Технології програмування

Лабораторна робота №3

**ООП**

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

**Завдання**

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.

**Хід роботи**

1. Розділив функції по різним файлам та переробив їх по стилю ООП
2. Створив клас Student у файлі student.py.
3. Створив studentList куди переніс перероблені функції додавання, видалення, оновлення елементів і т.д.
4. Створив файл utils.py для підгрузки та завантаження даних з csv файла.
5. Створив файл main та додав до нього перероблений функціонал def main() з другої лабораторної роботи.
6. Повністю переробив юніт-тести з попередньої л.р. також додав окремий файл csv формату для праці з тестами.
7. Запустив тести та перевірив правильність виконання задач та завдання.

Код student.py:

class Student:

    def \_\_init\_\_(self, name, phone, profession, groupNum):

        self.name = name

        self.phone = phone

        self.profession = profession

        self.groupNum = groupNum

Код studentList.py:

from student import \*

class StudentList():

    def \_\_init\_\_(self) -> None:

        self.students = []

    def \_\_str\_\_(self) -> str:

        string = ""

        for student in self.students:

            string += f"name {student.name}, phone {student.phone}, profession {student.profession}, groupNum {student.groupNum}\n"

        return string

    def addStudent(self, student: Student):

        insertPosition = 0

        for item in self.students:

            if student.name > item.name:

                insertPosition += 1

            else:

                break

        self.students.insert(insertPosition, student)

        print("New element has been added")

    def deleteStudent(self, studentName):

        deletePosition = -1

        for student in self.students:

            if student.name == studentName:

                deletePosition = self.students.index(student)

                break

        if deletePosition == -1:

            print("Element not found")

        else:

            print("Delete position " + str(deletePosition))

            del self.students[deletePosition]

    def updateStudent(self, studentName, newStudent: Student):

        updatePosition = -1

        for student in self.students:

            if studentName == student.name:

                updatePosition = self.students.index(student)

                break

        if updatePosition == -1:

            print("Element was not found")

            return

        if studentName != newStudent.name:

            del self.students[updatePosition]

            self.addStudent(newStudent)

        else:

            self.students[updatePosition].phone = newStudent.phone

            del newStudent

Код main.py:

from student import \*

from studentList import \*

from utils import \*

def main():

    studentList = StudentList()

    file = input("Enter the name of your file: ")

    studentList = Utils.loadFile(file, studentList)

    while True:

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

        match choice:

            case "C" | "c":

                print("New element will be created")

                student = Student(input("Enter name: "), input("Enter phone: "), input("Enter profession: "), input("Enter group number: "))

                studentList.addStudent(student)

            case "U" | "u":

                print("Existing element will be updated")

                name = input("Enter student name to be updated: ")

                newStudent = Student(input("Enter new name: "), input("Enter new phone: "), input("Enter new age: "), input("Enter new email: "))

                studentList.updateStudent(name, newStudent)

            case "D" | "d":

                print("Element will be deleted")

                name = input("Enter student name: ")

                studentList.deleteStudent(name)

            case "P" | "p":

                print("List will be printed")

                print(studentList)

            case "S" | "s":

                file = input("Enter CSV file name: ")

                Utils.saveFile(file, studentList)

            case "X" | "x":

                print("Exit")

                break

            case \_:

                print("Wrong choice")

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

    main()

Код utils.py:

from student import \*

from studentList import \*

import csv

class Utils:

    @staticmethod

    def loadFile(file, studentList):

        with open(file, newline='') as csvFile:

            reader = csv.DictReader(csvFile)

            for row in reader:

                student = Student(row["name"], row["phone"], row["profession"], row["groupNum"])

                studentList.addStudent(student)

        return studentList

    @staticmethod

    def saveFile(file, studentList):

            with open(file, 'w', newline="") as SaveFile:

                fieldnames = ["name", "phone", "profession", "groupNum"]

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

                writer.writeheader()

                for student in studentList.students:

                    writer.writerow({"name": student.name, "phone": student.phone, "profession": student.profession, "groupNum": student.groupNum})

                print("File is saved.")

Код unittest.py:

import unittest

from unittest.mock import patch

from io import StringIO

from main import main

import os

from utils import Utils

from studentList import \*

from student import \*

class Tests(unittest.TestCase):

    def test\_Student(self):

        student = Student("Bobb", "061234567", "CS", "221")

        self.assertEqual(student.name, "Bobb")

        self.assertEqual(student.phone, "061234567")

        self.assertEqual(student.profession, "CS")

        self.assertEqual(student.groupNum, "221")

    def setUp(self):

        self.studentList = StudentList()

        self.student1 = Student("Bob", "061234567", "CS", "221")

        self.student2 = Student("Zak", "061234567", "PE", "223")

    def test\_addStudent(self):

        self.studentList.addStudent(self.student1)

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

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

        self.studentList.addStudent(self.student2)

        self.assertEqual(len(self.studentList.students), 2)

        self.assertEqual(self.studentList.students[1].name, "Zak")

    def test\_deleteStudent(self):

        self.studentList.addStudent(self.student1)

        self.studentList.addStudent(self.student2)

        self.studentList.deleteStudent("Bob")

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

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

        self.studentList.deleteStudent("John")

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

    def test\_updateStudent(self):

        self.studentList.addStudent(self.student1)

        self.studentList.addStudent(self.student2)

        newStudent = Student("Bob", "061234567", "CS", "221")

        self.studentList.updateStudent("Bob", newStudent)

        self.assertEqual(len(self.studentList.students), 2)

        self.assertEqual(self.studentList.students[0].phone, "061234567")

        self.assertEqual(self.studentList.students[0].groupNum, "221")

        nonExistingStudent = Student("John", "061234567", "CS", "223")

        self.studentList.updateStudent("John", nonExistingStudent)

        self.assertEqual(len(self.studentList.students), 2)

    def test\_loadFile(self):

        self.testFile = "unittests.csv"

        self.studentList = StudentList()

        with open(self.testFile, "w") as file:

            file.write("name,phone,profession,groupNum\nBobb,061234567,CS,221")

        Utils.loadFile(self.testFile, self.studentList)

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

    def test\_saveFile(self):

        self.testFile = "unittests.csv"

        self.studentList = StudentList()

        student = Student("Bobb", "061234567", "CS", "221")

        self.studentList.addStudent(student)

        Utils.saveFile(self.testFile, self.studentList)

        self.assertTrue(os.path.exists(self.testFile))

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

    unittest.main()

**Висновок:**

Під час виконання даної лабораторної роботи використовуючи теоретичне підґрунтя про ООП у мові Python переробив програму телефонного довідника студентів використовуючи принципи ООП для формування відомостей про студентів.