# Савельев Алексей, Группа ИУ5-54Б, Вариант Г-17

# Задание

**Вариант Г.**

1. «Оркестр» и «Дирижер» связаны соотношением один-ко-многим. Выведите список всех оркестров, у которых название начинается с буквы «А», и список работающих в них дирижеров.
2. «Оркестр» и «Дирижер» связаны соотношением один-ко-многим. Выведите список оркестров с максимальной зарплатой дирижеров в каждом оркестре, отсортированный по максимальной зарплате.
3. «Оркестр» и «Дирижер» связаны соотношением многие-ко-многим. Выведите список всех связанных дирижеров и оркестров, отсортированный по оркестрам, сортировка по дирижерам произвольная.

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Класс 1** | **Класс 2** |
| 17 | Дирижер | Оркестр |

# Листинг программы

from operator import itemgetter  
  
""" Савельев Алескей, группа ИУ5-54Б, Вариант Г-17 """  
  
tasks = [  
 "Список оркестров на букву 'А' и их дирижеров", # Г1  
 "Список оркестров с max ЗП дирижоров, Сортировка по ЗП", # Г2  
 "Список всех связанных дирижеров и оркестров, Сортировка по оркестрам" # Г3  
]  
  
class Сonductor:  
 *"""Дирижер"""* def \_\_init\_\_(self, id, fio, sal, orc\_id):  
 self.id = id  
 self.fio = fio  
 self.sal = sal  
 self.dep\_id = orc\_id  
  
  
class Orcestra:  
 *"""Оркестр"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class CondOrc:  
 *"""  
 'Дирижер оркестра' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, orc\_id, cond\_id):  
 self.orc\_id = orc\_id  
 self.cond\_id = cond\_id  
  
# Оркестры  
orcestras = [  
 Orcestra(1, 'Симфонические оркестр'),  
 Orcestra(2, 'Духовой оркестр'),  
 Orcestra(3, 'Военный оркестр'),  
  
 Orcestra(11, 'Струнный оркестр'),  
 Orcestra(22, 'Амфитеатральный оркестр'),  
 Orcestra(33, 'Асинхронный оркестр')  
]  
  
# Дирижеры  
conductors = [  
 Сonductor(1, 'Гавриленко ИА', 38000, 1),  
 Сonductor(2, 'Воробьев ГА', 42000, 2),  
 Сonductor(3, 'Нинтедович ВВ', 45000, 22),  
 Сonductor(4, 'Вайбенко КК', 62000, 3),  
 Сonductor(5, 'Петрухин НА', 52000, 33)  
]  
  
# Связь  
conds\_orcs = [  
 CondOrc(1, 1),  
 CondOrc(2, 2),  
 CondOrc(3, 3),  
 CondOrc(3, 4),  
 CondOrc(3, 5),  
  
 CondOrc(11, 1),  
 CondOrc(22, 2),  
 CondOrc(33, 3),  
 CondOrc(33, 4),  
 CondOrc(33, 5)  
]  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(c.fio, c.sal, o.name)  
 for o in orcestras  
 for c in conductors  
 if c.dep\_id == o.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(o.name, c\_o.orc\_id, c\_o.cond\_id)  
 for o in orcestras  
 for c\_o in conds\_orcs  
 if o.id == c\_o.orc\_id]  
  
 many\_to\_many = [(c.fio, c.sal, orc\_name)  
 for orc\_name, orc\_id, cond\_id in many\_to\_many\_temp  
 for c in conductors if c.id == cond\_id]  
  
 print('\n\033[36mЗадание \033[33mГ1 \033[35m' + tasks[0] + "\033[0m")  
 res\_11 = list(filter(lambda x: x[2].startswith('А'), one\_to\_many))  
 for i in res\_11:  
 print(i)  
  
  
 print('\n\033[36mЗадание \033[33mГ2 \033[35m' + tasks[1] + "\033[0m")  
 res\_12\_unsorted = []  
 # Перебираем все оркестры  
 for o in orcestras:  
 # Список дирижеров орекстра  
 o\_conds = list(filter(lambda i: i[2] == o.name, one\_to\_many))  
 # Если оркестр не пустой  
 if len(o\_conds) > 0:  
 # Зарплаты дирижеров оркестра  
 o\_sals = [sal for \_, sal, \_ in o\_conds]  
 # мксимальная зарплата дирижеров оркестра  
 o\_sals\_max = max(o\_sals)  
 res\_12\_unsorted.append((o.name, o\_sals\_max))  
  
 # Сортировка по максимальной зарплате  
 # itemgetter(0) - сорт. по fio  
 # itemgetter(1) - сорт. по зп  
 # itemgetter(2) - сорт. по оркестрам  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 for i in res\_12:  
 print(i)  
  
 print('\n\033[36mЗадание \033[33mГ3 \033[35m' + tasks[2] + "\033[0m")  
 res\_13 = sorted(many\_to\_many, key=itemgetter(2))  
 for i in res\_13:  
 print(i)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# Результат работы программы

