Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ: Радиотехнический (РТ)

КАФЕДРА: Системы обработки информации и управления (ИУ5)

РК 2.

Выполнила Надыршина А.А.

Проверил Гапанюк Ю.Е.

2022 г.

Задание.

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).

Задание РК1.

Предметная область E, вариант 21. Классы: Язык программирования, Оператор.

Задания:

1. «Язык программирования» и «Оператор» связаны соотношением один-ко-многим. Выведите список всех языков, которые начинаются с буквы «P», и список его операторов.
2. «Язык программирования» и «Оператор» связаны соотношением один-ко-многим. Выведите список языков со средним количеством букв в названии операторов, отсортированный по среднему количеству букв. Среднее количество букв должна быть округлено до 2 знаков после запятой.
3. «Язык программирования» и «Оператор» связаны соотношением многие-ко-многим. Выведите список всех операторов, у которых название начинается с буквы «w», и названия их языков.

Текст программы.

Рефакторинг.

rk1.py

```python
from operator import itemgetter

class Operator:
    """Оператор"""
    def __init__(self, id, name, typ, num, lan_id):
        self.id = id
        self.name = name
        self.typ = typ
        self.num = num
        self.lan_id = lan_id

class Language:
    """Язык программирования"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class OpLan:
    """
    'Оператор языка программирования' для реализации
    связи многие-ко-многим
    """
    def __init__(self, lan_id, op_id):
        self.lan_id = lan_id
        self.op_id = op_id

# Языки
Languages = [
    Language(1, 'Python'),
    Language(2, 'C++'),
    Language(3, 'Pascal'),
    Language(4, 'Java'),
```

```
    Language(5, 'C'),
    Language(6, 'Delphi'),
]

# Операторы
Operatops = [
    Operatop(1, 'switch/case', 'Условие', 10, 1),
    Operatop(2, '(type)', 'Приведение типа', 4, 2),
    Operatop(3, 'Goto', 'Безусловный переход', 4, 3),
    Operatop(4, 'while', 'Цикл', 5, 1),
    Operatop(5, 'if', 'Условие', 2, 1),
]

Operatops_Languages = [
    OpLan(1,1),
    OpLan(4,1),

    OpLan(1,4),
    OpLan(2,4),
    OpLan(3,4),
    OpLan(4,4),
    OpLan(5,4),
    OpLan(6,4),

    OpLan(1,5),
    OpLan(2,5),
    OpLan(3,5),
    OpLan(4,5),
    OpLan(5,5),
    OpLan(6,5),
    OpLan(2,2),
    OpLan(5,2),

    OpLan(3,3),
    OpLan(6,3),
]


    # Соединение данных один-ко-многим
def one_to_many(Languages,Operatops):
    return [(e.name, e.num, e.typ, d.name)
        for d in Languages
        for e in Operatops
        if e.lan_id==d.id]

    # Соединение данных многие-ко-многим
def many_to_many_temp(Languages,Operatops_Languages):
    return [(d.name, ed.lan_id, ed.op_id)
        for d in Languages
        for ed in Operatops_Languages
        if d.id==ed.lan_id]

def many_to_many(many_to_many_temp,Operatops):
    return [(e.name, e.num, lan_name)
        for lan_name, lan_id, op_id in many_to_many_temp(Languages,Operatops_Languages)
        for e in Operatops
        if e.id==op_id]

def task1(one_to_many):
    res_1 = list(filter(lambda x: 'P' in x[3], one_to_many(Languages,Operatops)))
    return res_1
```

```
    return res_1

def task2(one_to_many):
    avg_len = dict()
    res_2=[]
    for link in one_to_many(Languages,Operatops):
        if (link[3] in avg_len):
            avg_len[link[3]].append(link[1])
        else:
            avg_len[link[3]] = [link[1]]
    for key, value in avg_len.items():
        res_2.append((key, round(sum(value) / len(value), 2)))
    return res_2

def task3(many_to_many):
    res_3=[]
    res_4 = list(filter(lambda x: x[0][0] == 'w', many_to_many(many_to_many_temp,Operatops)))
    for i in range(len(res_4)):
        res_3.append((res_4[i][0], res_4[i][2]))
    return res_3


if __name__ == '__main__':
    task1(one_to_many)
    task2(one_to_many)
    task3(many_to_many)
```

Test.py

```
import unittest
import rk1

class testRk(unittest.TestCase):
    def setUp(self):
        self.test1 = [('switch/case', 10, 'Условие', 'Python'),
                      ('while', 5, 'Цикл', 'Python'),
                      ('if', 2, 'Условие', 'Python'),
                      ('Goto', 4, 'Безусловный переход', 'Pascal')]
        self.test2 = [('Python', 5.67),
                      ('C++', 4.0),
                      ('Pascal', 4.0)]
        self.test3 = [('while', 'Python'),
                      ('while', 'C++'),
                      ('while', 'Pascal'),
                      ('while', 'Java'),
                      ('while', 'C'),
                      ('while', 'Delphi')]
    def test1_rk(self):
        self.assertEqual(rk1.task1(rk1.one_to_many),self.test1)
    def test2_rk(self):
        self.assertEqual(rk1.task2(rk1.one_to_many),self.test2)
    def test3_rk(self):
        self.assertEqual(rk1.task3(rk1.many_to_many),self.test3)

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

Результат

```
...
----------------------------------------------------------------------
Ran 3 tests in 0.002s
```