Московский государственный технический
университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Отчет по рубежному контролю №1

Выполнил:
студент группы ИУ5-53Б

Баркалова Ирина

Дата: 25.09.2021

Москва. 2021 г.

Вариант РК:
Вариант В.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

4	Компьютер	Дисплейный класс

Код программы:

classes.py:

```
class Display:

    def __init__(self, id: int, matrix_type: str, diagonal: float):
        self.id = id
        self.matrix_type = matrix_type
        self.diagonal = diagonal

    def __str__(self):
        return "Processor with id: " + str(self.id) + "; matrix_type: " + self.matrix_type +
"; diagonal: " + str(
            self.diagonal)

    def __repr__(self) -> str:
        return self.__str__()

    def values(self):
        return self.matrix_type, self.diagonal


class PC:

    def __init__(self, id: int, name: str, price: int, disp_id: int):
        self.id = id
        self.name = name
        self.price = price
        self.disp_id = disp_id

    def __str__(self):
        return "PC with id: " + str(self.id) + "; name: " + self.name + "; price: " + str(
            self.price) + "; disp_id: " + str(self.disp_id)

    def __repr__(self) -> str:
        return self.__str__()

    def values(self):
        return self.name, self.price


class PC_Display:
    def __init__(self, disp_id: int, pc_id: int):
        self.disp_id = disp_id
```

```
        self.pc_id = pc_id

    def __repr__(self) -> str:
        return self.__str__()

    def values(self):
        return self.disp_id, self.pc_id
```

source.py:

```
from classes import PC, Display, PC_Display

list_of_display = [Display(1, "IPS", 13),
                   Display(2, "TN", 14),
                   Display(3, "VA", 15.6),
                   Display(4, "VA", 24),
                   Display(5, "IPS", 12),
                   Display(6, "TN", 13),
                   Display(7, "TN", 17),
                   Display(8, "VA", 15.6),
                   Display(9, "IPS", 24),
                   Display(10, "IPS", 17)]

list_of_pc = [PC(1, "Lenovo IdeaCentre G5 14IMB05", 88000, 8),
              PC(2, "HP Pavilion Gaming TG01", 110000, 1),
              PC(3, "ZOTAC MAGNUS ONE", 187000, 7),
              PC(4, "Apple Mac mini 2020 M1", 173000, 6),
              PC(5, "Acer Aspire TC-895", 34000, 7),
              PC(6, "Apple iMac 24", 248000, 1),
              PC(7, "Gigabyte GB-BR", 93000, 4),
              PC(8, "HP M01", 61000, 3)]

list_of_pc_display = [PC_Display(1, 8),
                      PC_Display(2, 7),
                      PC_Display(3, 6),
                      PC_Display(4, 5),
                      PC_Display(5, 4),
                      PC_Display(6, 3),
                      PC_Display(7, 2),
                      PC_Display(8, 1),
                      PC_Display(9, 1),
                      PC_Display(10, 7)]
```

utils.py:

```
def print_pretty_table(data, cell_sep=' | ', header_separator=True):
    rows = len(data)
    cols = len(data[0])
    col_width = []

    for col in range(cols):
        columns = [str(data[row][col]) for row in range(rows)]
        col_width.append(len(max(columns, key=len)))

    separator = "-+-".join('-' * n for n in col_width)
    for i, row in enumerate(range(rows)):
        if i == 1 and header_separator:
            print(separator)

        result = []
        for col in range(cols):
            item = str(data[row][col]).rjust(col_width[col])
            result.append(item)
        print(cell_sep.join(result))


max_int = 1000000000000


def change_elem(x, new_val, id):
    x[id] = new_val
    return x
```

request_1.py:

```
from source import list_of_display, list_of_pc
import utils

a = [i for i in range(10)]
startLetter = "A"
correct_pc = list(filter(lambda x: x.name.startswith(startLetter), list_of_pc))
id_display = list(map(lambda x: x.disp_id, correct_pc))
correct_display = list(map(lambda x: (x.matrix_type, x.id), filter(lambda x: x.id in
id_display, list_of_display)))

display_by_id = {}
for (matrix_type, disp_id) in correct_display:
    display_by_id[disp_id] = matrix_type

results = [["PC name", "price", "matrix type"]]
for pc in correct_pc:
    tmp_arr = []
    tmp_arr.extend(pc.values())
    if pc.disp_id in display_by_id:
        tmp_arr.append(display_by_id[pc.disp_id])
    else:
        tmp_arr.append("--------------------")
    results.append(tmp_arr)

utils.print_pretty_table(results)
```

request_2.py:

```
from source import list_of_display, list_of_pc
import utils


id_display = list(map(lambda x: x.id, list_of_display))
prices = list(map(lambda x: (x.price, x.disp_id), filter(lambda x: x.disp_id in id_display,
list_of_pc)))

correct_price_by_id = {}
for (price, disp_id) in prices:
    if disp_id not in correct_price_by_id or correct_price_by_id[disp_id] > price:
        correct_price_by_id[disp_id] = price

results = []
for display in list_of_display:
    tmp_arr = []
    tmp_arr.extend(display.values())
    if display.id in correct_price_by_id:
        tmp_arr.append(correct_price_by_id[display.id])
    else:
        tmp_arr.append(utils.max_int)
    results.append(tmp_arr)

results = [["matrix type", "diagonal", "price"]] + list(map(lambda x: utils.change_elem(x, "--
---------------", 2) if x[2] == utils.max_int else x,
                                                            sorted(results,
key=lambda x: x[2])))

utils.print_pretty_table(results)
```

request_3.py:

```
from source import list_of_display, list_of_pc, list_of_pc_display
import utils

id_display = list(map(lambda x: x.id, list_of_display))
pcs = list(filter(lambda x: x.disp_id in id_display, list_of_pc))

pcs_by_id = {}
for pc in pcs:
```

```
    if pc.disp_id in pcs_by_id:
        pcs_by_id[pc.disp_id].append(pc)
    else:
        pcs_by_id[pc.disp_id] = [pc]

for key in pcs_by_id.keys():
    pcs_by_id[key] = sorted(pcs_by_id[key], key=lambda x: x.name)

results = [["matrix type", "diagonal", "PC"]]
for processor in list_of_display:
    tmp_arr = []
    tmp_arr.extend(processor.values())
    if processor.id in pcs_by_id:
        tmp_arr.append(pcs_by_id[processor.id])
    else:
        tmp_arr.append("-------------")
    results.append(tmp_arr)

utils.print_pretty_table(results)
```

Результаты работы программы:

Request_1:

PC name	price	matrix type
Apple Mac mini 2020 M1	173000	TN
Acer Aspire TC-895	34000	TN
Apple iMac 24	248000	IPS

Request_2:

matrix type	diagonal	price
TN	17	34000
VA	15.6	61000
VA	15.6	88000
VA	24	93000
IPS	13	110000
TN	13	173000
TN	14	-----------------
IPS	12	-----------------
IPS	24	-----------------
IPS	17	-----------------

Request_3:

matrix type	diagonal	PC
IPS	13	[PC with id: 6; name: Apple iMac 24; price: 248000; disp_id: 1, PC with id: 2; name: HP Pavilion Gaming TG01; price: 110000; disp_id: 1]
TN	14	-------------
VA	15.6	[PC with id: 8; name: HP M01; price: 61000; disp_id: 3]
VA	24	[PC with id: 7; name: Gigabyte GB-BR; price: 93000; disp_id: 4]
IPS	12	-------------
TN	13	[PC with id: 4; name: Apple Mac mini 2020 M1; price: 173000; disp_id: 6]
TN	17	[PC with id: 5; name: Acer Aspire TC-895; price: 34000; disp_id: 7, PC with id: 3; name: ZOTAC MAGNUS ONE; price: 187000; disp_id: 7]
VA	15.6	[PC with id: 1; name: Lenovo IdeaCentre G5 14IMB05; price: 88000; disp_id: 8]
IPS	24	-------------
IPS	17	-------------