МГТУ им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Рубежный контроль №1

«Базовые компоненты интернет-технологий»

Студентка группы ИУ5-31Б: Преподаватель кафедры ИУ5:

Слепченкова Светлана Дмитриевна Гапанюк Юрий Евгеньевич

Москва, 2022

**Вариант А. Предметная область 19.**

1. «Производитель» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
2. «Производитель» и «Деталь» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
3. «Производитель» и «Деталь» связаны соотношением многие-комногим. Выведите список всех отделов, у которых в названии присутствует слово «ООО.», и список работающих в них сотрудников.

**Листинг программы**

# -\*- coding: cp1251 -\*- from operator import itemgetter class Detail: #Деталь def \_\_init\_\_(self, id, name, cost, count\_det, det\_id): self.id = id self.name = name self.cost = cost self.det\_id = det\_id self.count = count\_det

class Manuf: #Dep #Производитель def \_\_init\_\_(self, id, name): self.id = id self.name = name

class DetManuf: #DetManuf

#'Производимые производителем детали' для реализации связи многие-ко-многим def \_\_init\_\_(self, dep\_id, det\_id):

self.dep\_id = dep\_id self.det\_id = det\_id

# Производители

Manufs = [

Manuf(1, 'OАO."РЖД"'),

Manuf(2, 'OOO."Яндекс"'),

Manuf(3, 'OOO."СветОчка"'),

]

# Детали

Details = [

Detail(1, "Гайка", 450, 400, 1),

Detail(2, "Винт", 390, 1000, 1),

Detail(3, "Винт", 600, 50, 2),

Detail(4, "Гильза", 630, 100, 3),

Detail(5, "Фланец", 880, 370, 3),

]

# Связи

Detail\_Manuf = [

DetManuf(1,1),

DetManuf(2,2),

DetManuf(3,3),

] def main():

# Соединение данных один-ко-многим one\_to\_many = [(e.name, e.cost, d.name, e.count)

for d in Manufs for e in Details if e.det\_id == d.id]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(d.name, ed.dep\_id, ed.det\_id) for d in Manufs for ed in Detail\_Manuf if d.id == ed.dep\_id]

many\_to\_many = [(e.name, e.cost, dep\_name) for dep\_name, dep\_id, det\_id in many\_to\_many\_temp for e in Details if e.id == det\_id]

print('Example A1')

res\_11 = sorted(one\_to\_many, key = itemgetter(2)) print(res\_11)

print('\nExample A2') res\_12\_unsorted = []

# Перебираем всех производителей for d in Manufs:

# Список деталей производителей

d\_details = list(filter(lambda i: i[2] == d.name, one\_to\_many)) if len(d\_details) > 0:

# Стоимоть деталей у производителя

d\_sals = [cost\*count for \_,cost,\_,count in d\_details] # Суммарная стоимость деталей у производителя d\_sals\_sum = sum(d\_sals)

res\_12\_unsorted.append((d.name, d\_sals\_sum)) # Сортировка по стоимости имеющихся деталей

res\_12 = sorted(res\_12\_unsorted, key = itemgetter(1), reverse = True) print(res\_12)

print('\nExample A3') res\_13 = {}

# Перебираем всех производителей (будем проверять наличие слова: "ООО.") for d in Manufs: if 'OOO.' in d.name: d\_detailz = list(filter(lambda i: i[2] == d.name, one\_to\_many)) if len(d\_detailz) > 0: # Только название деталей

d\_detailz\_names = [x for x,\_,\_,\_ in d\_detailz]

# Добавляем результат в словарь

# ключ - производитель, значение - список деталей res\_13[d.name] = d\_detailz\_names print(res\_13)

if \_\_name\_\_ == '\_\_main\_\_': main()

**Результаты выполнения:**

