**Лабораторна робота №3**

**Тема**: ООП.

**Мета**: 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**

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, видалення 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.
6. основний функціонал програми має бути покритий Юніт тестами.

**Хід роботи**

1. **Програма:** Керує списком студентів із можливістю додавання, оновлення, видалення та перегляду. Дані зберігаються у CSV-файлі.
2. **Функції:**

* Student: Клас для створення студентів із методами перетворення в словник і назад.
* StudentList: Клас для керування списком студентів (додавання, оновлення, видалення, пошук, сортування).
* FileManager: Клас для завантаження/збереження студентів у файл.

1. **Тести:**

* Тести для перевірки функцій add\_student, update\_student, delete\_student, find\_student.
* Тести на збереження та завантаження даних через FileManager.

1. **Робота:**

* Завантаження даних із файлу при запуску.
* Вибір дій через меню: створення, оновлення, видалення, друк списку.
* Збереження змін перед виходом.

Текст програми:

|  |
| --- |
| #main.py  import sys  from student\_list import StudentList  from utils import FileManager  from student import Student  def main():  group = StudentList()  file\_manager = FileManager("students.csv")  group.students = file\_manager.load()  while True:  print("\nAvailable actions:")  print("[C] Create, [U] Update, [D] Delete, [P] Print, [X] Exit")  choice = input("Your choice: ").strip().lower()  if choice == 'c':  name = input("Name: ")  phone = input("Phone: ")  email = input("Email: ")  group\_name = input("Group: ")  group.add\_student(Student(name, phone, email, group\_name))  print("Student added successfully.")  elif choice == 'u':  name = input("Name of the student to update: ")  phone = input("New phone (or press Enter to skip): ")  email = input("New email (or press Enter to skip): ")  group\_name = input("New group (or press Enter to skip): ")  if group.update\_student(name, phone, email, group\_name):  print("Student updated successfully.")  else:  print("Student not found.")  elif choice == 'd':  name = input("Name of the student to delete: ")  if group.delete\_student(name):  print("Student deleted successfully.")  else:  print("Student not found.")  elif choice == 'p':  print("List of students:")  group.print\_all()  elif choice == 'x':  file\_manager.save(group.students)  print("Changes saved. Goodbye!")  break  else:  print("Invalid choice, please try again.")  if \_\_name\_\_ == "\_\_main\_\_":  main()  #student.py  class Student:  def \_\_init\_\_(self, name, phone, email, group):  self.name = name  self.phone = phone  self.email = email  self.group = group  def \_\_str\_\_(self):  return f"{self.name} - {self.phone}, {self.email}, {self.group}"  def to\_dict(self):  return {  "name": self.name,  "phone": self.phone,  "email": self.email,  "group": self.group  }  @classmethod  def from\_dict(cls, data):  return cls(data['name'], data['phone'], data['email'], data['group'])  #student\_list.py  from student import Student  class StudentList:  def \_\_init\_\_(self):  self.students = []  def add\_student(self, student):  self.students.append(student)  self.students.sort(key=lambda x: x.name)  def delete\_student(self, name):  for student in self.students:  if student.name == name:  self.students.remove(student)  return True  return False  def update\_student(self, name, phone=None, email=None, group=None):  for student in self.students:  if student.name == name:  if phone:  student.phone = phone  if email:  student.email = email  if group:  student.group = group  return True  return False  def find\_student(self, name):  for student in self.students:  if student.name == name:  return student  return None  def print\_all(self):  for student in self.students:  print(student)  #test\_all.py  import unittest  from student import Student  from student\_list import StudentList  from utils import FileManager  import os  class TestStudentList(unittest.TestCase):  def setUp(self):  self.group = StudentList()  self.group.add\_student(Student("Alice", "123", "alice@example.com", "CS231"))  self.group.add\_student(Student("Bob", "456", "bob@example.com", "CS231"))  def test\_add\_student(self):  self.group.add\_student(Student("Charlie", "789", "charlie@example.com", "CS231"))  self.assertEqual(len(self.group.students), 3)  def test\_delete\_student(self):  self.assertTrue(self.group.delete\_student("Alice"))  self.assertFalse(self.group.delete\_student("Nonexistent"))  def test\_update\_student(self):  self.assertTrue(self.group.update\_student("Alice", phone="111"))  self.assertEqual(self.group.find\_student("Alice").phone, "111")  class TestFileManager(unittest.TestCase):  def setUp(self):  self.filename = "test\_students.csv"  self.file\_manager = FileManager(self.filename)  self.students = [  Student("Alice", "123", "alice@example.com", "CS231"),  Student("Bob", "456", "bob@example.com", "CS231")  ]  self.file\_manager.save(self.students)  def tearDown(self):  if os.path.exists(self.filename):  os.remove(self.filename)  def test\_load(self):  students = self.file\_manager.load()  self.assertEqual(len(students), 2)  self.assertEqual(students[0].name, "Alice")  def test\_save(self):  new\_students = [Student("Charlie", "789", "charlie@example.com", "CS231")]  self.file\_manager.save(new\_students)  students = self.file\_manager.load()  self.assertEqual(len(students), 1)  if \_\_name\_\_ == "\_\_main\_\_":  unittest.main()  #test\_student\_management.py  import unittest  from student import Student  from student\_list import StudentList  from utils import FileManager  class TestStudentManagement(unittest.TestCase):  def setUp(self):  self.student\_list = StudentList()  self.student\_list.add\_student(Student("Alice", "123456789", "alice@example.com", "CS231"))  self.student\_list.add\_student(Student("Bob", "987654321", "bob@example.com", "CS231"))  def test\_add\_student(self):  new\_student = Student("Charlie", "555555555", "charlie@example.com", "CS231")  self.student\_list.add\_student(new\_student)  self.assertEqual(len(self.student\_list.students), 3)  self.assertEqual(self.student\_list.students[2].name, "Charlie")  def test\_update\_student(self):  self.student\_list.update\_student("Alice", phone="111111111", email="alice\_new@example.com")  student = self.student\_list.find\_student("Alice")  self.assertEqual(student.phone, "111111111")  self.assertEqual(student.email, "alice\_new@example.com")  def test\_delete\_student(self):  self.student\_list.delete\_student("Bob")  self.assertEqual(len(self.student\_list.students), 1)  self.assertIsNone(self.student\_list.find\_student("Bob"))  def test\_find\_student(self):  student = self.student\_list.find\_student("Alice")  self.assertIsNotNone(student)  self.assertEqual(student.name, "Alice")  def test\_save\_and\_load(self):  file\_name = "test\_students.csv"  FileManager.save\_to\_csv(self.student\_list, file\_name)  loaded\_list = StudentList()  FileManager.load\_from\_csv(loaded\_list, file\_name)  self.assertEqual(len(loaded\_list.students), 2)  self.assertEqual(loaded\_list.students[0].name, "Alice")  self.assertEqual(loaded\_list.students[1].name, "Bob")  if \_\_name\_\_ == "\_\_main\_\_":  unittest.main()  #utils.py  import csv  from student import Student  class FileManager:  def \_\_init\_\_(self, filename):  self.filename = filename  def load(self):  students = []  try:  with open(self.filename, mode='r', encoding='utf-8') as file:  reader = csv.DictReader(file)  for row in reader:  students.append(Student.from\_dict(row))  except FileNotFoundError:  print(f"File {self.filename} not found. Starting with an empty list.")  return students  def save(self, students):  with open(self.filename, mode='w', encoding='utf-8', newline='') as file:  fieldnames = ["name", "phone", "email", "group"]  writer = csv.DictWriter(file, fieldnames=fieldnames)  writer.writeheader()  writer.writerows([student.to\_dict() for student in students]) |

**Висновок:** Під час виконання роботи використовуючи теоретичне підґрунтя про ООП у мові Python перероблено програму телефонного довідника студентів використовуючи принципи ООП для формування відомостей про студентів.