Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э.Баумана (национальный исследовательский университет)»

Факультет «Информатика и системы управления»

Курс «Базовые компоненты интернет-технологий»

Рубежный контроль №1

Выполнил:

студент группы ИУ5-33Б Цуприков Дмитрий

подпись: , дата:

Проверил:

преподаватель Юрий Гапанюк

подпись: , дата:

2022 г.

Задание

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 запросов:

1. «Отдел» и «Сотрудник» связаны отношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны отношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны отношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

Вариант 21 предметной области:
Библиотека – Язык программирования.

Исходный код

```
from operator import itemgetter
from math import inf

class Lib:
    """Библиотека"""
    def __init__(self, id, name, version, size, lang_id):
        self.id = id
        self.name = name
        self.version = version
        self.size = size
        self.lang_id = lang_id

class Lang:
    """Язык программирования"""
    def __init__(self, id, title):
        self.id = id
        self.title = title

class LibLang:
    """
    'Библиотеки языка программирования' для реализации связи многие-ко-многим
    """
    def __init__(self, lib_id, lang_id):
        self.lib_id = lib_id
        self.lang_id = lang_id

#Библиотеки
libs = [
    Lib(1, 'Algorithm', 1.24, 4348, 1),
    Lib(2, 'String', 1.15, 5258, 1),
    Lib(3, 'String', 1.15, 5258, 2),
    Lib(4, 'Mathpolib', 1.08, 9101, 2),
    Lib(5, 'Pony', 1.11, 4093, 3),
    Lib(6, 'Alamofire', 1.47, 7295, 4),
]

#Языки программирования
langs = [
    Lang(1, 'C++'),
    Lang(2, 'Python'),
```

```
    Lang(3, 'Ruby'),
    Lang(4, 'Swift'),
]

libs_langs = [
    LibLang(1, 1),
    LibLang(2, 1),
    LibLang(3, 2),
    LibLang(4, 2),
    LibLang(5, 3),
    LibLang(6, 4),
]

def main():
    """Основная функция"""
    #Соединение данных один-ко-многим
    one_to_many = [(li.name, li.version, li.size, la.title)
                   for la in langs
                   for li in libs
                   if li.lang_id == la.id]

    #Соединение данных многие-ко-многим
    many_to_many_temp = [(la.title, lila.lib_id, lila.lang_id)
                         for la in langs
                         for lila in libs_langs
                         if la.id == lila.lang_id]

    many_to_many = [(li.name, li.version, li.size, [la.title
for la in langs if la.id == li_la.lang_id][0])
                    for li in libs
                    for li_la in libs_langs
                    if li.id == li_la.lib_id]
    '''
    Выведите список языков программирования и связанных с ними
библиотек из представленных,
    названия которых начинаются с буквы 'A'
    '''

    print('Задание B1')
    res_1 = list(filter(lambda i: i[0][0] == 'A',
one_to_many))
    print(res_1)
```

```
    '''
    Выведите список названий языков программирования с
минимальным размером библиотеки из представленных
    в каждом языке программирования, отсортированный по
минимальному размеру библиотеки
    '''

    print('\nЗадание B2')
    min_sizes = {}
    for name, version, size, title in one_to_many:
        min_sizes[title] = min(min_sizes.get(title, inf),
size)
    print(sorted(min_sizes.items(), key=itemgetter(1)))

    '''
    Выведите список связанных языков программирования и
библиотек, отсортированный по номеру версии языка
    '''

    print('\nЗадание B3')
    res_3 = sorted(many_to_many, key=itemgetter(1))
    print(*res_3, sep = '\n')
    print('\nСпасибо за тестирование!')

if __name__ == '__main__':
    main()
```

Пример выполнения

```
/Users/dmitriytsuprikov/PycharmProjects/rk1_BKIT/venv/bin/python /Users/dmitriytsuprikov/PycharmProjects/rk1_BKIT/main.py
Задание B1
[('Algorithm', 1.24, 4348, 'C++'), ('Alamofire', 1.47, 7295, 'Swift')]

Задание B2
[('Ruby', 4093), ('C++', 4348), ('Python', 5258), ('Swift', 7295)]

Задание B3
('Mathpolib', 1.08, 9101, 'Python')
('Pony', 1.11, 4093, 'Ruby')
('String', 1.15, 5258, 'C++')
('String', 1.15, 5258, 'Python')
('Algorithm', 1.24, 4348, 'C++')
('Alamofire', 1.47, 7295, 'Swift')

Спасибо за тестирование!

Process finished with exit code 0
```