**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 ИУ5-34Б |  |
| Братухин Александр |  |
| Подпись и дата:  13.12.21 | Подпись и дата: |

Москва, 2021г.

# Постановка задачи

Рубежный контроль представляет собой разработку тестов на языке Python.

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).

# Файл rk.py:

from operator import itemgetter

class Pupil:

    """Ученик"""

    def \_\_init\_\_(self, id, fio, average\_score, clas\_id):

        self.id = id

        self.fio = fio

        self.avr = average\_score

        self.clas\_id = clas\_id

class Clas:

    """Номер класса"""

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class PupClass:

    """Соединение для связи многие ко многим"""

    def \_\_init\_\_(self, clas\_id, pupil\_id):

        self.clas\_id = clas\_id

        self.pupil\_id = pupil\_id

clases = [Clas(1, 'А1'),

    Clas(2, 'Б2'),

    Clas(3, 'В3'),

    Clas(11, '1 класс (продленка)'),

    Clas(22, '2 класс (продленка)'),

    Clas(33, '3 класс (продленка)')]

pupils = [Pupil(1, 'Сиденко', 4.5, 1),

    Pupil(2, 'Волков', 3.75, 3),

    Pupil(3, 'Ананьев', 5.0, 2),

    Pupil(4, 'Варина', 4.33, 1),

    Pupil(5, 'Верешко', 4.69, 3)]

pupclases = [PupClass(1,1),

    PupClass(3,2),

    PupClass(2,3),

    PupClass(1,4),

    PupClass(3,5),

    PupClass(11,4),

    PupClass(22,1),

    PupClass(33,5),

    PupClass(22,3),

    PupClass(33,2)]

def G1(one\_to\_many):

    res\_11 = [(cl.name, list(fio for fio, avr, name in one\_to\_many if name == cl.name))

            for cl in clases if cl.name[0] == 'А']

    return res\_11

def G2(one\_to\_many):

    res\_12\_unsorted = []

    # Перебираем все классы

    for cl in clases:

        # Список учеников в данном классе

        clpupils = list(filter(lambda x: x[2] == cl.name, one\_to\_many))

        # Если есть хоть один ученик

        if len(clpupils) > 0:

            res\_12\_unsorted.append((cl.name, max(clpupils, key = lambda x: x[1])[1]))

    # Сортировка по среднему баллу

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

    return res\_12

def G3(many\_to\_many):

    res\_13 = []

    # Выделяем по одному человеку в список

    for fio, avr, clas in many\_to\_many:

        res\_13.append((fio, clas))

    # Сортируем список по первому ключу - названию класса

    res\_13 = sorted(res\_13, key = itemgetter(1))

    return res\_13

def main():

    one\_to\_many = [(pup.fio, pup.avr, cl.name)

        for cl in clases

        for pup in pupils

        if pup.clas\_id == cl.id]

    many\_to\_many\_temp = [(cl.name, pupcl.clas\_id, pupcl.pupil\_id)

        for cl in clases

        for pupcl in pupclases

        if cl.id == pupcl.clas\_id]

    many\_to\_many = [(pup.fio, pup.avr, clas)

        for clas, clas\_id, pupil\_id in many\_to\_many\_temp

        for pup in pupils if pup.id == pupil\_id]

    print('Задание Г1', G1(one\_to\_many),

          'Задание Г2', G2(one\_to\_many),

          'Задание Г3', G3(many\_to\_many),

          sep = '\n')

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

    main()

**Файл TDD.py:**

import unittest

from rk import \*

one\_to\_many = [(pup.fio, pup.avr, cl.name)

    for cl in clases

    for pup in pupils

    if pup.clas\_id == cl.id]

many\_to\_many\_temp = [(cl.name, pupcl.clas\_id, pupcl.pupil\_id)

    for cl in clases

    for pupcl in pupclases

    if cl.id == pupcl.clas\_id]

many\_to\_many = [(pup.fio, pup.avr, clas)

    for clas, clas\_id, pupil\_id in many\_to\_many\_temp

    for pup in pupils if pup.id == pupil\_id]

class TestTDD(unittest.TestCase):

    def test1(self):

        self.assertEqual(G1(one\_to\_many), [('А1', ['Сиденко', 'Варина'])])

    def test2(self):

        self.assertEqual(G2(one\_to\_many), [('Б2', 5.0), ('В3', 4.69), ('А1', 4.5)])

    def test3(self):

        self.assertEqual(G3(many\_to\_many), [('Варина', '1 класс (продленка)'),

        ('Сиденко', '2 класс (продленка)'), ('Ананьев', '2 класс (продленка)'),

        ('Верешко', '3 класс (продленка)'), ('Волков', '3 класс (продленка)'), ('Сиденко', 'А1'),

        ('Варина', 'А1'), ('Ананьев', 'Б2'), ('Волков', 'В3'), ('Верешко', 'В3')])

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

    unittest.main()

# Результат выполнения

# 