**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» Отчет по РК№2

## Вариант 21

Выполнил: Проверил:

студент группы РТ5-31Б преподаватель каф.

ИУ5

Росин С. М. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2023г.

# Текст программы

Main.py

from operator import itemgetter

class Detail:

def \_\_init\_\_(self, id, name, weight, comp\_id):

self.id = id

self.name = name

self.weight = weight

self.comp\_id = comp\_id

class Manufacturer:

def \_\_init\_\_(self, detail\_id, name):

self.id = detail\_id

self.name = name

class ManufacturerDetail:

def \_\_init\_\_(self, manufact\_id, detail\_id):

self.manufact\_id = manufact\_id

self.detail\_id = detail\_id

def create\_manufacturers():

return [

Manufacturer(1, 'Ява'),

Manufacturer(2, 'Волга'),

Manufacturer(3, 'ВАЗ'),

Manufacturer(4, 'Тройка'),

Manufacturer(5, 'Мальборо')

]

def create\_details():

return [

Detail(1, 'болт', 173, 1),

Detail(2, 'винт', 140, 2),

Detail(3, 'штуцер', 97, 3),

Detail(4, 'шуруп', 201, 4)

]

def create\_manufacturer\_details():

return [

ManufacturerDetail(1, 1),

ManufacturerDetail(1, 2),

ManufacturerDetail(2, 2),

ManufacturerDetail(2, 4),

ManufacturerDetail(3, 3),

ManufacturerDetail(4, 2),

ManufacturerDetail(4, 4),

ManufacturerDetail(5, 2),

ManufacturerDetail(5, 3),

ManufacturerDetail(5, 4)

]

def get\_one\_to\_many(details, manufacturers):

return [(d.name, d.weight, m.name)

for d in details

for m in manufacturers

if d.id == m.id

]

def get\_many\_to\_many(one\_to\_many, manufacturer\_details):

many\_to\_many\_temp = [(m.name, md.manufact\_id, md.detail\_id)

for m in manufacturers

for md in manufacturer\_details

if m.id == md.manufact\_id

]

return [(d.name, d.weight, manufact\_name)

for manufact\_name, manufact\_id, detail\_id in many\_to\_many\_temp

for d in details if d.id == detail\_id

]

def task\_a2(one\_to\_many):

res\_2\_unsorted = []

for m in set(item[2] for item in one\_to\_many):

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

if len(m\_details) > 0:

m\_weights = [weight for \_, weight, \_ in m\_details]

m\_weights\_sum = sum(m\_weights)

res\_2\_unsorted.append((m, m\_weights\_sum))

return sorted(res\_2\_unsorted, key=itemgetter(1), reverse=True)

def task\_a3(many\_to\_many):

res\_3 = {}

for m in set(item[2] for item in many\_to\_many):

if 'о' in m:

m\_details = list(filter(lambda i: i[2] == m, many\_to\_many))

m\_details\_names = [x for x, \_, \_ in m\_details]

res\_3[m] = m\_details\_names

return res\_3

def main():

details = create\_details()

manufacturers = create\_manufacturers()

manufacturer\_details = create\_manufacturer\_details()

one\_to\_many = get\_one\_to\_many(details, manufacturers)

many\_to\_many = get\_many\_to\_many(one\_to\_many, manufacturer\_details)

print('Задание A1')

res\_1 = sorted(one\_to\_many, key=itemgetter(2))

print(res\_1)

print('Задание A2')

res\_2 = task\_a2(one\_to\_many)

print(res\_2)

print('Задание А3')

res\_3 = task\_a3(many\_to\_many)

print(res\_3)

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

main()

# Программа тестов

Testing.py

import unittest

from your\_program import (create\_manufacturers, create\_details, create\_manufacturer\_details,

get\_one\_to\_many, get\_many\_to\_many, task\_a2, task\_a3)

class TestYourProgram(unittest.TestCase):

def setUp(self):

self.details = create\_details()

self.manufacturers = create\_manufacturers()

self.manufacturer\_details = create\_manufacturer\_details()

self.one\_to\_many = get\_one\_to\_many(self.details, self.manufacturers)

self.many\_to\_many = get\_many\_to\_many(self.one\_to\_many, self.manufacturer\_details)

def test\_get\_one\_to\_many(self):

expected\_result = [('болт', 173, 'Ява'), ('винт', 140, 'Волга'), ('штуцер', 97, 'ВАЗ'), ('шуруп', 201, 'Тройка')]

self.assertEqual(get\_one\_to\_many(self.details, self.manufacturers), expected\_result)

def test\_task\_a2(self):

expected\_result = [('Волга', 341), ('Тройка', 341), ('Ява', 173), ('ВАЗ', 97)]

self.assertEqual(task\_a2(self.one\_to\_many), expected\_result)

def test\_task\_a3(self):

expected\_result = {'Волга': ['винт', 'шуруп'], 'Ява': ['болт'], 'ВАЗ': ['штуцер']}

self.assertEqual(task\_a3(self.many\_to\_many), expected\_result)

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

unittest.main()

# Результаты Тестов

