МГТУ им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Рубежный контроль №2

«Базовые компоненты интернет-технологий»

Студент группы ИУ5-31Б: Преподаватель кафедры ИУ5:

Строганов Георгий Константинович Гапанюк Юрий Евгеньевич

Москва, 2022

**Условия РК-1**

**Вариант А. Предметная область 20.**

1. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
2. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
3. «Поставщик» и «Деталь» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «ООО.», и список работающих в них сотрудников.

**Условия РК-2**

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).

**Листинг программы**

**RK\_2.py**

# -\*- coding: cp1251 -\*-

from operator import itemgetter

class Detail:

#Деталь

def \_\_init\_\_(self, id, name, cost, count\_det, det\_id):

self.id = id

self.name = name

self.cost = cost

self.det\_id = det\_id

self.count = count\_det

class Provider:

#Поставщик

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class DetProv:

#'Поставляемые поставщиком детали' для реализации связи многие-ко-многим

def \_\_init\_\_(self, dep\_id, det\_id):

self.dep\_id = dep\_id

self.det\_id = det\_id

# Поставщики

Providers = [

Provider(1, 'OOO."Рога и копыта"'),

Provider(2, 'OOO."Моя оборона"'),

Provider(3, 'ИП."Рога и подковы"'),

]

# Детали

Details = [

Detail(1, "Болт", 250, 300, 1),

Detail(2, "Саморез", 375, 200, 1),

Detail(3, "Вкладыш", 450, 50, 2),

Detail(4, "Бобышка", 350, 60, 2),

Detail(5, "Штуцер", 500, 100, 3),

]

# Связи деталей и поставщиков

Detail\_Provider = [

DetProv(1,1),

DetProv(2,2),

DetProv(3,3),

]

# Все связи

def connections():

# Соединение данных один-ко-многим

one\_to\_many = [(e.name, e.cost, d.name, e.count)

for d in Providers

for e in Details

if e.det\_id == d.id]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(d.name, ed.dep\_id, ed.det\_id)

for d in Providers

for ed in Detail\_Provider

if d.id == ed.dep\_id]

many\_to\_many = [(e.name, e.cost, dep\_name)

for dep\_name, dep\_id, det\_id in many\_to\_many\_temp

for e in Details if e.id == det\_id]

return one\_to\_many, many\_to\_many

# Вывод результатов заданий

def Example\_1(one\_to\_many):

return sorted(one\_to\_many, key = itemgetter(2))

def Example\_2(one\_to\_many):

res\_12\_unsorted = []

# Перебираем всех поставщиков

for d in Providers:

# Список деталей поставщиков

d\_details = list(filter(lambda i: i[2] == d.name, one\_to\_many))

if len(d\_details) > 0:

# Стоимоть деталей у постащика

d\_sals = [cost\*count for \_,cost,\_,count in d\_details]

# Суммарная стоимость деталей у поставщика

d\_sals\_sum = sum(d\_sals)

res\_12\_unsorted.append((d.name, d\_sals\_sum))

# Сортировка по стоимости имеющихся деталей

return sorted(res\_12\_unsorted, key = itemgetter(1), reverse = True)

def Example\_3(one\_to\_many):

res\_13 = {}

# Перебираем всех поставщиков (будем проверять наличие слова: "OOO.")

for d in Providers:

if 'OOO.' in d.name:

d\_detailz = list(filter(lambda i: i[2] == d.name, one\_to\_many))

if len(d\_detailz) > 0:

# Только название деталей

d\_detailz\_names = [x for x,\_,\_,\_ in d\_detailz]

# Добавляем результат в словарь

# ключ - поставщик, значение - список деталей

res\_13[d.name] = d\_detailz\_names

return res\_13

one\_to\_many, many\_to\_many = connections()

def main():

print('Example A1')

print(Example\_1(one\_to\_many))

print('Example A2')

print(Example\_2(one\_to\_many))

print('Example A3')

print(Example\_3(one\_to\_many))

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Test\_TDD\_RK\_2.py**

# -\*- coding: cp1251 -\*-

import pytest

from RK\_2 import Example\_1

from RK\_2 import Example\_2

from RK\_2 import Example\_3

from RK\_2 import one\_to\_many

#Разделил на элементы для удобства чтения

def test\_result\_example\_1():

temp = Example\_1(one\_to\_many)

assert temp[0] == ('Вкладыш', 450, 'OOO."Моя оборона"', 50)

assert temp[1] == ('Бобышка', 350, 'OOO."Моя оборона"', 60)

assert temp[2] == ('Болт', 250, 'OOO."Рога и копыта"', 300)

assert temp[3] == ('Саморез', 375, 'OOO."Рога и копыта"', 200)

assert temp[4] == ('Штуцер', 500, 'ИП."Рога и подковы"', 100)

def test\_result\_example\_2():

temp = Example\_2(one\_to\_many)

assert temp[0] == ('OOO."Рога и копыта"', 150000)

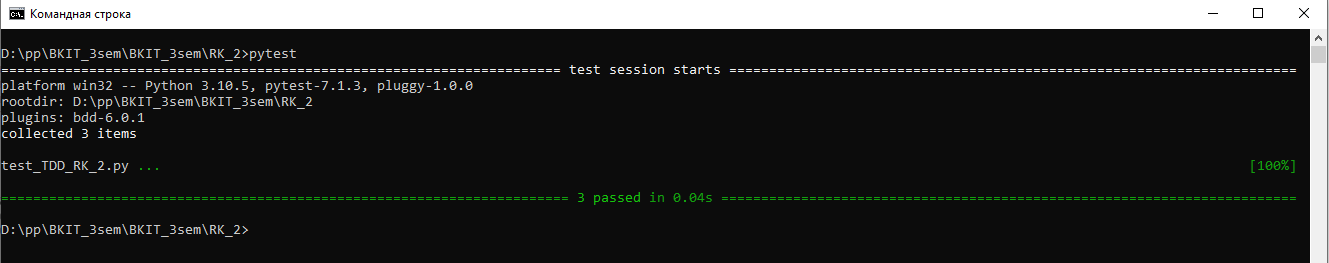
assert temp[1] == ('ИП."Рога и подковы"', 50000)

assert temp[2] == ('OOO."Моя оборона"', 43500)

def test\_result\_example\_3():

temp = Example\_3(one\_to\_many)

assert temp == {'OOO."Рога и копыта"': ['Болт', 'Саморез'], 'OOO."Моя оборона"': ['Вкладыш', 'Бобышка']}

**Результаты выполнения модульного тестирования:**