Московский государственный технический университет им. Н.Э. Баумана

Факультет «Парадигмы и конструкции языков программирования»
Кафедра ИУ5 «Системы обработки информации и управления»

РК № 2
Вариант 6(Г)

Выполнил:
студент группы ИУ5-34Б:
Каятский Павел Евгеньевич
Подпись и дата:

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

Москва, 2023 г.

Условия рубежного контроля №2 по курсу ПиКЯП

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(с рефакторингом)

```
#Вариант Г, вариант 6( Класс 1 - Дом, Класс 2 - улица)

"""""«Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите
список всех отделов, у которых название начинается с буквы «А», и список
работающих в них сотрудников.
«Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список
улиц с максимальной высотой дома на каждой улице, отсортированный по
максимальной высоте.
«Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список
всех связанных домов и улиц, отсортированный по улицам, сортировка по
сотрудникам произвольная.
"""
from operator import itemgetter

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

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

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


Houses = [
    House(1, 3, 7, 1),
    House(2, 13, 9, 2),
    House(3, 15, 5, 2),
    House(4, 45, 15, 3),
    House(5, 5, 7, 1),
    House(6,88,22,5),
    House(7,54,19,4),
    House(8,15,15,5),

]

Streets = [
    Street(1, "Zeleniy prospect"),
    Street(2, "Abramovskaya street"),
```

```
    Street(3, "Adelmanovskaya shosse"),
    Street(4, "Voskresenskaya street"),
    Street(5, "Babaevskiy proezd"),
    Street(6, "Timura Frunze street"),
    Street(7,"Peshehodniy bulvar"),
]

Houses_Streets = [
    House_Street(1,1),
    House_Street(1,2),
    House_Street(1,3),
    House_Street(2,4),
    House_Street(2,5),
    House_Street(3,7),
    House_Street(3,6),
    House_Street(3,5),
    House_Street(4,4),
    House_Street(4,1),
    House_Street(5,2),


]

def t1_sol(one_to_many):
    t1_sol_list = []
    for i in range(0, len(one_to_many)):
        sname_temp = one_to_many[i][2]
        if sname_temp[0] == "A":
            t1_sol_list.append(one_to_many[i])
    return t1_sol_list

def t2_sol(one_to_many):
    t2_sol_list = []
    one_to_many_temp = []
    sname_temp = []
    for i in range(0, len(one_to_many)):
        temp1 = one_to_many[i]
        temp1_sname = one_to_many[i][2]
        temp1_max_height = one_to_many[i][1]
        for j in range(i + 1, len(one_to_many)):
            if one_to_many[j][2] == temp1_sname and one_to_many[j][1] >
temp1_max_height:
                temp1 = one_to_many[j]
        if temp1 not in one_to_many_temp:
            if temp1[2] not in sname_temp:
                one_to_many_temp.append(temp1)
                sname_temp.append(temp1[2])
    t2_sol_list = (sorted(one_to_many_temp, key=itemgetter(1)))
    return t2_sol_list

def t3_sol(many_to_many):
    t3_sol_list = []
    t3_sol_list = (sorted(many_to_many, key=itemgetter(2)))
    return t3_sol_list

def main():
    one_to_many = [(h.number,h.height, s.name)
                   for h in Houses
                   for s in Streets
                   if h.street_id == s.id]
    many_to_many_temp = [(s.name, hs.house_id, hs.street_id)
                         for s in Streets
                         for hs in Houses_Streets
                         if s.id == hs.street_id]
```

```
    many_to_many = [(h.number, h.height, street_name)
                    for street_name, street_id, house_id in many_to_many_temp
                    for h in Houses if h.id == house_id]



    print("Task Г1")
    print(t1_sol(one_to_many))

    print("Task Г2")
    print(t2_sol(one_to_many))

    print("Task Г3")
    print(t3_sol(many_to_many))


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

Tddtest.py

```
import unittest
from RK1 import *

class TestsRK2(unittest.TestCase):
    Houses = [
        House(1, 3, 7, 1),
        House(2, 13, 9, 2),
        House(3, 15, 5, 2),
        House(4, 45, 15, 3),
        House(5, 5, 7, 1),
        House(6, 88, 22, 5),
        House(7, 54, 19, 4),
        House(8, 15, 15, 5),

    ]

    Streets = [
        Street(1, "Zeleniy prospect"),
        Street(2, "Abramovskaya street"),
        Street(3, "Adelmanovskaya shosse"),
        Street(4, "Voskresenskaya street"),
        Street(5, "Babaevskiy proezd"),
        Street(6, "Timura Frunze street"),
        Street(7, "Peshehodniy bulvar"),
    ]

    def test1(self):
        one to many = [(h.number, h.height, s.name)
                       for h in Houses
                       for s in Streets
                       if h.street_id == s.id]
        self.assertEqual(t1_sol(one_to_many),
                         [(13, 9, 'Abramovskaya street'), (15, 5,
'Abramovskaya street'), (45, 15, 'Adelmanovskaya shosse')])

    def test2(self):
        one_to_many = [(h.number, h.height, s.name)
                       for h in Houses
                       for s in Streets
                       if h.street_id == s.id]
        self.assertEqual(t2_sol(one_to_many),
```

```
                          [(3, 7, 'Zeleniy prospect'),
                           (13, 9, 'Abramovskaya street'),
                           (45, 15, 'Adelmanovskaya shosse'),
                           (54, 19, 'Voskresenskaya street'),
                           (88, 22, 'Babaevskiy proezd')])


    def test3(self):
        many_to_many_temp = [(s.name, hs.house_id, hs.street_id)
                             for s in Streets
                             for hs in Houses_Streets
                             if s.id == hs.street_id]
        many_to_many = [(h.number, h.height, street_name)
                        for street_name, street_id, house_id in
many_to_many_temp
                        for h in Houses if h.id == house_id]
        self.assertEqual(t3_sol(many_to_many),
                         [(13, 9, 'Abramovskaya street'), (13, 9,
'Abramovskaya street'), (15, 5, 'Adelmanovskaya shosse'), (5, 7, 'Babaevskiy
proezd'), (5, 7, 'Babaevskiy proezd'), (54, 19, 'Peshehodniy bulvar'), (88,
22, 'Timura Frunze street'), (45, 15, 'Voskresenskaya street'), (45, 15,
'Voskresenskaya street'), (3, 7, 'Zeleniy prospect'), (3, 7, 'Zeleniy
prospect')])

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

Результат выполнения Тестов

```
C:\Users\kayat\AppData\Local\Microsoft\WindowsApps\python3.11.exe "A:/PyCharm 2023.2.3/plugins/python/helpers/pycharm/_jb_unittest_runner.py
Testing started at 15:00 ...
Launching unittests with arguments python -m unittest A:\Учёба\МГТУ\МГТУ 3 сем\Парадигмы и конструкции ЯП\Лабораторные работы\РК1\RK1\tddtes


Ran 3 tests in 0.009s

OK

Process finished with exit code 0
```