МГТУ им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Рубежный контроль №2
«Базовые компоненты интернет-технологий»

Студентка группы ИУ5-31Б:
Гапеева Олеся Радиковна

Преподаватель кафедры ИУ5:
Гапанюк Юрий Евгеньевич

Москва, 2022

Вариант А. Предметная область 3.

1. «Водитель» и «Автопарк» связаны соотношением один-ко-многим. Выведите список всех связанных водителей и автопарков, отсортированный по автопаркам, сортировка по водителям произвольная.
2. «Водитель» и «Автопарк» связаны соотношением один-ко-многим. Выведите список автопарков с суммарным рейтингом в каждом автопарке, отсортированный по суммарному рейтингу.
3. «Водитель» и «Автопарк» связаны соотношением многие-ко-многим. Выведите список всех автопарком, у которых в названии присутствует слово «taxi», и список находящихся в них файлов.

Задание:

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Измененный код РК1:

```
from operator import itemgetter


class Driver:
    def __init__(self, Id, Name, Rating, IdDr):
        self.id = Id
        self.name = Name
        self.rating = Rating  # количественный признак
        self.ida = IdDr


class Autopark:
    def __init__(self, Id, Name):
        self.id = Id
        self.name = Name

class DrivOfAu:
    def __init__(self, IdA, IdD):
        self.idd = IdD
        self.ida = IdA

def one_many(drivers, autos):
    return [(d.name, d.rating, a.name)
            for a in autos
            for d in drivers
            if d.ida == a.id]


    # Соединение данных многие-ко-многим
def many_many(drivers, autos, driverofauto):
    many_to_many_temp = [(a.name, da.ida, da.idd)
                         for a in autos
                         for da in driverofauto
                         if a.id == da.ida]

    return [(d.name, d.rating, au_name)
                   for au_name, auto_id, driver_id in many_to_many_temp
                   for d in drivers
                   if d.id == driver_id]

def Task1(drivers, autos):
    res_11 = sorted(one_many(drivers, autos), key=itemgetter(2))
    return res_11


def Task2(drivers, autos) -> list:
    res_12_unsorted =[]
    # Перебираем все автопакри
```

```
    for d in autos:
        # Список водителей автопарка
        d_autos = list(filter( lambda i: i[2] == d.name, one_many(drivers,
autos)))
        # Если автопарк не пустой
        if len(d_autos) > 0:
            # Рейтинги водителей автопарка
            d_sizes = [sal for _, sal, _ in d_autos]
            # Суммарный рейтинг водителей автопарка
            d_sizes_sum = sum(d_sizes)
            res_12_unsorted.append((d.name, d_sizes_sum))

    # Сортировка по суммарной зарплате
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def Task3(drivers, autos, driverofauto) -> list:
    res_13 = {}
    # Перебираем все автопарки
    for d in autos:
        if 'taxi' in d.name:
            # Список водителей автопарка
            d_autos = list(filter(lambda i: i[2] == d.name, many_many(drivers,
autos,driverofauto)))
            # Только фамилия водителей
            d_autos_names = [x for x, _, _ in d_autos]
            # Добавляем результат в словарь
            # ключ - автопарк, значение - список фамилий
            res_13[d.name] = d_autos_names
    return res_13
```

Код тестов:

```
import unittest
from rk1_modified import *

class Different_linksTestCase(unittest.TestCase):
    def setUp(self):
        self.drivers = [
            Driver(1, 'Viktorov', 4.3, 1),
            Driver(2, 'Ephimov', 4.6, 2),
            Driver(3, 'Petrov', 5.0, 3),
            Driver(4, 'Laviev', 4.87, 3),
            Driver(5, 'Cotov', 4.77, 3),
        ]
        self.auto = [
            Autopark(1, 'I_taxi'),
            Autopark(2, 'Citimobile'),
            Autopark(3, 'Yandex taxi'),
            Autopark(11, 'Maxim'),
            Autopark(22, 'Vine taxi'),
            Autopark(33, 'Drive'),
        ]
        self.driverofauto = [DrivOfAu(1, 1),
                DrivOfAu(2, 2),
                DrivOfAu(3, 3),
                DrivOfAu(3, 4),
                DrivOfAu(3, 5),
                DrivOfAu(11, 1),
                DrivOfAu(22, 2),
                DrivOfAu(33, 3),
                DrivOfAu(33, 4),
                DrivOfAu(33, 5),
        ]

    def test_Task_1(self):
        expected = [('Ephimov', 4.6, 'Citimobile'),
         ('Viktorov', 4.3, 'I_taxi'),
         ('Petrov', 5.0, 'Yandex taxi'),
         ('Laviev', 4.87, 'Yandex taxi'),
         ('Cotov', 4.77, 'Yandex taxi')
         ]
        result = Task1(self.drivers, self.auto)
        self.assertEqual(expected, result)

    def test_Task_2(self):
            expected = [('Yandex taxi', 14.64), ('Citimobile', 4.6), ('I_taxi',
4.3)]
            result = Task2(self.drivers, self.auto)
            self.assertEqual(expected, result)
```

```
    def test_Task_3(self):
            expected = {'I_taxi': ['Viktorov'],
            'Yandex taxi': ['Petrov', 'Laviev', 'Cotov'],
            'Vine taxi': ['Ephimov']}
            result = Task3(self.drivers, self.auto, self.driverofauto)
            self.assertEqual(expected, result)
```

Результат тестирования:

```
PS C:\Users\olgap\Documents\BKIT\RK1> python.exe -m unittest test.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```