Условия рубежного контроля №1 по курсу ПиК ЯП

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 В.

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

3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

```
from dataclasses import dataclass

from typing import List, Dict

from itertools import groupby

from pprint import pprint



@dataclass(order=True)

class Book:

    """Книга"""

    id:       int

    title:    str

    price:    float

    store_id: int


    @property

    def first_letter(self) -> str:

        return self.title[0]



@dataclass(order=True)

class Bookstore:

    """Книжный магазин"""

    id:   int

    name: str



@dataclass
```

```
class BookBookstore:
    """Связи многие-ко-многим"""
    book_id:  int
    store_id: int



bookstores: List[Bookstore] = \
    [
        Bookstore(1, 'Книжный мир'),
        Bookstore(2, 'Буквоед'),
        Bookstore(3, 'Азбука-Аттикус'),
        Bookstore(4, 'Читай-город'),
        Bookstore(5, 'Дом книги'),
    ]

books: List[Book] = \
    [
        Book(1, 'Анна Каренина', 500.0, 1),
        Book(2, 'Преступление и наказание', 600.0, 2),
        Book(3, 'Алые паруса', 450.0, 3),
        Book(4, 'Мастер и Маргарита', 550.0, 2),
        Book(5, 'Алиса в стране чудес', 700.0, 1),
    ]

book_bookstores: List[BookBookstore] = \
    [
        BookBookstore(1, 1),
        BookBookstore(2, 2),
        BookBookstore(3, 3),
        BookBookstore(4, 2),
```

```
        BookBookstore(5, 1),
        BookBookstore(1, 3),
        BookBookstore(3, 2),
        BookBookstore(5, 2),
    ]


def main():

    store_dict: Dict[int, Bookstore] = {store.id: store for store in bookstores}

    one_to_many = \
        [
            (book, store_dict[book.store_id])
            for book in books
            if book.store_id in store_dict
        ]

    book_dict = {book.id: book for book in books}

    many_to_many = \
        [
            (book_dict[bb.book_id], store_dict[bb.store_id])
            for bb in book_bookstores
            if bb.book_id in book_dict and bb.store_id in store_dict
        ]

    print('Задание B1')
    res_1 = \
        [
```

```
            (book.title, store.name)
            for book, store in one_to_many
            if book.first_letter == 'A'
        ]
    pprint(res_1)


    print('\nЗадание B2')


    sorted_one_to_many = sorted(one_to_many, key=lambda x: x[1].name)
    grouped = groupby(sorted_one_to_many, key=lambda x: x[1].name)


    res_2 = sorted(
        [
            (store_name, min(book.price for book, _ in books_in_store))
            for store_name, books_in_store in grouped
        ],
        key=lambda x: x[1]
    )
    pprint(res_2, width=60)


    print('\nЗадание B3')


    pprint([(book.title, book.price, store.name) for book, store in sorted(many_to_many,
key=lambda x: x[0].title)])




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

Результаты выполнения:

```
Задание B1
[('Анна Каренина', 'Книжный мир'),
 ('Алые паруса', 'Азбука-Аттикус'),
 ('Алиса в стране чудес', 'Книжный мир')]

Задание B2
[('Азбука-Аттикус', 450.0),
 ('Книжный мир', 500.0),
 ('Буквоед', 550.0)]

Задание B3
[('Алиса в стране чудес', 700.0, 'Книжный мир'),
 ('Алиса в стране чудес', 700.0, 'Буквоед'),
 ('Алые паруса', 450.0, 'Азбука-Аттикус'),
 ('Алые паруса', 450.0, 'Буквоед'),
 ('Анна Каренина', 500.0, 'Книжный мир'),
 ('Анна Каренина', 500.0, 'Азбука-Аттикус'),
 ('Мастер и Маргарита', 550.0, 'Буквоед'),
 ('Преступление и наказание', 600.0, 'Буквоед')]
```