

**Московский государственный технический
университет им. Н.Э. Баумана**

**Факультет “Радиотехнический”
Кафедра “Системы обработки информации и управления”**

Курс «Парадигмы и конструкции языка»

Отчет по рубежному контролю 1

Выполнил:
студент группы РТ5-31Б:
Зеленева И.Е.

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва, 2025 г.

Описание задания

Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим. Пример классов данных для предметной области Сотрудник-Отдел: 1. Класс «Сотрудник», содержащий поля:

- ID записи о сотруднике;
- Фамилия сотрудника;
- Зарплата (количественный признак);
- ID записи об отделе. (для реализации связи один-ко-многим) 2. Класс «Отдел», содержащий поля:

- ID записи об отделе;
- Наименование отдела. 3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:

- ID записи о сотруднике;
- ID записи об отделе. 2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам. 3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков). Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результаты ее выполнения.

Вариант Е.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате. Средняя зарплата должна быть округлена до 2 знака после запятой (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений; для округления необходимо использовать функцию <https://docs.python.org/3/library/functions.html#round>*).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.

| | | |
|---|--------------|-----------|
| 8 | Жесткий диск | Компьютер |
|---|--------------|-----------|

Текст программы:

```
class HardDisk:
```

```
    """Жесткий диск"""
```

```
    def __init__(self, id, name, capacity, price, computer_id):
```

```
        self.id = id
```

```
        self.name = name
```

```
        self.capacity = capacity
```

```
        self.price = price
```

```
        self.computer_id = computer_id
```

```
class Computer:
```

```
    """Компьютер"""
```

```
    def __init__(self, id, name):
```

```
        self.id = id
```

```
        self.name = name
```

```
class HardDiskComputer:
```

```
    """Для реализации связи многие-ко-многим"""
```

```
    def __init__(self, computer_id, hard_disk_id):
```

```
        self.computer_id = computer_id
```

```
        self.hard_disk_id = hard_disk_id
```

```

def main():
    computers = [
        Computer(1, "Домашний компьютер"),
        Computer(2, "Рабочий"),
        Computer(3, "Игровой компьютер"),
        Computer(4, "Офисный"),
        Computer(5, "Серверный компьютер"),
        Computer(6, "Сломанный")
    ]
    hard_disks = [
        HardDisk(1, "Барракуда 1TB", 1000, 45.90, 1),
        HardDisk(2, "Восток 500GB", 500, 25.50, 2),
        HardDisk(3, "Север 2TB", 2000, 85.00, 3),
        HardDisk(4, "WD Blue 1TB", 1000, 42.80, 3),
        HardDisk(5, "Samsung 870 2TB", 2000, 92.75, 3),
        HardDisk(6, "Toshiba 500GB", 500, 28.30, 4),
        HardDisk(7, "Hitachi 1TB", 1000, 47.50, 5),
        HardDisk(8, "Kingston 2TB", 2000, 88.00, 6)
    ]
    # Связи многие-ко-многим
    hard_disks_computers = [
        HardDiskComputer(1, 1),
        HardDiskComputer(2, 2),
        HardDiskComputer(3, 3),
        HardDiskComputer(3, 4),
        HardDiskComputer(3, 5),
        HardDiskComputer(4, 6),
        HardDiskComputer(5, 7),
        HardDiskComputer(6, 3),
        HardDiskComputer(6, 4),
        HardDiskComputer(6, 5)
    ]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, hdc.computer_id, hdc.hard_disk_id)

```

```
for c in computers
    for hdc in hard_disks_computers
        if c.id == hdc.computer_id]
```

```
many_to_many = [(hd.name, hd.capacity, hd.price, comp_name)
                 for comp_name, comp_id, hd_id in many_to_many_temp
                 for hd in hard_disks if hd.id == hd_id]
```

```
print('Задание E1')
print('Список всех компьютеров, у которых в названии присутствует слово
"компьютер", и список их жестких дисков:')
```

```
res1 = {}
for c in computers:
    if 'компьютер' in c.name.lower():
        c_hard_disks = list(filter(lambda i: i[3] == c.name, many_to_many))
        if c_hard_disks:
            hard_disk_names = [hd[0] for hd in c_hard_disks]
            res1[c.name] = hard_disk_names
```

```
for computer_name, hard_disks_list in res1.items():
    print(f'\nКомпьютер: {computer_name}')
    for hd_name in hard_disks_list:
        print(f' Жесткий диск: {hd_name}')
```

```
print('\nЗадание E2')
print('Список компьютеров со средней ценой жестких дисков в каждом,
отсортированный по средней цене:')
```

```
res2_unsorted = []
for c in computers:
    c_hard_disks = list(filter(lambda i: i[3] == c.name, many_to_many))
    if c_hard_disks:
        c_prices = [price for _, _, price, _ in c_hard_disks]
```

```

    avg_price = round(sum(c_prices) / len(c_prices), 2)
    res2_unsorted.append((c.name, avg_price, len(c_hard_disks)))
res2 = sorted(res2_unsorted, key=lambda x: x[1])

for computer_name, avg_price, count in res2:
    print(f'Компьютер: {computer_name}, Средняя цена: {avg_price} руб.
(дисков: {count}))')

print('\nЗадание E3')
print('Список всех жестких дисков, у которых название начинается с буквы
"Б", и названия их компьютеров:')

res3 = []
for hd in hard_disks:
    if hd.name.startswith('Б'):
        hd_computers = list(filter(lambda i: i[0] == hd.name, many_to_many))
        for hd_comp in hd_computers:
            res3.append((hd.name, hd.price, hd_comp[3]))

res3_sorted = sorted(res3, key=lambda x: x[0])

for hd_name, hd_price, comp_name in res3_sorted:
    print(f'Жесткий диск: {hd_name}, Цена: {hd_price} руб. -> Компьютер:
{comp_name}')

if __name__ == '__main__':
    main()

```

```
C:\Users\User\Repository\pikyp zeleneva_i\rk1\.venv\Scripts\python.exe C:\Users\User\Repository\pikyp zeleneva_i\rk1\main.py
```

Задание E1

Список всех компьютеров, у которых в названии присутствует слово "компьютер", и список их жестких дисков:

Компьютер: Домашний компьютер

Жесткий диск: Барракуда 1TB

Компьютер: Игровой компьютер

Жесткий диск: Север 2TB

Жесткий диск: WD Blue 1TB

Жесткий диск: Samsung 870 2TB

Компьютер: Серверный компьютер

Жесткий диск: Hitachi 1TB

Задание E2

Список компьютеров со средней ценой жестких дисков в каждом, отсортированный по средней цене:

Компьютер: Рабочий, Средняя цена: 25.5 руб. (дисков: 1)

Компьютер: Офисный, Средняя цена: 28.3 руб. (дисков: 1)

Компьютер: Домашний компьютер, Средняя цена: 45.9 руб. (дисков: 1)

Компьютер: Серверный компьютер, Средняя цена: 47.5 руб. (дисков: 1)

Компьютер: Игровой компьютер, Средняя цена: 73.52 руб. (дисков: 3)

Компьютер: Сломанный, Средняя цена: 73.52 руб. (дисков: 3)

Задание E3

Список всех жестких дисков, у которых название начинается с буквы "Б", и названия их компьютеров:

Жесткий диск: Барракуда 1TB, Цена: 45.9 руб. -> Компьютер: Домашний компьютер

Process finished with exit code 0