ИУ5-32Б,

Турдиев Ихтияр

Рубежный контроль 2

Условия:

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).

Текст программы

Main.py

```
from operator import itemgetter


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


class Operator:
    def __init__(self, id, name, ascii_code,prog_lang_id):
        self.id = id
        self.name = name
        self.ascii_code = ascii_code
        self.prog_lang_id = prog_lang_id


class ProgLangOp:
    def __init__(self, prog_lang_id, op_id):
        self.prog_lang_id = prog_lang_id
        self.op_id = op_id


prog_langs = [
    ProgramLang(1, "C++"),
    ProgramLang(2, "Java"),
    ProgramLang(3, "Python"),
]

ops = [
    Operator(1, "Plus", 43,1),
    Operator(2, "Minus", 45,2),
    Operator(3, "Equal", 61,3),
    Operator(4, "Ampersand", 38, 3),
    Operator(5, "Divide", 47, 1),
    Operator(5, "Multiply", 42, 1)
```

```
]

pl_ops = [
    ProgLangOp(1, 1),
    ProgLangOp(2, 2),
    ProgLangOp(3, 3),
    ProgLangOp(3, 4),
    ProgLangOp(1, 5),
]


def first_task(op_list):
    res_1 = sorted(op_list, key=itemgetter(0))
    return res_1


def second_task(op_list):
    res_2 = []
    temp_dict = dict()
    for i in op_list:
        if i[2] in temp_dict:
            temp_dict[i[2]] += 1
        else:
            temp_dict[i[2]] = 1
    for i in temp_dict.keys():
        res_2.append((i, temp_dict[i]))

    res_2.sort(key=itemgetter(1), reverse=True)
    return res_2


def third_task(op_list, end_ch):
    res_3 = [(i[0], i[2]) for i in op_list if i[0].endswith(end_ch)]
    return res_3


def main():
    one_to_many = [(op.name, op.ascii_code, pl.name)
                   for pl in prog_langs
                   for op in ops
                   if op.prog_lang_id == pl.id]

    many_to_many_temp = [(pl.name, ps.prog_lang_id, ps.op_id)
                         for pl in prog_langs
                         for ps in pl_ops
                         if ps.prog_lang_id == pl.id]

    many_to_many = [(op.name, op.ascii_code, pl_name)
                    for pl_name, pl_id, op_id in many_to_many_temp
                    for op in ops if op.id == op_id]

    print('Задание Б1')
    print(first_task(one_to_many))

    print("\nЗадание Б2")
    print(second_task(one_to_many))
```

```
    print("\nЗадание Б3")
    print(third_task(many_to_many, 's'))


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

unit_tests.py

```
import main
from operator import itemgetter
import unittest


class TestMainMethods(unittest.TestCase):
    def test_first_task_method(self):
        test_list = [('second', 'second', 'second'), ('first', 'first', 'first'), ('third', 'third', 'third')]
        result = main.first_task(test_list)
        reference = sorted(test_list, key=itemgetter(0))
        self.assertEqual(result, reference)

    def test_second_task_method(self):
        test_list = [('A. Mercer', 120000, 'Resource department'), ('R. Gosling', 110000, 'Archive
department'),
                     ('E. Yeger', 80000, 'Resource department'), ('C. Nolan', 130000, 'Logistic department')]
        result = main.second_task(test_list)
        reference = [('Resource department', 2), ('Archive department', 1), ('Logistic department', 1)]
        self.assertEqual(result, reference)

    def test_third_method(self):
        test_list = [('A. Mercer', 120000, 'Resource department'), ('R. Gosling', 110000, 'Archive
department'),
                     ('E. Yeger', 80000, 'Resource department'), ('C. Nolan', 130000, 'Logistic department')]
        result = main.third_task(test_list, 'r')
        reference = [('A. Mercer', 'Resource department'), ('E. Yeger', 'Resource department')]
        self.assertEqual(result, reference)
```