**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Рубежный контроль №2

Вариант Г №19

Выполнил:

Студент ИУ5-34Б

Флоринский В. А.

Подпись и дата:

Проверил:

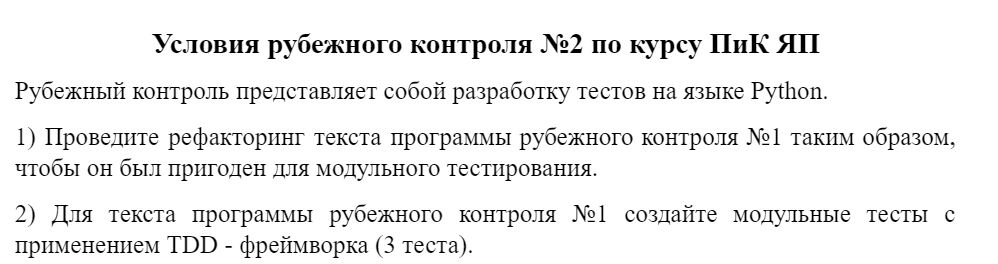
Преподаватель каф. ИУ5

Гапанюк Ю. Е.

Подпись и дата:

Москва, 2023 г.

**Задание.**

****

**Текст программ.**

**main.py:**

from prettytable import PrettyTable

from operator import itemgetter

class Detail:

    def \_\_init\_\_(self, id, name, price, sup\_id):

        self.id = id

        self.name = name

        self.price = price

        self.sup\_id = sup\_id

class Suppliers:

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class DetSup:

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

        self.sup\_id = sup\_id

        self.det\_id = det\_id

def a1\_solution(one\_to\_many):

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

    ans = []

    for el in arr:

        if el[2][0] == 'А':

            ans.append(el)

    if len(ans) != 0 : return (ans)

    else : return "Отделов на А нет!"

def a2\_solution(one\_to\_many):

    arr0 = []

    for d in suppliers:

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

        if len(d\_dets) > 0:

            d\_prices = [price for \_,price,\_ in d\_dets]

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

            arr0.append((d.name, d\_sals\_sum))

    res = sorted(arr0, key=itemgetter(1), reverse=True)

    return res

def a3\_solution(many\_to\_many):

    arr = {}

    for d in suppliers:

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

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

        arr[d.name] = d\_emps\_names

    arr = dict(sorted(arr.items()))

    return arr

suppliers = [

    Suppliers(1, "Звезда-Техника"),

    Suppliers(2, "Сибирские Детали"),

    Suppliers(3, "Волжский Агрекат"),

    Suppliers(4, "АгроПром Комбинат"),

]

details = [

    Detail(1, "Винт", 1500, 1),

    Detail(2, "Гайка", 1700, 2),

    Detail(3, "Гайка", 1900, 3),

    Detail(4, "Шайба",2000, 4),

    Detail(5, "Шплинт", 2100, 4),

]

detsup = [

    DetSup(1, 1),

    DetSup(1, 2),

    DetSup(2, 2),

    DetSup(2, 3),

    DetSup(3, 1),

    DetSup(3, 4),

    DetSup(4, 2),

    DetSup(4, 5),

]

def main():

    one\_to\_many = [(p.name, p.price, d.name)

                   for d in suppliers

                   for p in details

                   if p.sup\_id == d.id]

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

                         for d in suppliers

                         for ed in detsup

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

    many\_to\_many = [(e.name, e.price, sup\_name)

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

                    for e in details

                    if e.id == det\_id]

    print('Задание А1')

    print(a1\_solution(one\_to\_many))

    print('\nЗадание А2')

    print(a2\_solution(one\_to\_many))

    print('\nЗадание А3')

    print(a3\_solution(many\_to\_many))

main()

**test.py:**

import unittest

from main import \*

class Test\_Program(unittest.TestCase):

    suppliers = [

    Suppliers(1, "Звезда-Техника"),

    Suppliers(2, "Сибирские Детали"),

    Suppliers(3, "Волжский Агрекат"),

    Suppliers(4, "АгроПром Комбинат"),

    ]

    details = [

        Detail(1, "Винт", 1500, 1),

        Detail(2, "Гайка", 1700, 2),

        Detail(3, "Гайка", 1900, 3),

        Detail(4, "Шайба",2000, 4),

        Detail(5, "Шплинт", 2100, 4),

    ]

    detsup = [

        DetSup(1, 1),

        DetSup(1, 2),

        DetSup(2, 2),

        DetSup(2, 3),

        DetSup(3, 1),

        DetSup(3, 4),

        DetSup(4, 2),

        DetSup(4, 5),

    ]

    def test\_a1(self):

        one\_to\_many = [(p.name, p.price, d.name)

                   for d in suppliers

                   for p in details

                   if p.sup\_id == d.id]

        self.assertEqual(a1\_solution(one\_to\_many),

                        [('Шайба', 2000, 'АгроПром Комбинат'),

                         ('Шплинт', 2100, 'АгроПром Комбинат')])

    def test\_a2(self):

        one\_to\_many = [(p.name, p.price, d.name)

                   for d in suppliers

                   for p in details

                   if p.sup\_id == d.id]

        self.assertEqual(a2\_solution(one\_to\_many),

                         [('АгроПром Комбинат', 4100),

                          ('Волжский Агрекат', 1900),

                          ('Сибирские Детали', 1700),

                          ('Звезда-Техника', 1500)])

    def test\_a3(self):

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

                         for d in suppliers

                         for ed in detsup

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

        many\_to\_many = [(e.name, e.price, sup\_name)

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

                    for e in details

                    if e.id == det\_id]

        self.assertEqual(a3\_solution(many\_to\_many),

                         {'АгроПром Комбинат': ['Гайка', 'Шплинт'],

                          'Валжский Агрекат': ['Винт', 'Шайба'],

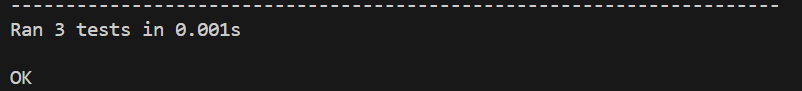
                          'Звезда-Техника': ['Винт', 'Гайка'],

                          'Сибирские Детали': ['Гайка', 'Гайка']})

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

    unittest.main()

**Результаты Тестов.**

****

Если при проверки функции a3\_solution намеренно сделать ошибку в тестах, то в результатах теста увидим ошибку:

