**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

РК №1

Вариант А №11

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-31 |  | преподаватель каф. ИУ5 |
| Карпова Ксения Павловна |  | Гапанюк Юрий Евгеньевич |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2023г.

Постановка задачи

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 А.**

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



Код программы

*# Программа Компьютер*from operator import itemgetter  
  
  
class Pr: *# Program* def \_\_init\_\_(self, id, name, comp\_id, size\_in\_gb):  
 self.id = id  
 self.name = name  
 self.comp\_id = comp\_id  
 self.size\_in\_gb = size\_in\_gb  
  
  
class Comp: *# Computer* def \_\_init\_\_(self, id, model, RAM, owner):  
 self.id = id  
 self.model = model  
 self.RAM = RAM  
 self.owner = owner  
  
  
class Pr\_Comp:  
 def \_\_init\_\_(self, pr\_id, comp\_id):  
 self.pr\_id = pr\_id  
 self.comp\_id = comp\_id  
  
  
programs = [  
 Pr(1, 'Microsoft Word', 1, 2.0),  
 Pr(2, 'Microsoft Excel', 1, 1.5),  
 Pr(3, 'Google Docs', 3, 1.0),  
 Pr(4, 'LibreOffice Writer', 1, 1.8),  
 Pr(5, 'OpenOffice Calc', 4, 1.2)  
]  
  
*# Rest of the code remains the same...*computers = [  
 Comp(1, 'Model A-2000X', '8 GB DDR4', 'Иванов Иван Иванович'),  
 Comp(2, 'UltraBook Pro 15S', '16 GB DDR4', 'Петрова Анна Сергеевна'),  
 Comp(3, 'GamingBeast X9000', '32 GB DDR4', 'Смирнов Сергей Владимирович'),  
 Comp(4, 'OfficeMaster 500', '64 GB DDR4', 'Козлова Екатерина Павловна'),  
 Comp(5, 'PerformanceElite 3000', '128 GB DDR4', 'Михайлов Алексей Дмитриевич')  
]  
  
pr\_comp = [Pr\_Comp(1, 1), Pr\_Comp(1, 2), Pr\_Comp(1, 4),  
 Pr\_Comp(2, 3), Pr\_Comp(2, 1),  
 Pr\_Comp(3, 4), Pr\_Comp(4, 5),  
 Pr\_Comp(3, 5)]  
  
  
def main():  
 one\_to\_many = [(p.name, p.size\_in\_gb, c.owner, c.model)  
 for p in programs  
 for c in computers  
 if p.comp\_id == c.id]  
 one\_to\_many\_sorted = sorted(one\_to\_many, key=itemgetter(0))  
  
 many\_to\_many\_temp = [(p.name, p\_c.pr\_id, p\_c.comp\_id)  
 for p\_c in pr\_comp  
 for p in programs  
 if p.id == p\_c.pr\_id]  
  
 many\_to\_many = [(pr\_name, c.model, c.owner)  
 for pr\_name, dep\_id, p\_c\_id in many\_to\_many\_temp  
 for c in computers if c.id == p\_c\_id]  
  
 *#########################################################################3* print('\033[91mЗадание А1\033[0m')  
  
 col\_width = [40, 20, 40, 20]  
  
 *# Выводим заголовок таблицы* print(  
 f"{'Название Программы':<{col\_width[0]}}{'Гбайты':<{col\_width[1]}}{'Владелец Компьютера':<{col\_width[2]}}{'Модель Компьютера':<{col\_width[3]}}")  
 print('-' \* sum(col\_width))  
  
 *# Выводим данные в виде таблицы* for row in one\_to\_many\_sorted:  
 print(''.join(f"{str(item):<{width}}" for item, width in zip(row, col\_width)))  
  
 *######################################################* print('\n\033[91mЗадание А2\033[0m')  
  
 res\_12\_unsorted = []  
 for c in computers:  
 c\_owners = list(filter(lambda i: i[2] == c.owner, one\_to\_many))  
 if len(c\_owners) > 0:  
 c\_Gb = [gb for \_, gb, \_, \_ in c\_owners]  
 c\_Gb\_sum = sum(c\_Gb)  
 res\_12\_unsorted.append((c.owner, c.RAM, c\_Gb\_sum))  
  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 col\_width\_2 = [40, 40, 20]  
  
 print(  
 f"{'Владелец Компьютера':<{col\_width\_2[0]}}{'Оперативная память':<{col\_width\_2[0]}}{'Занятая память (в GB)':<{col\_width\_2[2]}}")  
 print('-' \* sum(col\_width\_2))  
  
 for owner, ram, total\_size in res\_12:  
 print(f"{owner:<{col\_width\_2[0]}}{ram:<{col\_width\_2[1]}}{total\_size:<{col\_width\_2[2]}}")  
  
 *#################################################################################################* print('\n\033[91mЗадание А3\033[0m')  
  
 res\_13 = {}  
  
 for p in programs:  
 if 'Microsoft' in p.name:  
 p\_c = list(filter(lambda i: i[0] == p.name, many\_to\_many\_temp))  
 res\_13[p.name] = [c.model for \_, \_, p\_c\_id in p\_c for c in computers if c.id == p\_c\_id]  
  
 print(res\_13)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описаниеРезультаты