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 №5

Выполнил:

студент группы ИВТАСбд-22

Зейнетдинов М.М.

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:

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 представляет собой реализацию таблицы, которая содержит классы Row и MyTable.

Класс Row является наследником класса TemplateRow. Он содержит методы работы со строками таблицы, такие как их добавление, удаление, сортировка, поиск и т.д. Также класс Row имеет стандартную реализацию перегрузки операторов (repr, str), что упрощает работу с объектами этого класса. Доступ к свойствам объекта контролируется через метод setattr и метод getattr.

Класс MyTable описывает саму таблицу, которая содержит объекты класса Row. Класс MyTable имеет методы для работы с таблицей в целом: добавление строк, удаление строк, сортировка и т.д. Кроме того, класс MyTable реализует методы итерации через объекты класса Row (с помощью iter и next) и доступ по индексу(getitem).

Код также содержит примеры использования всех реализованных функций - например, сортировка и фильтрация данных из файла input.csv, изменение значения поля cost в строке таблицы, сохранение таблицы в файл и т.д.

**Описание возникших затруднений**

Организация связей между классами: для того, чтобы строка таблицы могла быть членом таблицы и использовать ее методы, необходимо организовать связь между этими классами.

Разработка тестов: для проверки корректности работы классов необходимо было разработать соответствующие тесты, которые покрывают все возможные сценарии использования классов.

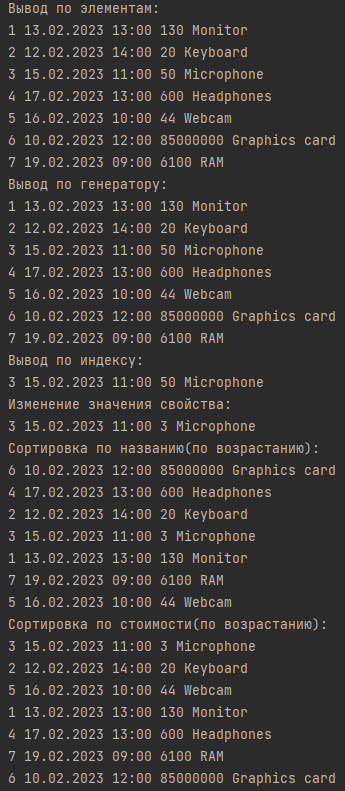
Описание альтернативных способов решения

Использование функционального программирования вместо объектно-ориентированного подхода, чтобы уменьшить количество кода и повысить читаемость.

Добавление обработки ошибок при выполнении операций с таблицей, таких как удаление несуществующей строки или неверный формат файла с данными.

Разделение методов класса MyTable на более мелкие и специализированные методы для упрощения поддержки и тестирования кода. Применение контекстных менеджеров для безопасного открытия и закрытия файлов с данными.

Тестирование



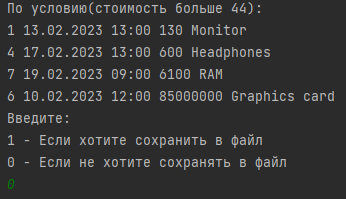


Рис1. Пример работы программы

Вывод

В данной лабораторной работе были приобретены базовые навыки объектно-ориентированного программирования на языке Python, в частности в работе со таблицами, вводом и выводом файлов и преобразованием типов данных.

Код программы

class TemplateRow:

def \_\_init\_\_(self, num):

self.num = num

def get\_num(self):

return self.num

def set\_num(self, val):

self.num = val

class Row(TemplateRow):

def \_\_init\_\_(self, temp\_arr):

super().\_\_init\_\_(int(temp\_arr[0]))

(self.date, self.cost, self.name) = temp\_arr[1], int(temp\_arr[2]), temp\_arr[3]

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

return str(self.num) + ' ' + self.date + ' ' + str(self.cost) + ' ' + self.name

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

return 'NUM: ' + str(self.num) + ' DATE: ' + self.date + ' COST: ' + str(self.cost) + ' NAME: ' + self.name

def \_\_setattr\_\_(self, \_\_name, \_\_value):

self.\_\_dict\_\_[\_\_name] = \_\_value

def \_\_getitem\_\_(self, \_\_name):

return self.\_\_dict\_\_[\_\_name]

class MyTable:

def \_\_init\_\_(self, file\_path=None):

self.index = 0

if file\_path is not None:

self.table = self.parse(file\_path)

else:

self.table = []

def \_\_iter\_\_(self):

return self

def \_\_next\_\_(self):

if self.index >= len(self.table):

self.index = 0

raise StopIteration

else:

iter\_row = self.table[self.index]

self.index += 1

return iter\_row

def \_\_getitem\_\_(self, id\_row):

return self.table[id\_row]

def add\_row(self, row):

if len(self.table) == 0:

self.table.append(Row([0] + row))

else:

self.table.append(Row([len(self.table) + 1] + row))

def delete\_row(self, row\_num):

self.table.remove(self.table[row\_num])

def sort\_table(self, name, is\_reversed):

temp\_list = MyTable()

temp\_list.table = sorted(self.table, key=lambda f: f[name], reverse=is\_reversed)

return temp\_list

def if\_list(self, data, op, value):

temp\_list = MyTable()

for elem in self.table:

if op == '>':

if int(elem[data]) > value:

temp\_list.table.append(elem)

elif op == '<':

if int(elem[data]) < value:

temp\_list.table.append(elem)

elif op == '==':

if int(elem[data]) == value:

temp\_list.table.append(elem)

elif op == '!=':

if int(elem[data]) != value:

temp\_list.table.append(elem)

elif op == '>=':

if int(elem[data]) >= value:

temp\_list.table.append(elem)

elif op == '<=':

if int(elem[data]) <= value:

temp\_list.table.append(elem)

return temp\_list

def generator(self):

self.index = 0

while self.index < len(self.table):

yield table[self.index]

self.index += 1

else:

self.index = 0

@staticmethod

def parse(file\_path):

temp\_table = []

with open(file\_path, 'r') as file:

for line in file:

temp\_row = Row(line.replace('\n', '').split(';'))

temp\_table.append(temp\_row)

return temp\_table

@staticmethod

def save(file\_path):

with open(file\_path, 'w') as file:

for row in table:

file.write(';'.join([str(row.num), row.date, str(row.cost), row.name]))

file.write('\n')

if \_\_name\_\_ == '\_\_main\_\_':

table = MyTable('input.csv')

print("Вывод по элементам:")

for x in table:

print(x)

print("Вывод по генератору:")

for x in table.generator():

print(x)

print("Вывод по индексу:")

print(table[2])

print("Изменение значения свойства:")

table.table[2].cost=3

print(table[2])

print("Сортировка по названию(по возрастанию):")

table = table.sort\_table('name', 0)

for x in table:

print(x)

print("Сортировка по стоимости(по возрастанию):")

table = table.sort\_table('cost', 0)

for x in table:

print(x)

print("По условию(стоимость больше 44):")

table = table.if\_list('cost', '>', 44)

for x in table:

print(x)

table.sort\_table('num', 0)

print("Введите:\n1 - Если хотите сохранить в файл\n0 - Если не хотите сохранять в файл")

check = int(input())

if check == 1:

print("Введите путь к файлу")

table.save(input())

else:

exit()