Кондратенко Катерина КБ-232

Лабораторна робота №3

**ООП**

**Мета роботи**: Використовуючи теоретичне підґрунтя про ООП у мові Python переробити програму телефонного довідника студентів використовуючи принципи ООП для формування відомостей про студентів.

**Хід роботи**

**Завдання до лабораторної роботи:** Переробити функціональність телефонного довідника студентів групи, що був розроблений у Лабораторній роботі №2 використовуючи принципи ООП:

1. розробити 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.

Текст програми:

**student.py**

class Student:

    def \_\_init\_\_(self, name, phone, email, address):

        self.name = name

        self.phone = phone

        self.email = email

        self.address = address

    def \_\_str\_\_(self):

        return f"Student name: {self.name}, Phone: {self.phone}, Email: {self.email}, Address: {self.address}"

**studentList.py**

class StudentList:

    def \_\_init\_\_(self):

        self.students = []

    def addStudent(self, student):

        insert\_position = 0

        for i in range(len(self.students)):

            if student.name.lower() > self.students[i].name.lower():

                insert\_position += 1

            else:

                break

        self.students.insert(insert\_position, student)

        print("New student has been added.")

    def deleteStudent(self, name):

        delete\_position = -1

        for i, student in enumerate(self.students):

            if name.lower() == student.name.lower():

                delete\_position = i

                break

        if delete\_position != -1:

            del self.students[delete\_position]

            print(f"Student with name {name} has been deleted.")

        else:

            print(f"No student with name {name} found.")

    def updateStudent(self, name, new\_name=None, phone=None, email=None, address=None):

        student = next((s for s in self.students if s.name == name), None)

        if not student:

            print("Student was not found.")

            return

        if new\_name:

            student.name = new\_name

        if phone:

            student.phone = phone

        if email:

            student.email = email

        if address:

            student.address = address

        print(f"Information about {name} has been updated.")

    def printAll(self):

        if not self.students:

            print("Student list is empty.")

        for student in self.students:

            print(student)

**file.py**

import csv

from student import Student

class FileHandler:

    def loadFromCsv(self, file\_name, student\_list):

        try:

            with open(file\_name, mode='r') as file:

                reader = csv.DictReader(file)

                for row in reader:

                    student = Student(row['name'], row['phone'], row['email'], row['address'])

                    studentList.addStudent(student)

            print(f"Data successfully loaded from {file\_name}")

        except FileNotFoundError:

            print(f"File {file\_name} not found. Starting with an empty student list.")

    def saveToCsv(self, file\_name, student\_list):

        with open(file\_name, mode='w', newline='') as file:

            fieldnames = ["name", "phone", "email", "address"]

            writer = csv.DictWriter(file, fieldnames=fieldnames)

            writer.writeheader()

            for student in studentList.students:

                writer.writerow({

                    "name": student.name,

                    "phone": student.phone,

                    "email": student.email,

                    "address": student.address

                })

        print(f"Data successfully saved to {file\_name}")

**main.py**

import sys

from student import Student

from studentList import StudentList

from file import FileHandler

def main():

    studentList = StudentList()

    fileHandler = FileHandler()

    if len(sys.argv) > 1:

        fileHandler.loadFromCsv(sys.argv[1], student\_list)

    while True:

        choice = input("Please specify the action [ C create, U update, D delete, P print, X exit ]: ")

        if choice.lower() == "c":

            name = input("Please enter student name: ")

            phone = input("Please enter student phone: ")

            email = input("Please enter student email: ")

            address = input("Please enter student address: ")

            student = Student(name, phone, email, address)

            studentList.addStudent(student)

        elif choice.lower() == "u":

            name = input("Please enter the name of the student to be updated: ")

            new\_name = input("Enter new name (leave blank to keep current): ") or None

            phone = input("Enter new phone (leave blank to keep current): ") or None

            email = input("Enter new email (leave blank to keep current): ") or None

            address = input("Enter new address (leave blank to keep current): ") or None

            studentList.updateStudent(name, new\_name, phone, email, address)

        elif choice.lower() == "d":

            name = input("Please enter name to be deleted: ")

            studentList.deleteStudent(name)

        elif choice.lower() == "p":

            studentList.printAll()

        elif choice.lower() == "x":

            if len(sys.argv) > 1:

                fileHandler.saveToCsv(sys.argv[1], student\_list)

            print("Exit()")

            break

        else:

            print("Wrong choice")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**test.py**

import unittest

from student import Student

from studentList import StudentList

class TestStudentList(unittest.TestCase):

    def setUp(self):

        self.studentList = StudentList()

        self.student = Student("John Doe", "123-456-7890", "john@example.com", "123 Elm St")

    def test\_add\_student(self):

        self.studentList.addStudent(self.student)

        self.assertEqual(len(self.studentList.students), 1)

        self.assertEqual(self.studentList.students[0].name, "John Doe")

    def test\_delete\_student(self):

        self.studentList.addStudent(self.student)

        self.studentList.deleteStudent("John Doe")

        self.assertEqual(len(self.studentList.students), 0)

    def test\_update\_student(self):

        self.studentList.addStudent(self.student)

        self.studentList.updateStudent("John Doe", new\_name="Jane Doe")

        self.assertEqual(self.studentList.students[0].name, "Jane Doe")

    def test\_print\_all(self):

        self.studentList.addStudent(self.student)

        from io import StringIO

        import sys

        capturedOutput = StringIO()

        sys.stdout = capturedOutput

        self.studentList.printAll()

        sys.stdout = sys.\_\_stdout\_\_

        self.assertIn("John Doe", capturedOutput.getvalue())

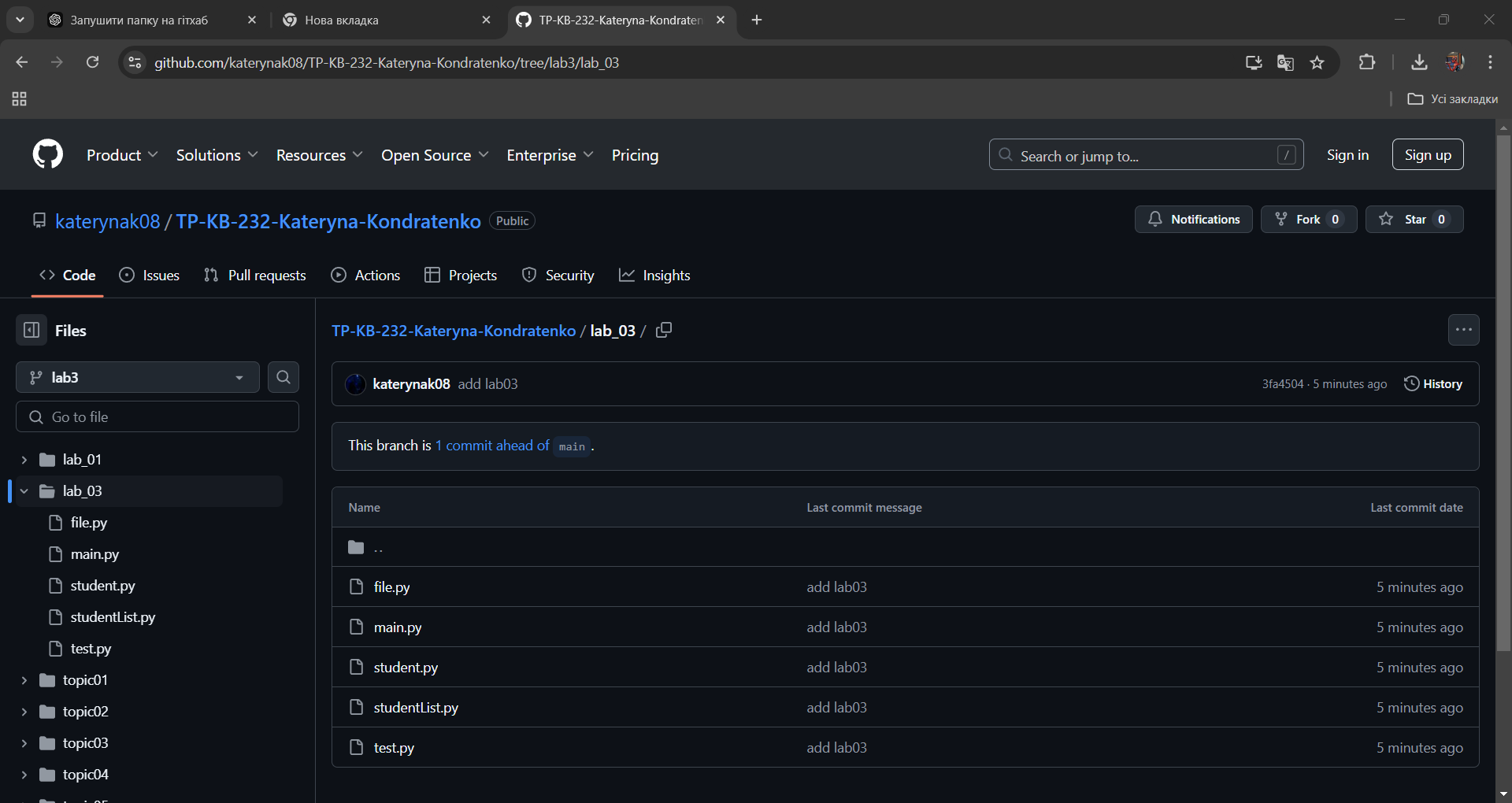
if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

Посилання на github:

https://github.com/katerynak08/TP-KB-232-Kateryna-Kondratenko/tree/lab3/lab\_03

Знімок екрану з посилання на github:



**Висновок**: у результаті виконання лабораторної роботи було здійснено переробку програми телефонного довідника студентів з використанням принципів об'єктно-орієнтованого програмування (ООП) на мові Python. Було створено класи для збереження та обробки інформації про студентів, що дозволило зробити код більш структурованим і зручним для подальшого розвитку. Впровадження ООП дозволило забезпечити гнучкість та масштабованість програми.