Московский государственный технический университет имени Н.Э.Баумана

Факультет Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

Отчет по рубежному контролю № 2 по курсу Базовые компоненты

Исполнитель
Студент группы РТ5-31Б _____ Платонов А.В.
«__»_____ 2022 г.

Проверил
Доцент кафедры ИУ5 _____ Гапанюк Ю.Е.
«__»_____ 2022 г.

г. Москва – 2022

Задание РК2

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 23. Классы: Язык программирования, Синтаксическая конструкция.

Задания:

1. «Язык программирования» и «Синтаксическая конструкция» связаны соотношением один-ко-многим. Выведите список всех языков, у которых в названии присутствует буква «C», и список его синтаксических конструкций.
2. «Язык программирования» и «Синтаксическая конструкция» связаны соотношением один-ко-многим. Выведите список языков со средним кол-вом букв в названии синтаксических конструкций, отсортированный по среднему кол-ву букв. Среднее кол-во букв в названии должно быть округлено до 2 знаков после запятой.
3. «Язык программирования» и «Синтаксическая конструкция» связаны соотношением многие-ко-многим. Выведите список всех синтаксических конструкций, у которых название начинается с буквы «е», и названия их языков программирования.

Листинг программы, в которой выполняются задания и для которой был проведён рефакторинг (RK1.py)

```
'''Вариант - E, вариант предметной области - 23
("Синтаксическая конструкция - язык программирования")'''
class SynCon:
    # Синтаксическая конструкция
    def __init__(self, id, name, len_con, lan_id):
        self.id = id
        self.name = name
        self.len_con = len_con  # кол-во символов в названии конструкции
        self.lan_id = lan_id
```

```
class LanProg:
    # Язык программирования
    def __init__(self, id, name):
        self.id = id
        self.name = name

class SynLan:
    # 'Синтаксические конструкция языка программирования' для реализации
    # связи многие-ко-многим

    def __init__(self, lan_id, syn_id):
        self.lan_id = lan_id
        self.syn_id = syn_id



# языки программирования
lans = [
    LanProg(1, 'C'),
    LanProg(2, 'C#'),
    LanProg(3, 'C++'),
    LanProg(4, 'Python'),
]

# синтаксические конструкции
syns = [
    SynCon(1, 'if', 2, 2),
    SynCon(2, 'else', 4, 2),
    SynCon(3, 'while', 5, 4),
    SynCon(4, 'for', 3, 4),
    SynCon(5, 'switch', 6, 3),
    SynCon(6, 'case', 4, 3),
    SynCon(6, 'elif', 4, 1)
]
```

```
syns_lans = [
    SynLan(1, 1),
    SynLan(1, 2),
    SynLan(1, 3),
    SynLan(1, 4),
    SynLan(1, 5),
    SynLan(1, 6),
    SynLan(2, 1),
    SynLan(2, 2),
    SynLan(3, 1),
    SynLan(3, 2),
    SynLan(3, 3),
    SynLan(3, 4),
    SynLan(3, 5),
    SynLan(3, 6),
    SynLan(4, 1),
    SynLan(4, 2),
    SynLan(4, 3),
    SynLan(4, 4),
]

one_to_many = [(s.name, s.len_con, l.name)
               for l in lans
               for s in syns
               if s.lan_id == l.id]

# Соединение данных многие-ко-многим
many_to_many_temp = [(l.name, sy.lan_id, sy.syn_id)
                     for l in lans
                     for sy in syns_lans
                     if l.id == sy.lan_id]

many_to_many = [(s.name, s.len_con, lan_name)
                for lan_name, lan_id, syn_id in many_to_many_temp
```

```
            for s in syns if s.id == syn_id]

def task1(one_to_many):
    #print('Задание E1')
    # выбираем языки, в названии которых есть 'C'
    res_1 = list(filter(lambda x: 'C' in x[2], one_to_many))
    return res_1

def task2(one_to_many):
    #print('\nЗадание E2')
    avg_len = dict()
    for link in one_to_many:
        if (link[2] in avg_len):
            avg_len[link[2]].append(link[1])
        else:
            avg_len[link[2]] = [link[1]]
    res_2 = []
    for key, value in avg_len.items():
        res_2.append(tuple([key, round(sum(value) / len(value), 2)]))
    res_2.sort(key=lambda x: x[1])
    return res_2

def task3(many_to_many):
    #print('\nЗадание E3')
    # выбираем синтаксические единицы, которые начинаются с 'e'
    res = list(filter(lambda x: x[0][0] == 'e', many_to_many))
    res_3 = []
    for i in range(len(res)):
        res_3.append(tuple([res[i][0], res[i][2]]))
    return res_3

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

```
    task3(many_to_many)
```

Листинг программы, в которой проводятся тесты (RK2 Platonov RT5-31B.py)

```
import unittest
import RK1

class testRK1(unittest.TestCase):
    def setUp(self):
        self.test1 = [('elif', 4, 'C'), ('if', 2, 'C#'), ('else', 4, 'C#'),
                      ('switch', 6, 'C++'), ('case', 4, 'C++')]
        self.test2 = [('C#', 3.0), ('C', 4.0), ('Python', 4.0), ('C++', 5.0)]
        self.test3 = [('else', 'C'), ('elif', 'C'), ('else', 'C#'),
                      ('else', 'C++'), ('elif', 'C++'), ('else', 'Python')]

    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()
```

Результаты работы программы

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