# Московский государственный технический университет им. Н.Э. Баумана

## Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

## Курс «Базовые компоненты интернет-технологий» Отчет по рубежному контролю №2

Выполнил: Проверил:

студент группы ИУ5-31Б преподаватель каф. ИУ5

Абуховский Иван Александрович

Гапанюк Юрий Евгеньевич

Подпись: Подпись:

Дата: Дата:

Москва, 2021 г.

# Рубежный контроль №2 Задание

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

**AbukhovskiyRK1IU5\_31B.py**

from operator import itemgetter

class Student:

"""Студент"""

def \_\_init\_\_(self, id, surname, dolgi, group\_id):

self.id = id

self.surname = surname

self.dolgi = dolgi

self.group\_id = group\_id

class Group:

"""Группа"""

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

self.id = id

self.name = name

class StudentGroup:

"""

'Студенты группы' для реализации

связи многие-ко-многим

"""

def \_\_init\_\_(self, group\_id, student\_id):

self.group\_id = group\_id

self.student\_id = student\_id

# Группы

groups = [

Group(1, 'ИУ5-31Б'),

Group(2, 'ИУ6-12Б'),

Group(3, 'ИУ7-53'),

Group(11, 'МТ5-11'),

Group(22, 'РК5-32Б'),

Group(33, 'СМ5-13'),

]

# Студенты

students = [

Student(1, 'Абуховский', 25, 1),

Student(2, 'Черноморец', 47, 2),

Student(3, 'Шагиахметов', 39, 3),

Student(4, 'Стельмах', 81, 4),

Student(5, 'Сыса', 17, 5),

]

students\_groups = [

StudentGroup(1,1),

StudentGroup(2,2),

StudentGroup(2,3),

StudentGroup(3,4),

StudentGroup(3,5),

StudentGroup(11,1),

StudentGroup(22,2),

StudentGroup(22,3),

StudentGroup(33,4),

StudentGroup(33,5),

]

def task\_a1(one\_to\_many):

one\_to\_many = one\_to\_many

return [i for i in sorted(one\_to\_many, key=itemgetter(2))]

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

res\_2\_unsorted = []

# Перебираем все группы

for g in groups:

# Список студентов группы

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

# Если класс не пустой

if len(g\_students) > 0:

# Долги за обучение студентов группы

g\_dolgii = [dolgi for \_, dolgi, \_ in g\_students]

# Суммарный долг студентов группы

g\_dolgii\_sum = sum(g\_dolgii)

res\_2\_unsorted.append((g.name, g\_dolgii\_sum))

# Сортировка по суммарному долгу

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

return res\_2

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

res\_3 = {}

# Перебираем все группы

for g in groups:

if 'Б' in g.name:

# Список студентов группы

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

# Только фамилии студентов

g\_students\_surnames = [x for x, \_, \_ in g\_students]

# Добавляем результат в словарь

# ключ - группа, значение - список фамилий

res\_3[g.name] = g\_students\_surnames

return res\_3

def main():

"""Основная функция"""

# Соединение данных один-ко-многим

one\_to\_many = [(s.surname, s.dolgi, g.name)

for g in groups

for s in students

if s.group\_id==g.id]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(g.name, sg.group\_id, sg.student\_id)

for g in groups

for sg in students\_groups

if g.id==sg.group\_id]

many\_to\_many = [(s.surname, s.dolgi, group\_name)

for group\_name, group\_id, student\_id in many\_to\_many\_temp

for s in students if s.id==student\_id]

print('Задание А1')

print(task\_a1(one\_to\_many))

print('\nЗадание А2')

print(task\_a2(one\_to\_many))

print('\nЗадание А3')

print(task\_a3(many\_to\_many))

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

main()

**test.py**

from AbukhovskiyRK1IU5\_31B import \*

import unittest

class Test(unittest.TestCase):

def setUp(self):

self.groups = [

Group(1, 'ИУ5-31Б'),

Group(2, 'ИУ6-12Б'),

Group(3, 'ИУ7-53'),

Group(11, 'МТ5-11'),

Group(22, 'РК5-32Б'),

Group(33, 'СМ5-13'),

]

self.students = [

Student(1, 'Абуховский', 25, 1),

Student(2, 'Черноморец', 47, 2),

Student(3, 'Шагиахметов', 39, 3),

Student(4, 'Стельмах', 81, 4),

Student(5, 'Сыса', 17, 5),

]

self.students\_groups = [

StudentGroup(1,1),

StudentGroup(2,2),

StudentGroup(2,3),

StudentGroup(3,4),

StudentGroup(3,5),

StudentGroup(11,1),

StudentGroup(22,2),

StudentGroup(22,3),

StudentGroup(33,4),

StudentGroup(33,5),

]

self.one\_to\_many = [(s.surname, s.dolgi, g.name)

for g in groups

for s in students

if s.group\_id==g.id]

self.many\_to\_many\_temp = [(g.name, sg.group\_id, sg.student\_id)

for g in groups

for sg in students\_groups

if g.id==sg.group\_id]

self.many\_to\_many = [(s.surname, s.dolgi, group\_name)

for group\_name, group\_id, student\_id in many\_to\_many\_temp

for s in students if s.id==student\_id]

def test\_task\_a1(self):

prediction = [('Абуховский', 25, 'ИУ5-31Б'), ('Черноморец', 47, 'ИУ6-12Б'), ('Шагиахметов', 39, 'ИУ7-53')]

self.assertEqual(task\_a1(self.one\_to\_many), prediction)

def test\_task\_a2(self):

prediction = [('ИУ6-12Б', 47), ('ИУ7-53', 39), ('ИУ5-31Б', 25)]

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

def test\_task\_a3(self):

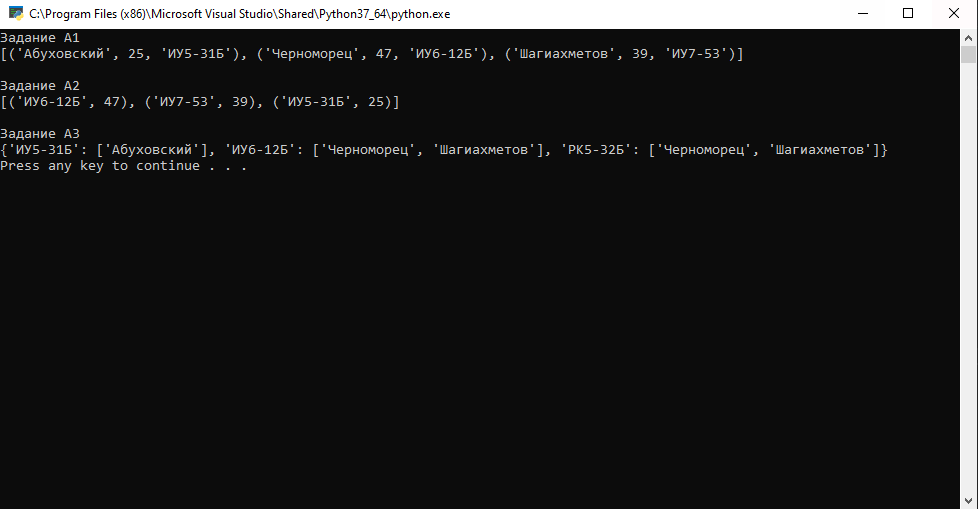
prediction = {'ИУ5-31Б': ['Абуховский'], 'ИУ6-12Б': ['Черноморец', 'Шагиахметов'], 'РК5-32Б': ['Черноморец', 'Шагиахметов']}

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

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

unittest.main()

# Результаты выполнения программы



### Программа была отредактирована так, чтобы результаты ее выполнения не изменились, но ее можно было модульно её протестировать.

### 

### (Тесты запускал в Python Shell, а не в Visual Studio, для удобства просмотра результатов и редактирования).

### Тестирование показывает, что программа работает исправно.

### Если поменять ожидаемый результат в тестах, то результат тестирования укажет на несоответствие полученного результата ожидаемому.

### 