|  |  |  |
| --- | --- | --- |
| Защищено:  Нардид А.Н.    "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |  | Демонстрация:  Нардид А.Н.  "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

**Отчет по рубежному контролю № 1 по курсу**

**Парадигмы и конструкции языков программирования**

**ГУИМЦ**

#### Тема работы: "Рубежный контроль представляет собой разработку программы на языке Python."

3

(количество листов)

|  |  |
| --- | --- |
| ИСПОЛНИТЕЛЬ: |  |
| студент группы ИУ5Ц-53Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | (подпись) |
| Пронин В.К. | "\_\_"\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Москва, МГТУ - 2024 г.

# Тема и задание для выполнения рубежного контроля.

Тема работы: 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».

**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.

**Варианты предметной области**

№ варианта: 30

Класс 1: Факультет

Класс 2: Университет

# Листинг программы

from operator import itemgetter

class Faculty:

    """Факультет"""

    def \_\_init\_\_(self, id, name, salary, uni\_id):

        self.id = id

        self.name = name

        self.salary = salary

        self.uni\_id = uni\_id

class University:

    """Университет"""

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class FacultyUniversity:

    """Факультеты университетов для связи многие-ко-многим"""

    def \_\_init\_\_(self, faculty\_id, university\_id):

        self.faculty\_id = faculty\_id

        self.university\_id = university\_id

# Университеты

universities = [

    University(1, "МГТУ"),

    University(2, "МГУ"),

    University(3, "НИУ ВШЭ")

]

# Факультеты

faculties = [

    Faculty(1, "ИУ5", 50000, 1),

    Faculty(2, "МТ4", 45000, 1),

    Faculty(3, "ГУИМЦ", 30000, 2),

    Faculty(4, "ИУ8", 60000, 3),

    Faculty(5, "РК9", 35000, 3)

]

# Связь многие-ко-многим

fac\_uni = [

    FacultyUniversity(1, 1),

    FacultyUniversity(2, 1),

    FacultyUniversity(3, 2),

    FacultyUniversity(4, 3),

    FacultyUniversity(5, 3)

]

def main():

    """Основная функция"""

    # Соединение данных один-ко-многим

    one\_to\_many = [(f.name, f.salary, u.name)

                   for u in universities

                   for f in faculties

                   if f.uni\_id == u.id]

    # Соединение данных многие-ко-многим

    many\_to\_many\_temp = [(u.name, fu.university\_id, fu.faculty\_id)

                         for u in universities

                         for fu in fac\_uni

                         if u.id == fu.university\_id]

    many\_to\_many = [(f.name, f.salary, uni\_name)

                    for uni\_name, uni\_id, fac\_id in many\_to\_many\_temp

                    for f in faculties if f.id == fac\_id]

    # Задание B1: Все факультеты, где название начинается с "И", и их университеты

    print("\nЗадание B1\n")

    res\_1 = [(f\_name, u\_name) for f\_name, \_, u\_name in one\_to\_many if f\_name.startswith("И")]

    print(res\_1)

    # Задание B2: Университеты с минимальной зарплатой на каждом факультете, отсортировано по зарплате

    print("\nЗадание B2\n")

    res\_2\_unsorted = []

    for u in universities:

        u\_faculties = list(filter(lambda x: x[2] == u.name, one\_to\_many))

        if u\_faculties:

            min\_salary = min(sal for \_, sal, \_ in u\_faculties)

            res\_2\_unsorted.append((u.name, min\_salary))

    res\_2 = sorted(res\_2\_unsorted, key=itemgetter(1))

    print(res\_2)

    # Задание B3: Список всех факультетов и университетов, отсортированный по факультетам

    print("\nЗадание B3\n")

    res\_3 = sorted(many\_to\_many, key=itemgetter(0))

    print(res\_3)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

# Результаты работы программы

C:\Users\Вячеслав\python\П\_и\_К\_ЯП\РК\_1>python RK\_1.py

Задание B1

[('ИУ5', 'МГТУ'), ('ИУ8', 'НИУ ВШЭ')]

Задание B2

[('МГУ', 30000), ('НИУ ВШЭ', 35000), ('МГТУ', 45000)]

Задание B3

[('ГУИМЦ', 30000, 'МГУ'), ('ИУ5', 50000, 'МГТУ'), ('ИУ8', 60000, 'НИУ ВШЭ'), ('МТ4', 45000, 'МГТУ'), ('РК9', 35000, 'НИУ ВШЭ')]