Студент: Пустовалов Григорий
Группа: ИУ5-32Б
Вариант: 20Б

Отчет по рубежному контролю №2

Условие РК1:
1. «Деталь» и «Поставщик» связаны соотношением один-ко-многим. Выведите список всех связанных поставщиков и деталей, отсортированный по поставщикам, сортировка по деталям произвольная.
2. «Деталь» и «Поставщик» связаны соотношением один-ко-многим. Выведите список деталей с количеством поставщиков у каждой детали, отсортированный по количеству поставщиков.
3. «Деталь» и «Поставщик» связаны соотношением многие-ко-многим. Выведите список всех поставщиков, у которых фамилия заканчивается на «ов», и названия их деталей.

Условие РК2:
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:

файл provider.py
```
class Provider:
    """Поставщик"""
    def __init__(self, id, fio, sal, det_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.det_id = det_id
```

файл detail.py
```
class Detail:
    """Деталь"""

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

файл detPro.py
```
class DetPro:
    """
    'Поставка детали' для реализации
    связи многие-ко-многим
    """
    def __init__(self, det_id, pro_id):
```

```
        self.det_id = det_id
        self.pro_id = pro_id
```

файл arrays.py

```
from detail import Detail
from provider import Provider
from detPro import DetPro

providers = [
    Provider(1, 'Сидоров', 25000, 3),
    Provider(2, 'Петров', 55000, 2),
    Provider(3, 'Васькин', 10000, 1),
    Provider(4, 'Иванов', 35000, 3),
    Provider(5, 'Ефимов', 95000, 3),
    Provider(6, 'Павлов', 999, 2),
]



dets = [
    Detail(1, 'вал'),
    Detail(2, 'гайка'),
    Detail(3, 'корпус'),

    Detail(4, 'вал (другой)'),
    Detail(5, 'гайка (другой)'),
    Detail(6, 'корпус (другой)'),
]

prov_dets = [
    DetPro(1, 1),
    DetPro(2, 2),
    DetPro(3, 3),
    DetPro(3, 4),
    DetPro(3, 5),

    DetPro(11, 1),
    DetPro(22, 1),
    DetPro(33, 3),
    DetPro(33, 3),
    DetPro(33, 5),
]
```

файл main.py (измененный)

```
from operator import itemgetter
from detail import Detail
from provider import Provider
from detPro import DetPro
from arrays import dets
from arrays import providers
from arrays import prov_dets

def task1(one_to_many):
    try:
        res_1 = sorted(one_to_many, key=itemgetter((0)))
        print('Задание Б1')
        print('№|Фамилия|ЗП|название детали')
        print('_____')
        for num, i in enumerate(res_1):
```

```
            print("{}|{}|{}|{}|{}".format(num + 1, i[0], i[1], i[2], i[3])) #сортировка по фамилии [1]
        print()
        return res_1
    except(TypeError):
        raise TypeError('выражение должно быть типа list[tuple[str, int, int, str]]')

def task2(one_to_many):
    res_2 = []
    for i in dets:
        detailsLambda = list(filter(lambda j: j[3] == i.name, one_to_many))
        if len(detailsLambda) > 0:
            res_2.append((i.name, len(detailsLambda)))
    res_2 = sorted(res_2, key = itemgetter(1), reverse=True)

    print('Задание Б2')
    print('№|Название детали| количество поставщиков')
    print('_____________________________________________')
    for num, i in enumerate(res_2):
        print("{}|{}|{}".format(num + 1, i[0], i[1]))
    print()
    return res_2


def task3(many_to_many):
    res_3 = []
    for i in many_to_many:
        if (i[0][-2] == "о") and (i[0][-1] == "в"):
            res_3.append(i)

    print('Задание Б3')
    print('№|Поставщик| название детали')
    print('_____________________________________________')
    for num, i in enumerate(res_3):
        print("{}|{}|{}".format(num + 1, i[0], i[2]))
    return res_3

def main():
    """Основная функция"""
    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.det_id, ed.pro_id)
                         for d in dets
                         for ed in prov_dets
                         if d.id == ed.pro_id
                         ]

    many_to_many = [(e.fio, e.sal, dep_name)
                    for dep_name, det_id, pro_id in many_to_many_temp
                    for e in providers
                    if e.id == pro_id
                    ]

    # Соединение данных один-ко-многим
    one_to_many = [(e.fio, e.sal, e.det_id, d.name)
                   for d in dets
```

```
                for e in providers
                if e.det_id == d.id
                ]

    task1(one_to_many)
    task2(one_to_many)
    task3(many_to_many)


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

файл test.py

```
from unittest import TestCase, main
from main import task1, task2, task3

class testrk2(TestCase):
    def test_task1(self):
        self.assertEqual(task1(
            [('Васькин', 10000, 1, 'вал'),
            ('Петров', 55000, 2, 'гайка'),
            ('Павлов', 999, 2, 'гайка'),
            ('Сидоров', 25000, 3, 'корпус'),
            ('Иванов', 35000, 3, 'корпус'),
            ('Ефимов', 95000, 3, 'корпус')]),

            [('Васькин', 10000, 1, 'вал'),
            ('Ефимов', 95000, 3, 'корпус'),
            ('Иванов', 35000, 3, 'корпус'),
            ('Павлов', 999, 2, 'гайка'),
            ('Петров', 55000, 2, 'гайка'),
            ('Сидоров', 25000, 3, 'корпус')])

    def test_task12(self):
        with self.assertRaises(TypeError) as e:
            task1(['str', 2.4, 'str', 1])
        self.assertEqual('выражение должно быть типа list[tuple[str, int, int, str]]', e.exception.args[0])

    def test_task2(self):
         self.assertEqual(task2(
            [('Васькин', 10000, 1, 'вал'),
            ('Петров', 55000, 2, 'гайка'),
            ('Павлов', 999, 2, 'гайка'),
            ('Сидоров', 25000, 3, 'корпус'),
            ('Иванов', 35000, 3, 'корпус'),
            ('Ефимов', 95000, 3, 'корпус')]),

            [('корпус', 3),
            ('гайка', 2),
            ('вал', 1)])

    def test_task3(self):
```

```
        self.assertEqual(task3(
            [('Сидоров', 25000, 'вал'),
            ('Сидоров', 25000, 'вал'),
            ('Сидоров', 25000, 'вал'),
            ('Петров', 55000, 'гайка'),
            ('Васькин', 10000, 'корпус'),
            ('Васькин', 10000, 'корпус'),
            ('Васькин', 10000, 'корпус'),
            ('Иванов', 35000, 'вал (другой)'),
            ('Ефимов', 95000, 'гайка (другой)'),
            ('Ефимов', 95000, 'гайка (другой)')]),

            [('Сидоров', 25000, 'вал'),
            ('Сидоров', 25000, 'вал'),
            ('Сидоров', 25000, 'вал'),
            ('Петров', 55000, 'гайка'),
            ('Иванов', 35000, 'вал (другой)'),
            ('Ефимов', 95000, 'гайка (другой)'),
            ('Ефимов', 95000, 'гайка (другой)')])

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