ИУ5-31Б Корецкий К.В.

Отчет по РК2

Текст измененной программы РК1 для модульного тестирования:

```
from operator import itemgetter

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

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

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

streets = [
    Street(1, 'Бауманская'),
    Street(2, 'Ладожская'),
    Street(3, 'Бригадирный переулок'),
    Street(4, 'Лефортовский переулок'),
    Street(5, 'Рубцовская набережная'),
]

houses = [
    House(1, 1, 2500000000, 'ГЗ', 1),
    House(2, 5, 4355000000, 'УЛК', 2),
    House(3, 10, 450000000, 'ГБОУ Карбышева', 3),
    House(4, 12, 150000000, 'Дом', 4),
    House(5, 4, 492000000, 'Дом', 5),
    House(6, 1, 120000000, 'Магазин', 5),
    House(7, 3, 240000000, 'Поликлиника МГТУ', 1)
]

houses_streets = [
    HouseStreet(1,1),
    HouseStreet(2,2),
    HouseStreet(3,3),
    HouseStreet(4,4),
    HouseStreet(5,5),
    HouseStreet(5,6),
```

```python
    HouseStreet(1,7)
]

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



def many_to_many(streets, houses):
    many_to_many_temp = [(s.name, hs.street_id, hs.house_id)
                         for s in streets
                         for hs in houses_streets
                         if s.id == hs.street_id]
    return [(h.id, street_id)
            for name, street_id, house_id in many_to_many_temp
            for h in houses if h.id==house_id]

def A1(streets, houses) -> list:
    res_31 = sorted(one_to_many(streets, houses), key=itemgetter(1, 0)) #sorted by
street name
    return(list(res_31))

def A2(streets, houses) -> list:
    res32 = []
    for i in streets:
        s_houses = [ _ for _ in filter(lambda a: a[1]==i.name ,one_to_many(streets,
houses) )]
        res32.append((i.name, sum([ _[3] for _ in s_houses])))
    return sorted(res32, key=itemgetter(1, 0))

def A3(streets, houses, str_to_find) -> list:
    res33 = []
    for i in filter(lambda a: str_to_find in streets[a[1]-1].name,
many_to_many(streets, houses)):
        res33.append((streets[i[1]-1].name, sorted([ _.name for _ in filter(lambda a:
a.street_id==i[1], houses)])))
    return sorted(res33, key=itemgetter(1, 0))

if __name__ == '__main__':
    print('Задание A1')
    print(A1(streets, houses))
    print('Задание A2')
    print(A2(streets, houses))
    print('Задание A3')
    print(A3(streets, houses, 'переулок'))
```

Текст программы тестирования:

```python
import unittest
from RK1_for_RK2 import Street, House, HouseStreet, A1, A2, A3
```

```python
class RK1_test(unittest.TestCase):

    def setUp(self):
        self.streets = [
            # id, name
            Street(1, 'Бауманская'),
            Street(2, 'Рубцовская набережная')
        ]
        self.houses = [
            # id, number, price, name, street_id
            House(1, 2, 25, 'ГЗ', 1),
            House(2, 5, 43, 'УЛК', 2),
            House(3, 12, 16, 'Дом', 2)
        ]
            # street_id, house_id
        self.houses_streets = [
            HouseStreet(1, 1),
            HouseStreet(2, 2),
            HouseStreet(2, 3)
        ]


    def test_A1(self):
        # expected list of tuples sorted ASC by street name
        expected_result = [
            ('ГЗ', 'Бауманская', 2, 25),
            ('Дом', 'Рубцовская набережная', 12, 16),
            ('УЛК', 'Рубцовская набережная', 5, 43)
        ]

        result = A1(self.streets, self.houses)
        self.assertEqual(result, expected_result)

    def test_A2(self):
        # expected tuple of streets sorted ASC by price
        expected_result = [
            ('Бауманская', 25),
            ('Рубцовская набережная', 59)
        ]
        result = A2(self.streets, self.houses)
        self.assertEqual(result, expected_result)

    def test_A3(self):
        # expected lsit of tuples with street names and houses on it, sorted by name
ASC
        expected_result = [
            ('Рубцовская набережная', ['Дом','УЛК'])
        ]
        result = A3(self.streets, self.houses, 'набережная')
        self.assertEqual(result, expected_result)
```

```python
if __name__ == '__main__':
    unittest.main()
```

Результат:

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```