Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «БКИТ»

Отчет по РК2

Выполнил:
студент группы ИУ5-35Б
Большаков Георгий
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2022 г.

Код измененной для модульного тестирования программы из РК1:

Файл rk2.py

```
from operator import itemgetter


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


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


class DrivOfAu: # связь автопарк - водитель
    def __init__(self, IdA, IdD):
        self.ida = IdA  #id автопарка
        self.idd = IdD  #id водителя


drivers = [
    Driver(1, 'Dubov', 4.3, 1),
    Driver(2, 'Zotov', 4.6, 2),
    Driver(3, 'Medvedev', 5.0, 3),
    Driver(4, 'Yudin', 4.87, 3),
    Driver(5, 'Volkov', 4.77, 3),
    Driver(6, 'Petrov', 4.23, 1),
    Driver(7, 'Markov', 4.54, 2),
]

autos = [
    Autopark(1, 'Uber'),
    Autopark(2, 'Yandex.Taxi'),
    Autopark(3, 'Yo-Taxi'),
    Autopark(4, 'Citymobil'),
    Autopark(5, 'Gett Taxi'),
    Autopark(6, 'Drive'),
]

# Связь автопарк - водитель
driverofauto = [
    DrivOfAu(1, 1),
    DrivOfAu(1, 6),
    DrivOfAu(2, 2),
    DrivOfAu(2, 7),
    DrivOfAu(3, 3),
    DrivOfAu(3, 4),
    DrivOfAu(3, 5),
    DrivOfAu(4, 1),
    DrivOfAu(4, 7),
    DrivOfAu(5, 2),
    DrivOfAu(5, 6),
    DrivOfAu(6, 3),
```

```
    DrivOfAu(6, 4),
    DrivOfAu(6, 5),
]

one_to_many = [(d.name, d.rating, a.name)
               for a in autos for d in drivers if d.ida == a.id]

many_to_many_temp = [(a.name, da.ida, da.idd)
                     for a in autos
                     for da in driverofauto
                     if a.id == da.ida]

many_to_many = [(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():
    res_1 = {}
    for d in drivers:
        if 'ov' == d.name[-2:]:
            d_ov = list(filter(lambda i: i[0] == d.name, many_to_many))

            d_autos_names = [x for _, _, x in d_ov] # [id, driver_name, au-
topark_name]

            res_1[d.name] = d_autos_names
    return res_1


def task2():
    res_2_unsorted = []
    for d in autos:
        d_autos = list(filter(lambda i: i[2] == d.name, many_to_many))

        if len(d_autos) > 0:
            d_rating = [rating for _, rating, _ in d_autos]

            d_rating_sum = sum(d_rating)
            d_rating_avg = '{:.2f}'.format(d_rating_sum / len(d_rating))

            res_2_unsorted.append((d.name, d_rating_avg))

        res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)
    return(res_2)

def task3():
    res_3 = {}
    for d in autos:
        if 'Y' == d.name[0]:
            d_autos = list(filter(lambda i: i[2] == d.name, many_to_many))

            d_autos_names = [x for x, _, _ in d_autos]

            res_3[d.name] = d_autos_names
    return(res_3)
```

Код программы для модульного тестирования:

Файл tdd.py

```
import unittest
from rk2 import task1, task2, task3

task_1_result = {'Dubov': ['Uber', 'Citymobil'],
                 'Markov': ['Yandex.Taxi', 'Citymobil'],
                 'Petrov': ['Uber', 'Gett Taxi'],
                 'Volkov': ['Yo-Taxi', 'Drive'],
                 'Zotov': ['Yandex.Taxi', 'Gett Taxi']
                 }

task_2_result = [('Yo-Taxi', '4.88'),
                 ('Drive', '4.88'),
                 ('Yandex.Taxi', '4.57'),
                 ('Citymobil', '4.42'),
                 ('Gett Taxi', '4.42'),
                 ('Uber', '4.27')
                ]

task_3_result = {'Yandex.Taxi': ['Zotov', 'Markov'], 'Yo-Taxi': ['Medvedev',
'Yudin', 'Volkov']}


class TasksTestCase(unittest.TestCase):
    def test_task_1(self):
        self.assertEqual(task_1_result, task1())

    def test_task_2(self):
        self.assertEqual(task_2_result, task2())

    def test_task_3(self):
        self.assertEqual(task_3_result, task3())
```

Результат:

```
(venv) PS C:\Users\Lenovo\Desktop\Уче6аМГТУ\СЕМ_3\БКИТ\РК-1> pytest -v tdd.py
============================= test session starts =============================
platform win32 -- Python 3.10.4, pytest-7.2.0, pluggy-1.0.0 -- C:\Users\Lenovo\Desktop\Уче6аМГТУ\СЕМ_3\БКиТ\РК-1\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Lenovo\Desktop\Уче6аМГТУ\СЕМ_3\БКИТ\РК-1
collected 3 items

tdd.py::TasksTestCase::test_task_1 PASSED                                [ 33%]
tdd.py::TasksTestCase::test_task_2 PASSED                                [ 66%]
tdd.py::TasksTestCase::test_task_3 PASSED                                [100%]

============================== 3 passed in 0.02s ==============================
```