Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологиий»

Рубежный контроль №2.

Выполнил:
студент группы ИУ5-34Б:
Гордеев Матвей Владиславович
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич
Подпись и дата:

2022

Условия рубежного контроля №2 по курсу БКИТ

Рубежный контроль представляет собой разработку тестов на языке Python.

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).

Код программы

main.py

```python
''' автопарк '''
class Autopark:
    def __init__(self, id, name):
        self.id = id
        self.name = name

''' водитель '''
class Driver:
    def __init__(self, id, name, Rate, Autopark_id):
        self.id = id
        self.name = name
        self.Autopark_id = Autopark_id
        self.Rate=Rate

''' связь водителей с автопарками '''
class DriverInAutopark:
    def __init__(self, Driver_id, Autopark_id):
        self.Driver_id = Driver_id
        self.Autopark_id = Autopark_id

Autoparks = [
    Autopark(1, "Citymobile"),
    Autopark(2, "BestTaxi"),
    Autopark(3, "Blablacar"),
    Autopark(11, "AutoFree"),
    Autopark(22, "Uber"),
    Autopark(33, "AutoForYou"),
]

Drivers = [
    Driver(1, "Mirshin", 3.0, 1),
    Driver(2, "Samsonov", 5.0, 2),
    Driver(3, "Victorovich", 2.1, 3),
    Driver(4, "Alexandrovich", 4.1, 3),
    Driver(5, "Grigoryevich", 1.4, 3)
]
```

```python

DriversInAutoparks = [
    DriverInAutopark(1, 1),
    DriverInAutopark(2, 2),
    DriverInAutopark(3, 3),
    DriverInAutopark(3, 4),
    DriverInAutopark(3, 5),
    DriverInAutopark(11, 1),
    DriverInAutopark(22, 2),
    DriverInAutopark(33, 3),
    DriverInAutopark(33, 4),
    DriverInAutopark(33, 5)
]


''' связь один-ко-многим'''

one_to_many = [(dr.name, dr.Rate, au.name)
               for au in Autoparks
               for dr in Drivers
               if dr.Autopark_id == au.id]

many_to_many_temp = [(au.name, DrInAu.Autopark_id, DrInAu.Driver_id)
               for au in Autoparks
               for DrInAu in DriversInAutoparks
               if au.id == DrInAu.Autopark_id]

''' связь много-ко-многим'''
many_to_many = [(dr.name, dr.Rate, AutoparkName)
               for AutoparkName, AutoparkId, DriverId in many_to_many_temp
               for dr in Drivers
               if dr.id == DriverId]


```

```python

'''============================Задания============================'''
def number_1(one_to_many):
    ans_1 = {}
    for DriverName, Rate, AutoparkName in one_to_many:
        if AutoparkName[0] == 'B': #Название автопарка начинается на B
            if AutoparkName in ans_1:
                ans_1[AutoparkName].append(DriverName)
            else:
                ans_1[AutoparkName] = [DriverName]
    return dict(ans_1.items())


def number_2(one_to_many):
    ans_2 = {}
    for TMP, Rate, AutoparkName in one_to_many:
        if AutoparkName in ans_2:
            ans_2[AutoparkName] = max(ans_2[AutoparkName], Rate)
        else:
            ans_2[AutoparkName] = Rate
    return dict({key: value for key, value in sorted(ans_2.items())}.items())

def number_3(many_to_many):
    ans_3 = []
    for DriverName, TMP, AutoparkName in many_to_many:
        ans_3.append((AutoparkName, DriverName))
    return list(sorted(ans_3, key=lambda x: len(x)))


```

TDD_tests.py

```python
import unittest
import sys, os

sys.path.append(os.getcwd())
from main import *
class TestNumbers(unittest.TestCase):
    def test_number_1(self):
        self.assertEqual(number_1(one_to_many),
                         {'BestTaxi': ['Samsonov'], 'Blablacar': ['Victorovich', 'Alexandrovich', 'Grigoryevich']})
    def test_number_2(self):
        self.assertEqual(number_2(one_to_many), {'BestTaxi': 5.0, 'Blablacar': 4.1, 'Citymobile': 3.0})
    def test_number_3(self):
        self.assertEqual(number_3(many_to_many), [('Citymobile', 'Mirshin'), ('BestTaxi', 'Samsonov'), ('Blablacar', 'Victorovich')])
```

Результат выполнения

```
Testing started at 0:12 ...


Ran 3 tests in 0.003s
Launching unittests with arguments python -m unit


OK

Process finished with exit code 0
```