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**

Вариант №6

Выполнил:

студент группы ИВТАСбд-22

Зяббаров Артур

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

Ульяновск, 2023

1. Задание по варианту

Необходимо переделать лабораторную работу №3 с использованием классов, описывающих предметную область, заданную вариантом, с реализацией следующих особенностей:

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

1. Описание реализации

Для описания ряда данных был создан класс RawModel. Данный класс хранил поля для данной предметной области.

Запись и чтение данных реализована с помощью класса Data. Этот класс также имеет функции для сортировки и выбора данных. Кроме того, он позволяет выводить данные с помощью переопределенного метода \_\_str\_\_.

1. Описание возникших затруднений

Трудности возникли с наследованием, а точнее что придумать в рамках этой лабораторной работы, а также с тем, как работает генератор и чем оно отличается от итератора.

1. Приложение

import os.path

path = 'C:\\Users\\opaf5\\OneDrive\\Рабочий стол\\lab3'

c=0

for i in os.listdir(path):

if os.path.exists(path) == True:

c=c+1

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 RawModel(Row):

idx = 0

number = 0

date = ""

time = ""

plateNumber = ""

carModel = ""

def \_\_init\_\_(self, idx: int, number: int, date: str, time: str, plateNumber: str, carModel: str):

super().\_\_init\_\_(idx)

self.idx = idx

self.number = number

self.date = date

self.time = time

self.plateNumber = plateNumber

self.carModel = carModel

def \_\_str\_\_(self):

return f"Запись №{self.idx}: номер = {self.number}, дата = {self.date}, время = {self.time},номерной знак = {self.plateNumber}, марка автомобиля = {self.carModel}"

def \_\_repr\_\_(self):

return f"RowModel(idx={self.idx},number={self.number},date={self.date},time={self.time},plateNumber={self.plateNumber},carModel={self.carModel})"

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, time, plateNumber, carModel) = line.replace("\n", "").split(",")

parsed.append(RawModel(int(idx), int(number), date, time, plateNumber, carModel))

return parsed

def sorted\_by\_carModel(self):

return sorted(self.data, key=lambda f: f.carModel)

def sorted\_by\_number(self):

return sorted(self.data, key=lambda f: f.number)

def select\_more\_than(self, value):

r = []

for d in self.data:

if (d.number > value):

r.append(d)

return r

def add\_new(self, number, date,time, plateNumber, carModel):

self.data.append(RawModel(len(self.data) + 1, number, date,time, plateNumber, carModel))

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.time},{r.plateNumber},{r.carModel}\n")

def print(self):

for r in self.data:

print(f"Запись №{r.idx}: номер = {r.number}, дата= {r.date},время= {r.time}, номерной знак = {r.plateNumber}, марка автомобиля = {r.carModel}")

def printd(self, d):

for r in d:

print(f"Запись №{r.idx}: номер = {r.number}, дата= {r.date},время= {r.time} номерной знак = {r.plateNumber}, марка автомобиля = {r.carModel}")

def get\_files\_count\_in\_directory(file):

(loc, dirs, files) = next(os.walk(file))

return len(files)

data = Data("C:\\Users\\opaf5\\OneDrive\\Рабочий стол\\lab3\\data2.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("\nСортировка по модели: ")

data.printd(data.sorted\_by\_carModel())#сортировка по имени

print("\nОтсортировано по номеру: ")

data.printd(data.sorted\_by\_number())#сортировка по номеру

print("\nТолько строки, у которых номер больше 56: ")

data.printd(data.select\_more\_than(56))

print(c)

# Добавление новых данных

#data.add\_new(355, "15.05.2021","00:00", "919", "Камаз")