Звіт до лабораторної роботи №3

**ООП**

Під час виконання практичного завдання до лабораторної роботи №3 було надано варіанти рішення до наступних задач:

1. озробити клас Студент групи з відповідними атрибутами;
2. розробити клас Список групи, має містити не словники, як виконано в лабораторній роботі №2, а об’єкти класу Студент групи; додавання нового запису, видаленні існуючого чи зміна даних має бути виконана через методи класу Список групи.
3. розробити клас для роботи з файлами для зчитування початкової інформації про список групи та збереження інформації по завершенню програми.
4. список студентів має містити не словники, як виконано в лабораторній роботі №2, а об’єкти класу Студент групи;
5. описання всіх класів мають міститися в окремих файлах, що мають відповідні імена(наприклад Studen, StudentList, Utils)
6. основний функціонал програми має бути покритий Юніт тестами.

**Виконання роботи:**

Сама программа була розбита по окремим файлах з відповідними назвами.

**Текст програми:**

**FileManager.py**

|  |
| --- |
| import csv  from Student import Student  from StudentList import StudentList  class FileManager:      @staticmethod      def load\_from\_csv(filename: str) -> StudentList:          student\_list = StudentList()          try:              with open(filename, newline='', encoding="utf-8") as csvfile:                  reader = csv.DictReader(csvfile)                  for row in reader:                      student = Student(row["name"], row["phone"], row["surname"], row["course"])                      student\_list.add\_student(student)          except FileNotFoundError:              print(f"File {filename} not found.")          return student\_list      @staticmethod      def save\_to\_csv(filename: str, student\_list: StudentList):          try:              with open(filename, mode="w", newline='', encoding="utf-8") as csvfile:                  fieldnames = ["name", "phone", "surname", "course"]                  writer = csv.DictWriter(csvfile, fieldnames=fieldnames)                  writer.writeheader()                  for student in student\_list.to\_dict\_list():                      writer.writerow(student)          except IOError:              print(f"Error writing to file {filename}.") |

main.py

|  |
| --- |
| import sys  from Student import Student  from StudentList import StudentList  from FileManager import FileManager  def main(filename):      student\_list = FileManager.load\_from\_csv(filename)      while True:          choice = input("Choose action [C create, U update, D delete, P print, X exit]: ").lower()          match choice:              case "c":                  name = input("Enter name: ")                  phone = input("Enter phone: ")                  surname = input("Enter surname: ")                  course = input("Enter course: ")                  student = Student(name, phone, surname, course)                  student\_list.add\_student(student)                  print("Student added.")              case "u":                  name = input("Enter name of student to update: ")                  phone = input("Enter phone: ")                  print("Enter new data:")                  new\_name = input("Name: ")                  new\_phone = input("Phone: ")                  new\_surname = input("Surname: ")                  new\_course = input("Course: ")                  updated\_student = Student(new\_name, new\_phone, new\_surname, new\_course)                  student\_list.update\_student(name, phone, updated\_student)              case "d":                  name = input("Enter name of student to delete: ")                  phone = input("Enter phone: ")                  student\_list.delete\_student(name, phone)                  print("Student deleted.")              case "p":                  student\_list.print\_all()              case "x":                  FileManager.save\_to\_csv(filename, student\_list)                  print("Data saved. Exiting...")                  break              case \_:                  print("Invalid option.")  if \_\_name\_\_ == "\_\_main\_\_":      if len(sys.argv) != 2:          print("Usage: python main.py <filename>")      else:          main(sys.argv[1]) |

**Student.py**

|  |
| --- |
| class Student:      def \_\_init\_\_(self, name: str, phone: str, surname: str, course: str):  *self*.name = name  *self*.phone = phone  *self*.surname = surname  *self*.course = course      def \_\_str\_\_(self):          return f"{*self*.name} {*self*.surname}, Phone: {*self*.phone}, Course: {*self*.course}"      def to\_dict(self):          return {              "name": *self*.name,              "phone": *self*.phone,              "surname": *self*.surname,              "course": *self*.course          } |

**StudentList.py**

|  |
| --- |
| from Student import Student  class StudentList:      def \_\_init\_\_(self):  *self*.students = []      def add\_student(self, student: Student):          """Додає нового студента до списку (в алфавітному порядку за іменем)."""  *self*.students.append(student)  *self*.students.sort(key=lambda s: s.name.lower())      def delete\_student(self, name: str, phone: str):          """Видаляє студента зі списку."""  *self*.students = [s for s in *self*.students if not (s.name == name and s.phone == phone)]      def update\_student(self, name: str, phone: str, updated\_student: Student):          """Оновлює інформацію про студента."""          for i, student in enumerate(*self*.students):              if student.name == name and student.phone == phone:  *self*.students[i] = updated\_student  *self*.students.sort(key=lambda s: s.name.lower())                  return          print("Student not found.")      def print\_all(self):          """Виводить список студентів."""          for student in *self*.students:              print(student)      def to\_dict\_list(self):          """Перетворює список студентів у формат списку словників."""          return [s.to\_dict() for s in *self*.students] |

Та було реалізовані юніт-тести для основних функцій програми.

|  |
| --- |
| import pytest  from Student import Student  from StudentList import StudentList  def test\_add\_student():      sl = StudentList()      student = Student("Alice", "123", "Brown", "1")      sl.add\_student(student)      assert len(sl.students) == 1      assert sl.students[0].name == "Alice"  def test\_delete\_student():      sl = StudentList()      student = Student("Alice", "123", "Brown", "1")      sl.add\_student(student)      sl.delete\_student("Alice", "123")      assert len(sl.students) == 0  def test\_update\_student():      sl = StudentList()      student = Student("Alice", "123", "Brown", "1")      sl.add\_student(student)      updated\_student = Student("Alice", "123", "Green", "2")      sl.update\_student("Alice", "123", updated\_student)      assert sl.students[0].surname == "Green"      assert sl.students[0].course == "2" |

**Посилання на github:**

<https://github.com/Alexxxxo/TP-KB-231-Oleksandr-Kozachok/tree/main/lab_03>

Висновок: у процесі виконання лабораторної роботи №3 було реалізовано завантаження даних із CSV-файлу, збереження довідника студентів у цьому ж форматі та переробка у ООП та створення юніт-тестів для перевірки основного функціоналу програми.