            local_list = f"Назва: {group.name}, Курс: {group.course}, Кількість
студентів: {len(group.students)}\n"
            for j in group.students:
                local_list = local_list + f"ID: {j.id}, {j.name} {j.surname}\n"
            list_group.append(local_list)
            list_st = "".join(list_group)
            text_data.delete(1.0, END)
            text_data.insert(1.0, ''.join(list_st))
        elif group is None:
            messagebox.showerror("Помилка", "Групи не існує")

    win1 = Tk()
    win1.title("Дані групи")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text='Введіть назву групи', bg='#4169E1').grid(row=0, padx=5,
pady=5, sticky="nsew")
    ent_group = Entry(win1, bg="#DEB887")
    ent_group.grid(row=1, padx=5, pady=5, sticky="nsew")
    text_data = Text(win1, width=100, height=20, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Показати", command=show_gr, bg="#5F9EA0").grid(row=3,
padx=5, pady=5, sticky="nsew")


def win_show_grs():
    def show_grs():
        list_group = []
        groups = Group.get_list_gr()
        for i in groups:
            local_list = f"Назва: {i.name}, Курс: {i.course}, Кількість студентів:
{len(i.students)}\n"
            for j in i.students:
                local_list = local_list + f"ID: {j.id}, {j.name} {j.surname}\n"
```

```
            list_group.append(local_list)
            list_st = "".join(list_group)
            text_data.delete(1.0, END)
            text_data.insert(1.0, ''.join(list_st))

    win1 = Tk()
    win1.title("Список всіх груп")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    text_data = Text(win1, width=100, height=20, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=0, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Показати", command=show_grs, bg="#5F9EA0").grid(row=1,
padx=5, pady=5, sticky="nsew")


def win_gr_st():
    choice = IntVar()

    def changeS_G():
        group = Group.findGroup(gr_name.get())
        ch = st_id.get().split(" ")
        if choice.get() == 1:
            for i in ch:
                if i and group is not None and
Group.rep_check(Student.findStudent(i)) is False:
                    group.addStudent(Student.findStudent(i))
                elif i and group is not None and
Group.rep_check(Student.findStudent(i)) is True:
                    messagebox.showerror("Помилка", "Цей стедент вже знаходиться у
групі")
                elif group is None:
                    messagebox.showerror("Помилка", "Групи не існує")
                elif i is None:
                    messagebox.showerror("Помилка", "Студента не існує")
        elif choice.get() == 2:
            for i in ch:
                if i is not None:
                    group.removeStudent(Student.findStudent(i))

    def radio1():
        choice.set(1)

    def radio2():
        choice.set(2)

    win1 = Tk()
    win1.title("Додавання та видалення студента з групи")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Назва групи", bg='#4169E1').grid(row=0, column=0,
columnspan=2, padx=5, pady=5, sticky="nsew")
    gr_name = Entry(win1, bg="#DEB887")
    gr_name.grid(row=1, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    rad1 = Radiobutton(win1, text="Додати студента", variable=choice, value=1,
command=radio1, bg='#4169E1')
    rad1.grid(row=2, column=0, padx=5, pady=5, sticky="nsew")
    rad2 = Radiobutton(win1, text="Видалити студента", variable=choice, value=2,
command=radio2, bg='#4169E1')
    rad2.grid(row=2, column=1, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Введіть ID студента/студентів(введення через пробіл)",
bg='#4169E1').grid(row=3, column=0,
```

```
columnspan=2, padx=5,

pady=5, sticky="nsew")
    st_id = Entry(win1, bg="#DEB887")
    st_id.grid(row=4, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Змінити", command=changeS_G, bg="#5F9EA0").grid(row=5,
column=0, columnspan=2, padx=5, pady=5,

sticky="nsew")
    choice.set(1)
    rad1.select()
```

Win_Dormitory.py:

```
from tkinter import *
from tkinter import messagebox
import Dormitory
import Student


def win_dm_st():
    choice = IntVar()

    def changeS_D():
        room = Dormitory.findRoom(rm_num.get())
        ch = st_id.get().split(" ")
        if choice.get() == 1:
            for i in ch:
                if i and room is not None and len(room.students) <
int(room.places) and Dormitory.rep_check(
                        Student.findStudent(i)) == False:
                    room.addStudent(Student.findStudent(i))
                elif i and room is not None and len(room.students) >=
int(room.places):
                    messagebox.showerror("Помилка", "Переповнення студентів у
кімнаті")
                    break
                elif i and room is not None and len(room.students) <
int(room.places) and Dormitory.rep_check(
                        Student.findStudent(i)) == True:
                    messagebox.showerror("Помилка", "Студент вже поселений у іншій
кімнаті")
                elif i is None:
                    messagebox.showerror("Помилка", "Студента не існує")
                elif room is None:
                    messagebox.showerror("Помилка", "Кімнати не існує")
        elif choice.get() == 2:
            for i in ch:
                if i and room is not None:
                    room.removeStudent(Student.findStudent(i))
                elif i is None:
                    messagebox.showerror("Помилка", "Студента не існує")
                elif room is None:
                    messagebox.showerror("Помилка", "Кімнати не існує")

    def radio1():
        choice.set(1)

    def radio2():
```

```
        choice.set(2)

    win1 = Tk()
    win1.title("Додавання та видалення студента з гуртожитку")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Номер кімнати", bg='#4169E1').grid(row=0, column=0,
columnspan=2, padx=5, pady=5, sticky="nsew")
    rm_num = Entry(win1, bg="#DEB887")
    rm_num.grid(row=1, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    rad1 = Radiobutton(win1, text="Додати студента", variable=choice, value=1,
command=radio1, bg='#4169E1')
    rad1.grid(row=2, column=0, padx=5, pady=5, sticky="nsew")
    rad2 = Radiobutton(win1, text="Видалити студента", variable=choice, value=2,
command=radio2, bg='#4169E1')
    rad2.grid(row=2, column=1, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Введіть ID студента/студентів(введення через пробіл)",
bg='#4169E1').grid(row=3, column=0,

columnspan=2, padx=5,

pady=5, sticky="nsew")
    st_id = Entry(win1, bg="#DEB887")
    st_id.grid(row=4, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Змінити", command=changeS_D, bg="#5F9EA0").grid(row=5,
column=0, columnspan=2, padx=5, pady=5,

sticky="nsew")
    choice.set(1)
    rad1.select()


def win_ch_rm():
    def changeS_D():
        room = Dormitory.findRoom(rm_num.get())
        pl = st_max.get()
        pl = int(pl)
        if room is not None and pl >= 0 and len(room.students) <= pl:
            pl = str(pl)
            room.places = pl
        elif room is not None and pl >= 0 and len(room.students) > pl:
            messagebox.showerror("Помилка", "Для початку випишіть зайвих
студентів")
        elif room is not None and pl < 0:
            messagebox.showerror("Помилка", "Кількість місць не може бути
від'ємним числом")
        elif room is None:
            messagebox.showerror("Помилка", "Кімнати не існує")
        elif pl is None:
            messagebox.showerror("Помилка", "Число введено неправильно")

    win1 = Tk()
    win1.title("Зміна даних про кімнату")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Номер кімнати", bg='#4169E1').grid(row=0, column=0,
columnspan=2, padx=5, pady=5, sticky="nsew")
    rm_num = Entry(win1, bg="#DEB887")
    rm_num.grid(row=1, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    Label(win1, text="Введіть нову максимальну кількість студентів",
bg='#4169E1').grid(row=3, column=0, columnspan=2,

padx=5, pady=5, sticky="nsew")
```

```
    st_max = Entry(win1, bg="#DEB887")
    st_max.grid(row=4, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Змінити", command=changeS_D, bg="#5F9EA0").grid(row=5,
column=0, columnspan=2, padx=5, pady=5,

sticky="nsew")


def win_ch_dm():
    def changeS_D():
        ch = val.get()
        dormitory = Dormitory.get_dormitory()
        if ch.isdigit() and int(ch) >= 0:
            dormitory.max_places = int(ch)
        elif ch.isdigit() and int(ch) < 0:
            messagebox.showerror("Помилка", "Кількість місць не може бути
від'ємним числом")
        else:
            messagebox.showerror("Помилка", "Число введено неправильно")

    win1 = Tk()
    win1.title("Змінити дані про гуртожиток")
    win1.geometry("200x200")
    win1.config(bg='#4169E1')
    Label(win1, text="Максимальне число студентів", bg='#4169E1').grid(row=3,
column=0, columnspan=2, padx=5, pady=5,

sticky="nsew")
    val = Entry(win1, bg="#DEB887")
    val.grid(row=4, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Змінити", command=changeS_D, bg="#5F9EA0").grid(row=5,
column=0, columnspan=2, padx=5, pady=5,

sticky="nsew")


def win_sh_st():
    choice = IntVar()

    def changeS_D():
        if choice.get() == 1:
            dormitory = Dormitory.get_dormitory()
            list_room = []
            numofstud = 0
            for i in dormitory.rooms:
                local_list = f"Номер кімнати: {i.number}, Кількість вільних місць:
{int(i.places) - len(i.students)}\n"
                for i in i.students:
                    local_list = local_list + f"ID: {i.id}, Студент: {i.name}
{i.surname}, Вік: {i.age} років\n"
                    numofstud += 1
                list_room.append(local_list)
                list_st = "".join(list_room)
                text_data.delete(1.0, END)
                text_data.insert(1.0, ''.join(list_st))
            text data.insert(0.0, f'Кількість студентів у гуртожитку:
{numofstud}\n')
        elif choice.get() == 2:
            list_room = []
            room = Dormitory.findRoom(rm_num.get())
            if room is not None:
                local_list = f"Номер кімнати: {room.number}, Кількість вільних
місць: {int(room.places) - len(room.students)}\n"
```

```
                for i in room.students:
                    local_list = local_list + f"ID: {i.id}, Студент: {i.name}
{i.surname}, Вік: {i.age} років\n"
                list_room.append(local_list)
                list_st = "".join(list_room)
                text_data.delete(1.0, END)
                text_data.insert(1.0, ''.join(list_st))
            elif room is None:
                messagebox.showerror("Помилка", "Кімнати не існує")

    def radio1():
        choice.set(1)

    def radio2():
        choice.set(2)

    win1 = Tk()
    win1.title("Інформація про гуртожиток")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Номер кімнати", bg='#4169E1').grid(row=1, column=0,
columnspan=2, padx=5, pady=5, sticky="nsew")
    rm_num = Entry(win1, bg="#DEB887")
    rm_num.grid(row=2, column=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    rad1 = Radiobutton(win1, text="Повна інформація", variable=choice, value=1,
command=radio1, bg='#4169E1')
    rad1.grid(row=3, column=1, padx=5, pady=5, sticky="nsew")
    rad2 = Radiobutton(win1, text="Інформація про кімнату", variable=choice,
value=2, command=radio2, bg='#4169E1')
    rad2.grid(row=3, column=0, padx=5, pady=5, sticky="nsew")
    text_data = Text(win1, width=100, height=20, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=0, columnspan=2, padx=5, pady=5, sticky="nsew")
    Button(win1, text="Показати", command=changeS_D, bg="#5F9EA0").grid(row=4,
column=0, columnspan=2, padx=5, pady=5,

sticky="nsew")
    choice.set(1)
    rad1.select()
```

Win_Search:

```
from tkinter import *
import Group
import Student
import Dormitory
import Room

def win_sr_st():

    def search():
        info = st_in.get()
        list_st = Student.get_list_st()
        list_fn = []
        list_stud = []
        for i in list_st:
            if info in i.name:
                list_fn.append(i)
            elif info in i.surname:
                list_fn.append(i)
        for student in list_fn:
```

```
            local_list = f"ID: {student.id}, {student.name} {student.surname}"
            for i in student.observers:
                if type(i) == Group.Group:
                    local_list += f', Група: {i.name}'
            local_list = local_list + '\n'
            list_stud.append(local_list)
        list_st = "".join(list_stud)
        text_data.delete(1.0, END)
        text_data.insert(1.0, ''.join(list_st))


    win1 = Tk()
    win1.title("Пошук")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Введіть ім'я або прізвище", bg='#4169E1').grid(row=0,
column=0,padx=5, pady=5, sticky="nsew")
    st_in = Entry(win1, bg="#DEB887")
    st_in.grid(row=1, column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    Button(win1, text="Знайти", command=search, bg="#5F9EA0").grid(row=2,
column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    text_data = Text(win1, width=60, height=10, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=3,padx=5, pady=5, sticky="nsew")

def win_sr_gr():

    def search():
        info = st_in.get()
        list_st = Group.findGroup(gr_in.get()).students
        list_fn = []
        list_stud = []
        for i in list_st:
            if info in i.name:
                list_fn.append(i)
            if info in i.surname:
                list_fn.append(i)
        for student in list_fn:
            local_list = f"ID: {student.id}, {student.name} {student.surname}\n"
            list_stud.append(local_list)
        list_st = "".join(list_stud)
        text_data.delete(1.0, END)
        text_data.insert(1.0, ''.join(list_st))


    win1 = Tk()
    win1.title("Пошук у групі")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Введіть назву групи", bg='#4169E1').grid(row=0,
column=0,padx=5, pady=5, sticky="nsew")
    gr_in = Entry(win1, bg="#DEB887")
    gr_in.grid(row=1, column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    Label(win1, text="Введіть ім'я або прізвище", bg='#4169E1').grid(row=2,
column=0,padx=5, pady=5, sticky="nsew")
    st_in = Entry(win1, bg="#DEB887")
    st_in.grid(row=3, column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    Button(win1, text="Знайти", command=search, bg="#5F9EA0").grid(row=4,
column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    text_data = Text(win1, width=60, height=10, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=5,padx=5, pady=5, sticky="nsew")
```

```
def win_sr_dm():

    def search():
        info = st_in.get()
        dormitory = Dormitory.get_dormitory()
        list_fn = []
        list_stud = []
        for i in dormitory.rooms:
            for j in i.students:
                if info in j.name:
                    list_fn.append(j)
                if info in j.surname:
                    list_fn.append(j)
        for student in list_fn:
            local_list = f"ID: {student.id}, {student.name} {student.surname}"
            for i in student.observers:
                if type(i) == Room.Room:
                    local_list += f', Кімната: {i.number}'
            local_list = local_list + '\n'
            list_stud.append(local_list)
        list_st = "".join(list_stud)
        text_data.delete(1.0, END)
        text_data.insert(1.0, ''.join(list_st))


    win1 = Tk()
    win1.title("Пошук у гуртожитку")
    win1.resizable(True, True)
    win1.config(bg='#4169E1')
    Label(win1, text="Введіть ім'я або прізвище", bg='#4169E1').grid(row=0,
column=0,padx=5, pady=5, sticky="nsew")
    st_in = Entry(win1, bg="#DEB887")
    st_in.grid(row=1, column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    Button(win1, text="Знайти", command=search, bg="#5F9EA0").grid(row=2,
column=0, columnspan=2,padx=5, pady=5, sticky="nsew")
    text_data = Text(win1, width=60, height=10, bg="#DEB887")
    text_data.configure(font=("Ariel", 11))
    text_data.grid(row=3,padx=5, pady=5, sticky="nsew")
```