Малышко Артём ИУ5-34Б

Отчёт РК2 по дисциплине

Парадигмы и конструкторы языков программирования

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).

Текст программы main.py

```
from operator import itemgetter

class House:
    def __init__(self, id, number, cost, street_id):
        self.id = id
        self.number = number
        self.cost = cost
        self.street_id = street_id

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

class HouStr:
    def __init__(self, street_id, house_id):
        self.street_id = street_id
        self.house_id = house_id

def get_one_to_many(houses, streets):
    return [(h.number, h.cost, s.name)
            for h in houses
            for s in streets
            if h.street_id == s.id]

def get_many_to_many(houses, streets, hou_str):
    return [(h.number, h.cost, s.name)
            for hs in hou_str
            for h in houses if h.id == hs.house_id
            for s in streets if s.id == hs.street_id]

def task_g1(houses, streets):
    res = {}
    for s in streets:
        if s.name[0] == 'A':
            h_s = [(house.number, house.cost) for house in houses if
house.street_id == s.id]
            res[s.name] = h_s
    return res

def task_g2(one_to_many):
    res = []
    for s_name in set([item[2] for item in one_to_many]):
```

```
        s_houses = list(filter(lambda i: i[2] == s_name, one_to_many))
        if s_houses:
            s_costs = [cost for _, cost, _ in s_houses]
            s_max = max(s_costs)
            res.append((s_name, s_max))
    return sorted(res, key=itemgetter(1), reverse=True)

def task_g3(many_to_many):
    return sorted(many_to_many, key=itemgetter(2))

def get_test_data():
    streets = [
        Street(1, 'Английский бульвар'),
        Street(2, 'Амурская улица'),
        Street(3, 'Абрикосовая улица'),
        Street(4, 'Железнодорожная улица'),
        Street(5, 'Приморская улица'),
        Street(6, 'Острякова'),
    ]
    houses = [
        House(1, 111, 9000, 1),
        House(2, 123, 4500, 2),
        House(3, 125, 3300, 2),
        House(4, 635, 1200, 3),
        House(5, 325, 6300, 4),
        House(6, 265, 4400, 4),
    ]
    hou_str = [
        HouStr(1, 1),
        HouStr(2, 2),
        HouStr(2, 3),
        HouStr(3, 4),
        HouStr(4, 5),
        HouStr(4, 6),
        HouStr(5, 1),
        HouStr(6, 2),
    ]
    return streets, houses, hou_str

def main():
    streets, houses, hou_str = get_test_data()
    one_to_many = get_one_to_many(houses, streets)
    many_to_many = get_many_to_many(houses, streets, hou_str)

    print("Задание Г1")
    print(task_g1(houses, streets))
    print("Задание Г2")
    print(task_g2(one_to_many))
    print("Задание Г3")
    print(task_g3(many_to_many))

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

test.py

```
import unittest

from main import get_test_data, get_one_to_many, get_many_to_many, task_g1,
task_g2, task_g3

class TestTasks(unittest.TestCase):
```

```
    def setUp(self):
        self.streets, self.houses, self.hou_str = get_test_data()
        self.one_to_many = get_one_to_many(self.houses, self.streets)
        self.many_to_many = get_many_to_many(self.houses, self.streets,
self.hou_str)
    def test_task_g1(self):
        result=task_g1(self.houses,self.streets)
        expected = {
            'Английский бульвар': [(111, 9000)],
            'Амурская улица': [(123, 4500), (125, 3300)],
            'Абрикосовая улица': [(635, 1200)],
        }
        self.assertEqual(result, expected)

    def test_task_g2(self):
        result = task_g2(self.one_to_many)
        expected = [('Английский бульвар', 9000),
                    ('Железнодорожная улица', 6300),
                    ('Амурская улица', 4500),
                    ('Абрикосовая улица', 1200)]
        self.assertEqual(result, expected)

    def test_task_g3(self):
        result = task_g3(self.many_to_many)
        expected= [(635, 1200, 'Абрикосовая улица'),
                   (123, 4500, 'Амурская улица'),
                   (125, 3300, 'Амурская улица'),
                   (111, 9000, 'Английский бульвар'),
                   (325, 6300, 'Железнодорожная улица'),
                   (265, 4400, 'Железнодорожная улица'),
                   (123, 4500, 'Острякова'),
                   (111, 9000, 'Приморская улица')]
        self.assertEqual(result, expected)
if __name__ == '__main__':
    unittest.main()
```

Выводит

```
Ran 3 tests in 0.002s

OK

Process finished with exit code 0
```