Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ульяновский государственный технический университет»

Кафедра: «Вычислительная техника»

Лабораторная работа №4

Дисциплина: «Разработка профессиональных приложений»

«Классы»

Вариант 22

Выполнил

Студент группы ИВТАСбд-22

Хабибуллин Х. Р.

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):

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

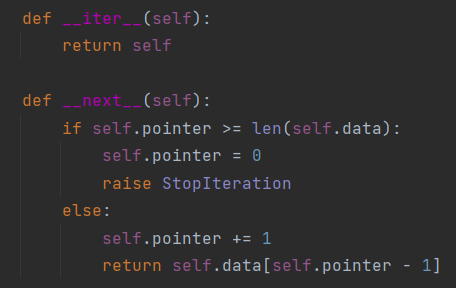
7. Должны быть реализованы генераторы

**Описание реализации**

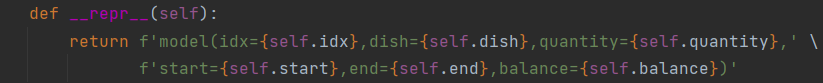
Для описания ряда данных был создан класс Model. Данный класс хранил поля для данной предметной области. Он наследует класс Row.

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\_\_. Также в этом классе были реализованы статические методы.

*Листинг 1. Итератор.*



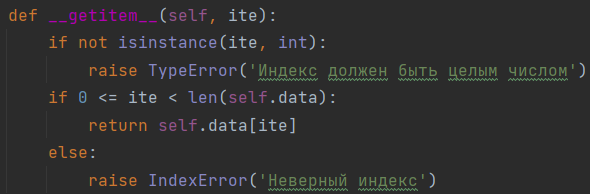
*Листинг 2. Перегрузка.*



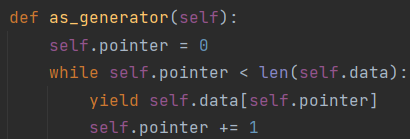
*Листинг 3. Запись значений в свойства.*



*Листинг 4. Доступ по индексу.*



*Листинг 5. Генератор.*

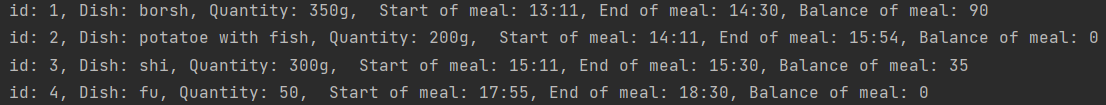


**Описание возникших затруднений**

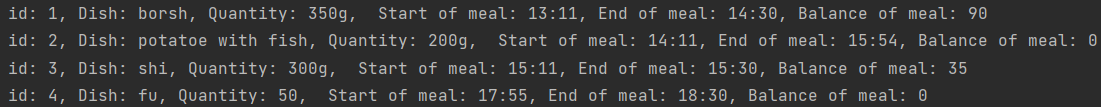
При выполнении работы затруднений не возникло.

**Тесты**

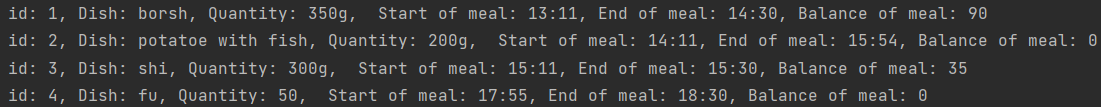
Вывод str:



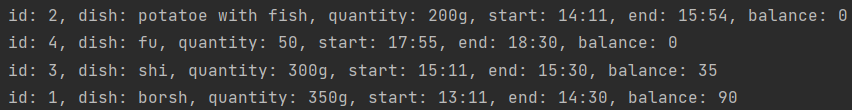
Вывод итератора:



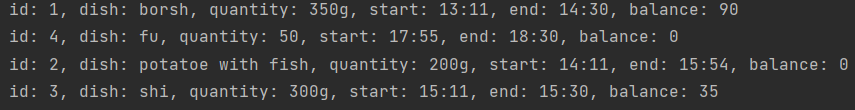
Вывод генератора:



Сортировка по атрибуту баланса:



Сортировка по тексту:



Сортировка по условию(номер id больше 3):



**Выводы о проделанной работе**

Выполняя данную работу, мы научились работать с классами, поняли основные принципы ООП.

**Приложение**

Полный код программы:

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, dish, quantity, start, end, balance = 0, '', '', '', '', 0

def \_\_init\_\_(self, idx: int, dish: str, quantity: str, start: str, end: str, balance: int):

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

self.idx = idx

self.dish = dish

self.quantity = quantity

self.start = start

self.end = end

self.balance = balance

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

return f'id: {self.idx}, Dish: {self.dish}, Quantity: {self.quantity}, ' \

f' Start of meal: {self.start}, End of meal: {self.end}, Balance of meal: {self.balance}'

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

return f'model(idx={self.idx},dish={self.dish},quantity={self.quantity},' \

f'start={self.start},end={self.end},balance={self.balance})'

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, ite):

if not isinstance(ite, int):

raise TypeError('Индекс должен быть целым числом')

if 0 <= ite < len(self.data):

return self.data[ite]

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") as raw\_csv:

for line in raw\_csv:

(idx, dish, quantity, start, end, balance) = line.replace("\n", "").split(";")

parsed.append(Model(int(idx), str(dish), str(quantity), str(start), str(end), str(balance)))

return parsed

def sorted\_by\_str(self):

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

def sorted\_by\_number(self):

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

def value(self, value):

r = []

for d in self.data:

if d.idx > value:

r.append(d)

return r

def add\_record(self, dish, quantity, start, end, balance):

self.data.append(Model(len(self.data) + 1, dish, quantity, start, end, balance))

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.dish},{r.quantity},{r.start},{r.end},{r.balance}\n")

def print(self):

for r in self.data:

print(f'id: {r.idx}, dish: {r.dish}, quantity: {r.quantity}, '

f'start: {r.start}, end: {r.end}, balance: {r.balance}')

@staticmethod

def print\_d(d):

for r in d:

print(f'id: {r.idx}, dish: {r.dish}, quantity: {r.quantity}, '

f'start: {r.start}, end: {r.end}, balance: {r.balance}')

data = Data("data1.csv")

# \_\_repr\_\_()

#print(repr(data), "/n")

# \_\_str\_\_()

#print(data, "\n")

# Итератор

#for item in iter(data):

# print(item)

#print('')

# Генератор

#for item in data.as\_generator():

# print(item)

#print('')

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

##print('')

#data.print\_d(data.sorted\_by\_str()) # сортировка по имени

#print('')

data.print\_d(data.value(3)) # номер больше 3