Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Базовые компоненты интернет технологий»

Отчет по рубежному контролю №2

Выполнил:		Проверил:
студент группы ИУ5-31Б		Гапанюк Ю.Е.
Слоква А. В.		
Подпись и дата: 23.12.2021		Подпись и дата:

г. Москва, 2021 г.

Текст программы:

```
from operator import itemgetter


class CD:
    """CD-диск"""

    def __init__(self, id, names, cost, dep_id):
        self.id = id
        self.names = names
        self.cost = cost
        self.dep_id = dep_id


class Lib:
    """Библиотека CD-дисков"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class CDLib:
    """
    'Сотрудники отдела' для реализации
    связи многие-ко-многим
    """

    def __init__(self, dep_id, emp_id):
        self.dep_id = dep_id
        self.emp_id = emp_id


# Библиотека CD-дисков
bibl = [
    Lib(1, 'vip- библиотека Ласточка'),
    Lib(2, 'библиотека Тутик'),
    Lib(3, 'vip- библиотека Подслушка'),
    Lib(11, 'библиотека Читай'),
    Lib(22, 'библиотека Умник'),
    Lib(33, 'vip- библиотека CD'),
]

# CD-диски
cds = [
    CD(1, 'диск Глаша', 2500, 1),
    CD(2, 'диск Лямур', 3500, 2),
    CD(3, 'диск Киберпанк', 400, 3),
    CD(4, 'диск Стас Михайлов', 3000, 3),
    CD(5, 'диск Брежнева', 250, 3),
]

cds_bibl = [
    CDLib(1, 1),
    CDLib(2, 2),
    CDLib(3, 3),
    CDLib(3, 4),
    CDLib(3, 5),
    CDLib(11, 1),
    CDLib(22, 2),
    CDLib(33, 3),
```

```
    CDLib(33, 4),
    CDLib(33, 5),
]


def sorting_by_name(table):
    return sorted(table, key=itemgetter(2))


def sorting_by_sum_size(table, bibl):
    res_12_unsorted = []
    # Перебираем все каталоги
    for d in bibl:
        # Список файлов каталога
        d_cds = list(filter(lambda i: i[2] == d.name, table))
        # Если библиотека не пустая
        if len(d_cds) > 0:
            # Стоимость дисков
            d_costs = [cost for _, cost, _ in d_cds]
            # Суммарная стоимость дисков
            d_costs_sum = sum(d_costs)
            res_12_unsorted.append((d.name, d_costs_sum))

    # Сортировка по суммарной стоимости дисков
    return sorted(res_12_unsorted, key=itemgetter(1), reverse=True)


def output_files_of_catalogs_with_PAPKA2(table, bibl):
    res_13 = {}
    # Перебираем все библиотеки
    for d in bibl:
        if 'vip-' in d.name:
            # Список дисков в библиотеке
            d_cds = list(filter(lambda i: i[2] == d.name, table))
            d_cds_names = [x for x, _, _ in d_cds]
            res_13[d.name] = d_cds_names

    return res_13


def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(e.names, e.cost, d.name)
                   for d in bibl
                   for e in cds
                   if e.dep_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
                         for d in bibl
                         for ed in cds_bibl
                         if d.id == ed.dep_id]

    many_to_many = [(e.names, e.cost, dep_name)
                    for dep_name, dep_id, emp_id in many_to_many_temp
                    for e in cds if e.id == emp_id]

    print('Задание А1')
    print(sorting_by_name(one_to_many))

    print('\nЗадание А2')
    print(sorting_by_sum_size(one_to_many, bibl))
```

```
    print('\nЗадание А3')
    print(output_files_of_catalogs_with_PAPКА2(many_to_many, bibl))


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

Результат выполнения программы:

Задание А1

[('диск Глаша', 2500, 'vip- библиотека Ласточка'), ('диск Киберпанк', 400, 'vip- библиотека Подслушка'), ('диск Стас Михайлов', 3000, 'vip- библиотека Подслушка'), ('диск Брежнева', 250, 'vip- библиотека Подслушка'), ('диск Лямур', 3500, 'библиотека Тутик')]

Задание А2

[('vip- библиотека Подслушка', 3650), ('библиотека Тутик', 3500), ('vip- библиотека Ласточка', 2500)]

Задание А3

{'vip- библиотека Ласточка': ['диск Глаша'], 'vip- библиотека Подслушка': ['диск Киберпанк', 'диск Стас Михайлов', 'диск Брежнева'], 'vip- библиотека CD': ['диск Киберпанк', 'диск Стас Михайлов', 'диск Брежнева']}

Текст программы тестирования:

```
from main import CD, Lib, CDLib, sorting_by_name, sorting_by_sum_size,
output_files_of_catalogs_with_PAPКА2
import unittest


class Tests(unittest.TestCase):
    def setUp(self):

        self.bibl = [
            Lib(1, 'vip- библиотека Ласточка'),
            Lib(2, 'библиотека Тутик'),
            Lib(3, 'vip- библиотека Подслушка'),
            Lib(11, 'библиотека Читай'),
            Lib(22, 'библиотека Умник'),
            Lib(33, 'vip- библиотека CD'),
        ]
        # Файлы
        self.cds = [
            CD(1, 'диск Глаша', 2500, 1),
            CD(2, 'диск Лямур', 3500, 2),
            CD(3, 'диск Киберпанк', 400, 3),
            CD(4, 'диск Стас Михайлов', 3000, 3),
            CD(5, 'диск Брежнева', 250, 3),
        ]
        self.cds_bibl = [
            CDLib(1, 1),
            CDLib(2, 2),
            CDLib(3, 3),
            CDLib(3, 4),
            CDLib(3, 5),
```

```
            CDLib(11, 1),
            CDLib(22, 2),
            CDLib(33, 3),
            CDLib(33, 4),
            CDLib(33, 5),
        ]
        # Соединение данных один-ко-многим
        self.one_to_many = [(e.names, e.cost, d.name)
                            for d in bibl
                            for e in cds
                            if e.dep_id == d.id]

        # Соединение данных многие-ко-многим
        self.many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
                                  for d in bibl
                                  for ed in cds_bibl
                                  if d.id == ed.dep_id]

        self.many_to_many = [(e.names, e.cost, dep_name)
                             for dep_name, dep_id, emp_id in
many_to_many_temp
                             for e in cds if e.id == emp_id]

    def test_sorting_by_name(self):
        result = sorting_by_name(self.one_to_many)
        desired_result = [('диск Глаша', 2500, 'vip- библиотека Ласточка'),
                          ('диск Киберпанк', 400, 'vip- библиотека
Подслушка'),
                          ('диск Стас Михайлов', 3000, 'vip- библиотека
Подслушка'),
                          ('диск Брежнева', 250, 'vip- библиотека
Подслушка'),
                          ('диск Лямур', 3500, 'библиотека Тутик')]

        self.assertEqual(result, desired_result)

    def test_sorting_by_sum(self):
        result = sorting_by_sum_size(self.one_to_many, self.catalogs)
        desired_result = [('vip- библиотека Подслушка', 3650), ('библиотека
Тутик', 3500),
                          ('vip- библиотека Ласточка', 2500)]
        self.assertEqual(result, desired_result)

    def test_output_PAPKA2(self):
        result = output_files_of_catalogs_with_PAPKA2(self.many_to_many,
self.catalogs)
        desired_result = {'vip- библиотека Ласточка': ['диск Глаша'],
                          'vip- библиотека Подслушка': ['диск Киберпанк',
'диск Стас Михайлов', 'диск Брежнева'],
                          'vip- библиотека CD': ['диск Киберпанк', 'диск Стас
Михайлов', 'диск Брежнева']}
        self.assertEqual(result, desired_result)
```

Пример работы тестирования: