**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Рубежный контроль №1

Вариант №2Б

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-52Б |  | преподаватель каф. ИУ5 |
| Бабин Артём |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2021 г.

**Описание задания:**

1. Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.
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).

**Вариант Б.**

1. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список всех связанных школьников и классов, отсортированный по школьникам, сортировка по классам произвольная.
2. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список классов с количеством школьников в каждом классе, отсортированный по количеству школьников.
3. «Класс» и «Учитель» связаны соотношением многие-ко-многим. Выведите список всех учителей, у которых фамилия заканчивается на «вич», и названия их классов.

**Текст программы**

class Class:  
 def \_\_init\_\_(self, id, number, letter):  
 self.id = id  
 self.number = number  
 self.letter = letter  
  
 def full\_name(self):  
 return f'{self.number}{self.letter}'  
  
  
class Student:  
 def \_\_init\_\_(self, id, first\_name, last\_name, age, id\_class):  
 self.id = id  
 self.first\_name = first\_name  
 self.last\_name = last\_name  
 self.age = age  
 self.id\_class = id\_class  
  
 def full\_name(self):  
 return f'{self.first\_name} {self.last\_name}'  
  
  
class Teacher:  
 def \_\_init\_\_(self, id, fio):  
 self.id = id  
 self.fio = fio  
  
  
class ClassTeacher:  
 def \_\_init\_\_(self, id, id\_class, id\_teacher):  
 self.id = id  
 self.id\_class = id\_class  
 self.id\_teacher = id\_teacher  
  
  
def main():  
 classes = [Class(0, 1, 'А'),  
 Class(1, 1, 'Б'),  
 Class(2, 2, 'А')]  
  
 students = [Student(0, 'Иван', 'Иванов', 8, 1),  
 Student(1, 'Пётр', 'Петров', 9, 2),  
 Student(2, 'Артём', 'Бабин', 9, 1),  
 Student(3, 'Анна', 'Перова', 10, 2),  
 Student(4, 'Михаил', 'Барышников', 8, 2),  
 Student(4, 'Дарья', 'Васильченко', 8, 0)]  
  
 teachers = [Teacher(0, 'Козлов Александр Дмитриевич'),  
 Teacher(1, 'Ахметова Фания Харисовна'),  
 Teacher(2, 'Скрипниченко Пётр Петрович')]  
  
 classes\_teachers = [ClassTeacher(0, 0, 0),  
 ClassTeacher(0, 0, 1),  
 ClassTeacher(0, 1, 0),  
 ClassTeacher(0, 1, 1),  
 ClassTeacher(0, 1, 2),  
 ClassTeacher(0, 2, 2)]  
  
 # part1  
 res1 = sorted([(stud.full\_name(), cls.full\_name()) for stud in students for cls in classes if stud.id\_class == cls.id], key=lambda x: x[0])  
 res2 = sorted({cls.full\_name(): len(list(filter(lambda x: x.id\_class == cls.id, students))) for cls in classes}.items(), key=lambda x: x[0], reverse=True)  
 res3 = {teach.fio: [cls.full\_name() for cls in classes if cls.id in [cls\_teach.id\_class for cls\_teach in classes\_teachers if cls\_teach.id\_teacher == teach.id]] for teach in teachers if str(teach.fio).endswith('вич')}  
 print(res1)  
 print(res2)  
 print(res3)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Экранная форма с результатом выполнения программы:**

