Технології програмування

Лабораторна робота №3

**ОПП**

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

**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.

**Хід роботи:**

1. Розробила клас, що дозволяє створювати об'єкти, які представляють студентів, і зберігати їх основні контактні дані (ім'я, телефон, електронну пошту, групу). А також він має метод для виведення інформації про студента.

Код student.py:

|  |
| --- |
| class Student:  def \_\_init\_\_(self, name, phone, gmail, group):  self.name = name  self.phone = phone  self.gmail = gmail  self.group = group    def \_\_str\_\_(self):  return f"name: {self.name}, phone: {self.phone}, gmail: {self.gmail}, group: {self.group}" |

Опис класу Student:

Конструктор \_\_init\_\_: У цьому класі конструктор приймає 4 параметри: name, phone, gmail, group. Значення цих параметрів зберігаються як атрибути об'єкта, тобто вони стають властивостями конкретного студента.

Метод \_\_str\_\_: Це метод визначає, як об'єкт класу буде виглядати, коли його вивести у вигляді рядка. Він повертає строку, яка містить інформацію про студента у вигляді: ім’я, телефон, електронну пошту та групу.

1. Розробила клас StudentList, що є контейнером для об’єктів класу Student. Це клас дозволяє керувати списком студентів, виконуючи різні операції з ним, такі як додавання, видалення, оновлення та виведення студентів.

Код studentList.py:

|  |
| --- |
| class StudentList:  def \_\_init\_\_(self):  self.students = []    def printAllList(self):  for student in self.students:  print(student)    def addNewElement(self, 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 deleteElement(self, name):  student = next((s for s in self.students if s.name == name), None)  if student:  self.students.remove(student) # Пряме видалення студента зі списку  print(f"Deleted student: {student.name}")  else:  print("Student not found.")    def updateElement(self, name, attribute, newValue):  student = next((s for s in self.students if s.name == name), None)  if student:  match attribute:  case "name":  student.name = newValue  case "phone":  student.phone = newValue  case "gmail":  student.gmail = newValue  case "group":  student.group = newValue  case \_:  print(f"Unknown attribute: {attribute}")  else:  print("Student not found.") |

Опис класу StudentList:

* Конструктор \_\_init\_\_: При створенні об`єкта класу StudentList ініціалізується порожній список students, який буде використовуватись для зберігання студентів.
* Метод printAllList: Цей метод виводить усіх студентів у списку, викликаючи метод \_\_str\_\_ для кожного студента (який визначено в класі Student). Для цього метод ітерує через список студентів і друкує кожен об'єкт.
* Метод addNewElement: Цей метод додає нового студента в список, зберігаючи його в алфавітному порядку за іменем. Для цього він порівнює ім’я нового студента з іменами вже існуючих студентів у списку і визначає позицію для вставки. Якщо ім’я нового студента лексикографічно більше за ім’я поточного студента в списку, позиція для вставки збільшується, інакше додається на поточну позицію.
* Метод deleteElement: Цей метод видаляє студента з списку за ім'ям. Він шукає студента в списку за допомогою генератора (через next), який знаходить перший студент, що відповідає заданому імені. Якщо студент знайдений, він видаляється зі списку. Якщо не знайдений — виводиться повідомлення про помилку.
* Метод updateElement: Цей метод дозволяє оновити певну інформацію про студента за його іменем. Він шукає студента за допомогою імені, а потім змінює одне з атрибутів студента на нове значення. Якщо атрибут, який потрібно змінити, не є правильним, виводиться повідомлення про помилку.

1. Розробила клас Utils, який надає два основні методи для роботи з даними студентів у форматі CSV.

Код utils.py:

|  |
| --- |
| import csv  from student import Student  from studentList import StudentList    class Utils:  def loadDataCSV(self, filename):  studentList = StudentList()  try:  with open(filename, 'r', newline='') as csvfile:  reader = csv.DictReader(csvfile, delimiter=';')  for row in reader:  student = Student(row["name"], row["phone"], row["gmail"], row["group"])  studentList.addNewElement(student)  print(f"The data from {filename} was loaded successfully.")  return studentList  except Exception as e:  print(f"Error loading data from {filename}: {e}")  return studentList    def saveDataCSV(self, filename, studentList):  try:  with open(filename, 'w', newline='') as csvfile:  fieldnames = ['name', 'phone', 'gmail', 'group']  writer = csv.DictWriter(csvfile, fieldnames=fieldnames, delimiter=';')  writer.writeheader()  for student in studentList.students:  writer.writerow({  'name': student.name,  'phone': student.phone,  'gmail': student.gmail,  'group': student.group  })  print(f"Data has been successfully saved to {filename}.")  except Exception as e:  print(f"Error saving data to {filename}: {e}") |

Опис класу Utils:

Метод loadDataCSV(self, filename): Цей метод відповідає за завантаження даних з файлу CSV в список студентів (StudentList). Він приймає параметр filename, що містить шлях до CSV файлу, який потрібно прочитати. Використовує Python модуль csv, щоб відкрити файл і прочитати його вміст за допомогою csv.DictReader. Це дозволяє працювати з CSV як з набором словників, де ключі — це назви колонок, а значення — відповідні дані з файлу. Для кожного рядка в файлі створюється об'єкт класу Student із значень, що відповідають ключам: name, phone, gmail, group. Потім цей об'єкт додається в список студентів через метод addNewElement класу StudentList. Якщо файл успішно прочитано, виводиться повідомлення про успіх. Якщо виникає помилка, виводиться повідомлення про помилку.

Метод saveDataCSV(self, filename, studentList): Цей метод відповідає за збереження даних студентів з об'єкта StudentList у файл CSV.

Він приймає два параметри: filename – шлях до файлу, в який потрібно зберегти дані; studentList – об'єкт класу StudentList, що містить список студентів. Метод відкриває файл для запису і створює запис через csv.DictWriter, вказуючи порядок колонок (fieldnames). Потім для кожного студента з studentList.students метод записує відповідні дані (ім’я, телефон, gmail, групу) в CSV файл. Якщо збереження пройшло успішно, виводиться повідомлення про успіх. Якщо виникає помилка, виводиться повідомлення про помилку.

1. Написала програму для керування списком студентів, яка дозволяє виконувати кілька операцій з даними студентів через командний інтерфейс. Програма працює з класамиStudent, StudentList та Utils й взаємодіє з файлом CSV для збереження та завантаження даних.

Код main.py:

|  |
| --- |
| import sys  from utils import Utils  from studentList import StudentList  from student import Student  from lab3\_test import studentAdd\_test    def main():  filename = "studentsData.csv"  studentList = StudentList()  forFileCVS = Utils()    if len(sys.argv) > 1:  studentList = forFileCVS.loadDataCSV(sys.argv[1])    while True:  choice = input("Please specify the action [ C create, U update, D delete, P print, X exit ] ")    match choice:  case "C" | "c":  name = input("Please enter student name: ")  phone = input("Please enter student phone: ")  gmail = input("Please enter student gmail: ")  group = input("Please enter student group: ")    student = Student(name, phone, gmail, group)  studentList.addNewElement(student)  studentList.printAllList()    case "U" | "u":  name = input("Please enter name to be updated: ")  attribute = input("What data do you want to change (name|phone|gmail|group): ")  newValue = input(f"Enter new value for {attribute}: ")  studentList.updateElement(name, attribute, newValue)  studentList.printAllList()    case "D" | "d":  name = input("Please enter name to be deleted: ")  studentList.deleteElement(name)  studentList.printAllList()    case "P" | "p":  studentList.printAllList()    case "X" | "x":  print("Exiting()")  forFileCVS.saveDataCSV(filename, studentList)  break    case \_:  print("Wrong choice.")    if \_\_name\_\_ == '\_\_main\_\_':  main() |

Опис:

Програма в файлі main реалізує керування списком студентів через текстовий інтерфейс. Вона дозволяє створювати, оновлювати, видаляти та переглядати студентів, а також зберігати дані у CSV файлі. Програма імпортує класи Utils, StudentList та Student. Клас StudentList відповідає за зберігання студентів і операції з ними, клас Utils дозволяє завантажувати та зберігати дані з CSV, а Student представляє студента з інформацією про ім’я, телефон, пошту та групу. При запуску програма завантажує дані з CSV файлу (якщо він вказаний в аргументі командного рядка) або працює з порожнім списком. Далі надається меню для виконання таких операцій:

* 1. C/c - Створення нового студента: Користувач вводить дані студента, який додається до списку.
* 2. U/u - Оновлення даних студента: Користувач обирає студента та атрибут для зміни.
* 3. D/d - Видалення студента: Користувач вводить ім'я студента для видалення.
* 4. P/p - Перегляд всіх студентів: Виведення списку студентів.
* 5. X/x - Вихід з програми: Програма зберігає дані в CSV і завершується.

Програма повторює цикл меню, поки не вибрана опція для виходу. У разі неправильної команди виводиться повідомлення про помилку.

1. Розробила Юнікс тести на основі нових змін у коді.

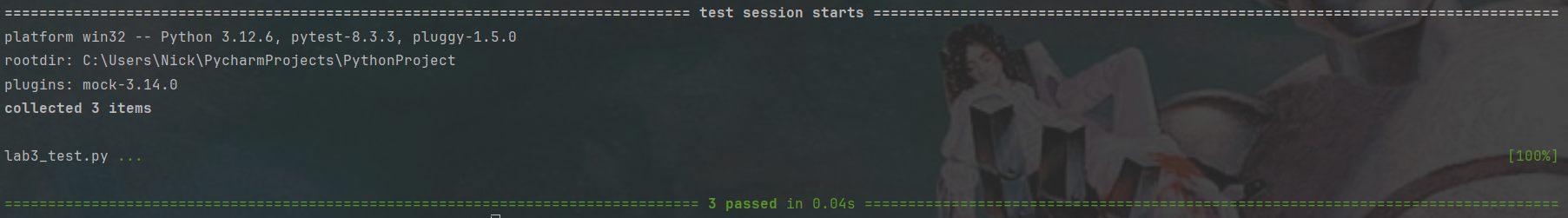
Код lab\_03\_test.py:

|  |
| --- |
| import pytest  from student import Student  from studentList import StudentList  from utils import Utils    @pytest.fixture  def fixStudents():  studentList = StudentList()  studentList.addNewElement(Student("Sofi", "0631234567", "sofi@gmail.com", "cs-95"))  return studentList    def test\_addNewElement(fixStudents):  initialLen = len(fixStudents.students)  fixStudents.addNewElement(Student("John", "1234567890", "john@gmail.com", "cs-101"))  assert len(fixStudents.students) == initialLen + 1  assert any(student.name == "John" for student in fixStudents.students)    def test\_updateNewElement(fixStudents):  fixStudents.updateElement("Sofi", "group", "cs-45")  updatedStudent = next(student for student in fixStudents.students if student.name == "Sofi")  assert updatedStudent.group == "cs-45"    def test\_deleteElement(fixStudents):  initialLen = len(fixStudents.students)  fixStudents.deleteElement("Sofi")  assert len(fixStudents.students) == initialLen - 1  assert not any(student.name == "Sofi" for student in fixStudents.students) |

Опис:

* Фікстура fixStudents: Це фікстура, яка створює новий список студентів (studentList) і додає одного студента з ім'ям "Sofi" до цього списку. Фікстура дозволяє повторно використовувати цей список студентів у кількох тестах, щоб не створювати його кожного разу заново.
* Тест test\_addNewElement: Тест перевіряє, чи можна додати нового студента до списку. Спочатку записується початкова кількість студентів у списку. Потім додається новий студент (John), і перевіряється, чи збільшилася кількість студентів на 1, а також чи присутній студент "John" у списку.
* Тест test\_updateNewElement: Тест перевіряє, чи можна оновити атрибут студента. Змінюється група студента "Sofi" на "cs-45", і перевіряється, чи оновився атрибут group для цього студента на нове значення.
* Тест test\_deleteElement: Тест перевіряє, чи можна видалити студента зі списку. Спочатку записується початкова кількість студентів у списку. Потім студент "Sofi" видаляється, і перевіряється, чи зменшилася кількість студентів на 1, а також чи дійсно "Sofi" більше немає в списку.

Результат:



Посилання на GitHub: <https://github.com/sofiiachub/TP-KB-231-Sofiia-Chub>

**Висновки:** У результаті розробки програми для телефонного довідника студентів групи з використанням принципів об'єктно-орієнтованого програмування було створено три основні класи: Student, StudentList та Utils. Всі класи реалізовані в окремих файлах. Тести юніт модернізувала до умов коду. Весь функціонал та робота програми залишилась не змінною і працює правильно.