Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №4**

Дисциплина: «Разработка профессиональных приложений»

Классы

Выполнил:

студент группы ИВТАПбд-21

Карпова А.С.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

**Задание по варианту**

Необходимо переделать лабораторную работу №3 с использованием классов, описывающих предметную область, заданную вариантом, с реализацией следующих особенностей (вполне возможно, что предлагаемое в 3 лабе задание для этого нужно будет расширить):

**Требования к программе:**

1. Класс должен содержать итератор
2. Должна быть реализована перегрузка стандартных операций (repr, например)
3. Должно быть реализовано наследование
4. Запись значений в свойства - только через \_\_setattr\_\_
5. Возможность доступа к элементам коллекции по индексу (\_\_getitem\_\_)
6. Должны быть реализованы статические методы
7. Должны быть реализованы генераторы

**Описание реализации**

Данный код является реализацией программы для работы с данными CSV-файла. Она позволяет считывать данные из файла, выводить их на экран и выполнять действия с данными.

Класс row является базовым классом для всех моделей (записей) данных. У него есть атрибут idx, который представляет индекс записи в списке. Методы get\_idx() и set\_idx() позволяют получить и изменить значение этого атрибута.

Класс model представляет модель данных, которая соответствует одной записи CSV-файла. Он наследуется от базового класса row и имеет свои атрибуты: number, date, fio, stip и kuda. Кроме того, у класса есть методы str() и repr(), которые возвращают строковое представление объекта и строку, представляющую его в коде соответственно. Атрибут dict используется для хранения данных.

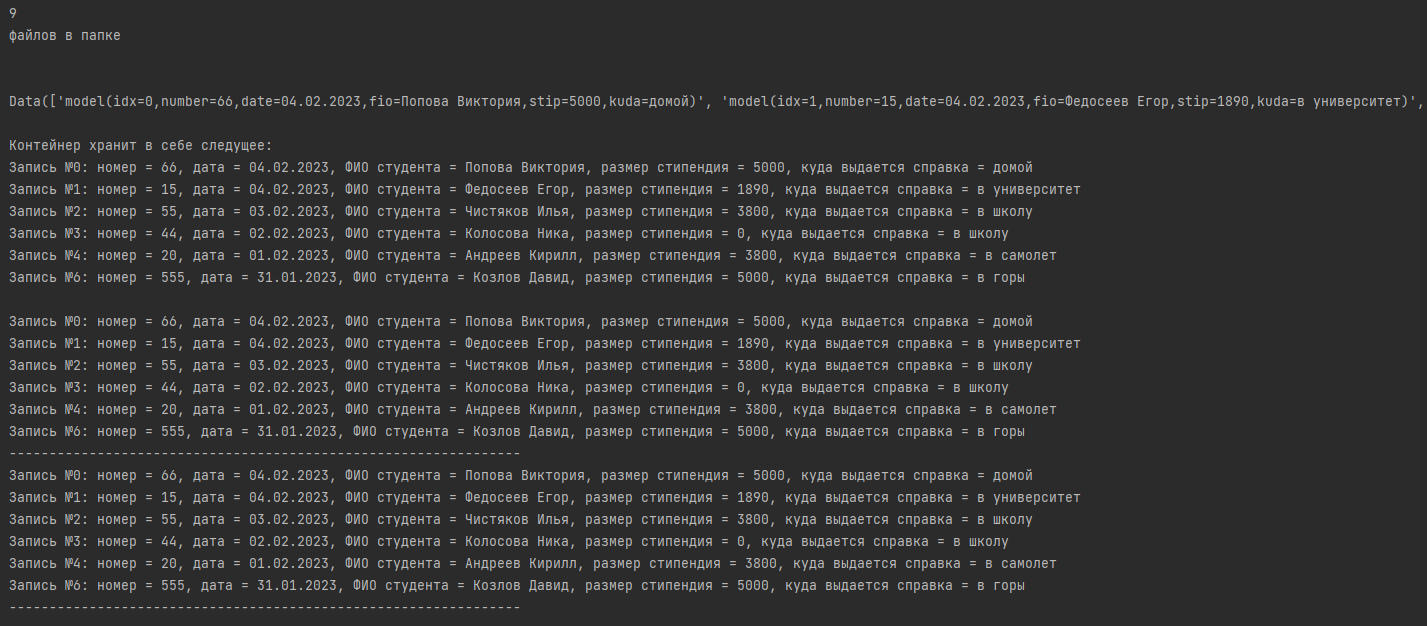
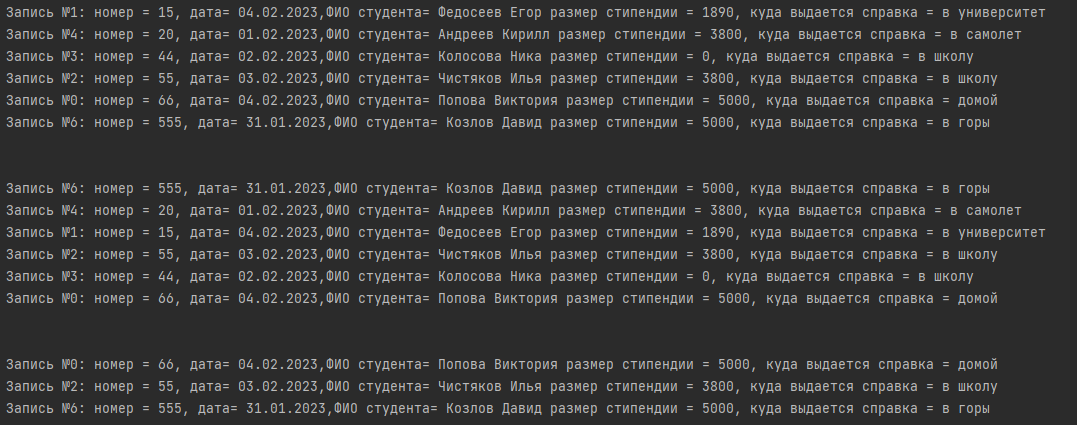
Класс Data представляет данные CSV-файла и выполняет действия с данными. У него есть атрибуты file\_path и data, которые представляют путь к файлу и список записей CSV-файла соответственно. Метод init() инициализирует эти атрибуты, вызывая метод parse(), который считывает данные из файла и создает объекты класса model. Методы print(), str(), repr(), sorted\_by\_kuda(), sorted\_by\_number(), value() и add\_new() выполняют различные действия с данными или преобразовывают их в различные форматы. Метод as\_generator() преобразует список данных в генератор. Метод next() и функция iter() обеспечивают итерационный доступ к данным. Метод getitem() позволяет получить доступ к элементу списка по индексу.

Функция save() используется для записи новых данных обратно в CSV-файл. Она вызывается в методе add\_new().

В основном блоке программы создается объект класса Data, который инициализируется путем к файлу. Затем на экран выводятся данные с помощью методов print(), printd() и sorted\_by\_number(), sorted\_by\_kuda(), value().

**Описание возникших затруднений**

Затруднений не возникло.

**Результаты работы программы***Рис. 1.1**Рис. 1.2*

**Код программы**

import os  
lst = os.listdir("D://papka") # your directory path  
number\_files = len(lst)  
print("\n")  
print (number\_files)  
print("файлов в папке")  
print("\n")  
  
class row():  
 idx = 0  
  
 def \_\_init\_\_(self, idx: int):  
 self.idx = idx  
  
 def get\_idx(self):  
 return self.idx  
  
 def set\_idx(self, val):  
 self.idx = val  
  
class model(row):  
 idx = 0  
 number = 0  
 date = ""  
 fio = ""  
 stip = ""  
 kuda = ""  
  
 def \_\_init\_\_(self, idx: int, number: int, date: str, fio: str, stip: int, kuda: str):  
 super().\_\_init\_\_(idx)  
 self.idx = idx  
 self.number = number  
 self.date = date  
 self.fio = fio  
 self.stip = stip  
 self.kuda = kuda  
  
 def \_\_str\_\_(self):  
 return f"Запись №{self.idx}: номер = {self.number}, дата = {self.date}, ФИО студента = {self.fio}, размер стипендия = {self.stip}, куда выдается справка = {self.kuda}"  
  
 def \_\_repr\_\_(self):  
 return f"model(idx={self.idx},number={self.number},date={self.date},fio={self.fio},stip={self.stip},kuda={self.kuda})"  
  
 def \_\_setattr\_\_(self, \_\_name, \_\_value):  
 self.\_\_dict\_\_[\_\_name] = \_\_value  
class Data():  
 file\_path = ""  
 data = {}  
 pointer = 0  
  
 def \_\_init\_\_(self, file):  
 self.file\_path = file  
 self.data = self.parse(file)  
  
 def \_\_str\_\_(self):  
 d\_str = '\n'.join([str(rm) for rm in self.data])  
 return f"Контейнер хранит в себе следущее:\n{d\_str}"  
  
 def \_\_repr\_\_(self):  
 return f"Data({[repr(rm) for rm in self.data]})"  
  
 def \_\_iter\_\_(self):  
 return self  
  
 def \_\_next\_\_(self):  
 if self.pointer >= len(self.data):  
 self.pointer = 0  
 raise StopIteration  
 else:  
 self.pointer += 1  
 return self.data[self.pointer - 1]  
  
 def \_\_getitem\_\_(self, item):  
 if not isinstance(item, int):  
 raise TypeError("Индекс должен быть целым числом.")  
 if 0 <= item < len(self.data):  
 return self.data[item]  
 else:  
 raise IndexError("Неверный индекс.")  
  
 def as\_generator(self):  
 self.pointer = 0  
 while self.pointer < len(self.data):  
 yield self.data[self.pointer]  
 self.pointer += 1  
  
 @staticmethod  
 def parse(file):  
 parsed = []  
 with open(file, "r", encoding='utf-8') as raw\_csv:  
 for line in raw\_csv:  
 (idx, number, date, fio, stip, kuda) = line.replace("\n", "").split(",")  
 parsed.append(model(int(idx), int(number), date, fio, stip, kuda))  
 return parsed  
  
 def sorted\_by\_kuda(self):  
 return sorted(self.data, key=lambda f: f.kuda)  
  
 def sorted\_by\_number(self):  
 return sorted(self.data, key=lambda f: f.number)  
  
 def value(self, value):  
 r = []  
 for d in self.data:  
 if (d.number > value):  
 r.append(d)  
 return r  
  
 def add\_new(self, number, date,fio, stip, kuda):  
 self.data.append(model(len(self.data) + 1, number, date,fio, stip, kuda))  
 self.save(self.file\_path, self.data)  
  
 @staticmethod  
 def save(file, new\_data):  
 with open(file, "w", encoding='utf-8') as f:  
 for r in new\_data:  
 f.write(f"{r.idx},{r.number},{r.date},{r.fio},{r.stip},{r.kuda}\n")  
  
 def print(self):  
 for r in self.data:  
 print(f"Запись №{r.idx}: номер = {r.number}, дата= {r.date},ФИО студента= {r.fio}, размер стипендии = {r.stip}, куда выдается справка = {r.kuda}")  
  
 def printd(self, d):  
 for r in d:  
 print(f"Запись №{r.idx}: номер = {r.number}, дата= {r.date},ФИО студента= {r.fio} размер стипендии = {r.stip}, куда выдается справка = {r.kuda}")  
  
def get\_files\_count\_in\_directory(file):  
 (loc, dirs, files) = next(os.walk(file))  
 return len(files)  
  
data = Data("D:\\papka\\data1.csv.txt")  
  
# \_\_repr\_\_()  
print(repr(data), "\n")  
  
# \_\_str\_\_()  
print(data, "\n")  
  
# Итератор  
for item in iter(data):  
 print(item)  
  
print("-" \* 64)  
  
# Генератор  
for item in data.as\_generator():  
 print(item)  
  
print("-" \* 64)  
  
  
print("\n")  
data.printd(data.sorted\_by\_number())#сортировка по номеру  
print("\n")  
data.printd(data.sorted\_by\_kuda())#сортировка по имени  
print("\n")  
data.printd(data.value(50))#номер больше 50