

Факультет РТ Радиотехнический

Кафедра ИУ5 Системы обработки информации и управления

**Отчет по рубежному контролю № 2 по курсу
Базовые компоненты интернет-технологий
Вариант 23**

Исполнитель

Студент группы РТ5-31Б _____ Яковенко Ю.С.

“ ____ ” _____ 2021 г.

Проверил

Доцент кафедры ИУ5 _____ Гапанюк Ю.Е.

“ ____ ” _____ 2021 г.

Содержание

1. Описание задания.....	3
2. Текст программы.....	3
3. Экранные формы с примерами выполнения программы	7

1. Описание задания

- Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
- Для текста программы рубежного контроля №1 создать модульные тесты с применением TDD - фреймворка (3 теста).

2. Текст программы

main.py

```
class SyncCons:
```

```
    """Синтаксическая конструкция"""
```

```
    def __init__(self, id, name, freq, pl_id):
```

```
        """
```

```
        Args:
```

```
            id (int): id синтаксической конструкции
```

```
            name (str): название конструкции
```

```
            freq (int): средняя частота использования в программе
```

```
            pl_id (int): id языка программирования (programming language)
```

```
        """
```

```
        self.id = id
```

```
        self.name = name
```

```
        self.freq = freq
```

```
        self.pl_id = pl_id
```

```
class PL:
```

```
    """ЯП"""
```

```
    def __init__(self, id, name):
```

```
        """
```

```
        Args:
```

```
            id (int): id ЯП
```

```
            name (str): название ЯП
```

```
        """
```

```
        self.id = id
```

```
        self.name = name
```

```

class ConsLp:
    """
    Конструкции языка (для реализации связи многие-ко-многим)
    """

    def __init__(self, cons_id, pl_id):
        self.cons_id = cons_id
        self.pl_id = pl_id

# Языки программирования
pls = [
    PL(1, 'Java'),
    PL(2, 'JS'),
    PL(3, 'Python'),
    PL(4, 'C++')
]

# Синтаксические конструкции
# Сложно выделить конструкцию, которая есть только в одном языке
cons = [
    SyncCons(1, 'for', 10, 1),
    SyncCons(2, 'while', 5, 3),
    SyncCons(3, 'do while', 3, 4),
    SyncCons(4, 'switch', 7, 4),
    SyncCons(5, 'class', 15, 4),
    SyncCons(6, 'function', 20, 2)
]

cons_lps = [
    ConsLp(1, 1),
    ConsLp(2, 1), # конструкции в языке Java

    ConsLp(1, 2),
    ConsLp(5, 2), # конструкции в языке JS
    ConsLp(6, 2),

    ConsLp(1, 3),
    ConsLp(6, 3), # конструкции в языке Python

    ConsLp(4, 4),
    ConsLp(3, 4), # конструкции в языке C++
]

```

```

# Соединение данных один-ко-многим
one_to_many = [(c.name, c.freq, p.name)
                for c in cons
                for p in pls
                if c.pl_id == p.id]

# Соединение данных многие-ко-многим
many_to_many_temp = [(pl.name, cl.pl_id, cl.cons_id)
                      for pl in pls
                      for cl in cons_lps
                      if pl.id == cl.pl_id]

many_to_many = [(c.name, c.freq, pl_name)
                 for pl_name, pl_id, cons_id in many_to_many_temp
                 for c in cons if c.id == cons_id]

def t1():
    # Задание E1
    """
        «ЯП» и «Синтаксическая конструкция» связаны соотношением один-ко-многим.
        Выведите список всех конструкций, у которых в названии присутствует слово while,
        и список языков программирования, где есть
        данная конструкция.
    """

    return list(filter(lambda i: "while" in i[0], one_to_many))

def t2():
    # Задание E2
    """
        «ЯП» и «Синтаксическая конструкция» связаны соотношением один-ко-многим.
        Выведите список языков программирования со средней частотой встречаемости
        данных конструкций, отсортированный по средней частоте встречаемости.
    """

    dictOfCons = {}

    for item in one_to_many: # создание словаря (язык программирования : список с
                             частотой встречаемости всех конструкций в этом языке)
        if (item[2] in dictOfCons):
            dictOfCons[item[2]].append(item[1])
        else:

```

```

dictOfCons[item[2]] = [item[1]]

return sorted([(key, round(sum(dictOfCons[key]) / len(dictOfCons[key]), 2))
               for key in dictOfCons.keys()], key=lambda x: x[1])

def t3():
    # Задание Е3
    """
    «ЯП» и «Синтаксическая конструкция» связаны соотношением многие-ко-многим.
    Выведите список всех конструкций, у которых частота встречаемости больше 25, и
    названия языков программирование, в которых они встречаются.
    """

    d = {}

    """
    создание словаря
    конструкция языка: {
        общая частота встречаемости во всех языках программирования : (int),
        список с языками программирования, где конструкция встречается : ([str])
    }
    """

    for item in many_to_many:
        if (item[0] in d):
            d[item[0]]["number"] += item[1]
            d[item[0]]["PLs"].append(item[2])
        else:
            d[item[0]] = {
                "number": item[1],
                "PLs": [item[2]]
            }

    return [(key, d[key]["PLs"])
            for key in d.keys()
            if d[key]["number"] > 25
            ]

```

test.py

```
import unittest
import main


class Test(unittest.TestCase):
    def test_t1(self):
        self.assertEqual(main.t1(), [('while', 5, 'Python'), ('do while', 3, 'C++')])

    def test_tk2(self):
        self.assertEqual(main.t2(), [('Python', 5.0), ('C++', 8.33), ('Java', 10.0), ('JS', 20.0)])

    def test_tk3(self):
        self.assertEqual(main.t3(), [('for', ['Java', 'JS', 'Python']), ('function', ['JS', 'Python'])])

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

3. Экранные формы с примерами выполнения программы



```
-----
Ran 3 tests in 0.000s

OK
```