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
Вариант 11

Выполнил:
студент группы ИУ5-34Б:
Мамоу Асман
Подпись и дата:

проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е
Подпись и дата:

Москва, 2022 г.

Условия РК №1

Вариант Г. Предметная область №11

1. «Компьютер» и «Программа» связаны соотношением один-ко-многим. Выведите список всех компьютеров, у которых название начинается с буквы «А», и список программ в ней.

2. «Компьютер» и «Программа» связаны соотношением один-ко-многим. Выведите список компьютеров с максимальным количеством программ в каждом компьютере, отсортированный по максимальному количеству.

3. «Компьютер» и «Программа» связаны соотношением многие-ко-многим. Выведите список всех связанных компьютеров и программ, отсортированный по компьютерам, сортировка программам произвольная.

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).

Код программы к РК №1

```
from operator import itemgetter
class Programm:
    #Программа
    def __init__(self, id, name, count, pc_id):
        self.id = id
        self.name = name
        self.count = count
        self.pc_id = pc_id

class PC:
    #Компьютер
    def __init__(self, id, title):
        self.id = id
        self.title = title

class ProgrammPc: #связь многие-ко-многим
    def __init__(self, programm_id, pc_id):
        self.pc_id = pc_id
        self.programm_id = programm_id


#Компьютеры
programms = [
    Programm(1, 'Programm №1', 20000, 1),
    Programm(2, 'Programm №2', 3000, 2),
    Programm(3, 'Programm №3', 15000, 2),
    Programm(4, 'Programm №4', 86400, 3),
    Programm(5, 'Programm №5', 30000, 4)
]
#Программы
PCs = [
    PC(1, 'APC'),
    PC(2, 'BPC'),
    PC(3, 'CPC'),
```

```
    PC(4, 'DPC')
]
Programm_PC = [
    ProgrammPc(1, 1),
    ProgrammPc(2, 2),
    ProgrammPc(3, 3),
    ProgrammPc(4, 4),
    ProgrammPc(5, 1),
    ProgrammPc(5, 1),
    ProgrammPc(2, 3),
    ProgrammPc(3, 2),
    ProgrammPc(4, 1),
    ProgrammPc(5, 4)
]
# Соединение данных один-ко-многим
one_to_many = [(P.name, P.count, C.title)
               for C in PCs
               for P in programms
               if C.id == P.pc_id]

# Соединение данных многие-ко-многим
many_to_many_temp = [(C.title, X.pc_id, X.programm_id)
                     for C in PCs
                     for X in Programm_PC
                     if C.id == X.pc_id]

many_to_many = [(P.name, P.count, pc_title)
                for pc_title, pc_id, programm_id in many_to_many_temp
                for P in programms if P.id == programm_id]
def task_1(one_to_many):
    task_1 = {}
    for C in PCs:
        if C.title[0] == 'A':
            pc_p = list(filter(lambda i: i[2] == C.title, one_to_many))
            pc_p_names = [x for x, _, _ in pc_p]
            task_1[C.title] = [pc_p_names]
    return task_1

def task_2 (one_to_many):
    task_2_unsorted = []
    for C in PCs:
        pc_p = list(filter(lambda i: i[2] == C.title, one_to_many))
        if len(pc_p) > 0:
            p_count = [count for _, count, _ in pc_p]
            p_count_sum = max(p_count)
            task_2_unsorted.append((C.title, p_count_sum))
    task_2 = sorted(task_2_unsorted, key=itemgetter(1), reverse=True)
    return task_2

def task_3 (many_to_many):
    task_3 = sorted(many_to_many, key=itemgetter(2))
    return task_3

def main():
    print('\nЗадание Г1')
    print(task_1(one_to_many))
    print('\nЗадание Г2')
    print(task_2(one_to_many))
    print('\nЗадание Г3')
    print(task_3(many_to_many))

if __name__ == '__main__':
    main()
```

Код программы к РК №2 testing.py

```
import os
import sys
import unittest

sys.path.append(os.getcwd())  # current working directory
from RK1 import task_1, task_2, task_3
from RK1 import one_to_many, many_to_many


class test_task_1(unittest.TestCase):
    def testtask_1(self):
        self.assertEqual(task_1(one_to_many), {'APC': [['Programm №1']]})


class test_task_2(unittest.TestCase):
    def testtask_2(self):
        self.assertEqual(task_2(one_to_many), [('CPC', 86400), ('DPC', 30000),
('APC', 20000), ('BPC', 15000)])


class test_task_3(unittest.TestCase):
    def testtask_3(self):
        self.assertEqual(task_3(many_to_many), [('Programm №1', 20000, 'APC'),
                                                ('Programm №5', 30000, 'APC'),
                                                ('Programm №5', 30000, 'APC'),
                                                ('Programm №4', 86400, 'APC'),
                                                ('Programm №2', 3000, 'BPC'),
                                                ('Programm №3', 15000, 'BPC'),
                                                ('Programm №3', 15000, 'CPC'),
                                                ('Programm №2', 3000, 'CPC'),
                                                ('Programm №4', 86400, 'DPC'),
                                                ('Programm №5', 30000, 'DPC')])


if __name__ == '__main__':
    unittest.main()
```

Результат

```
Testing started at 15:07 ...
Launching pytest with arguments C:/Users/asman/PycharmProjects/RK1/testing.py --no-header --no-summary -q in C:\Users\asman\PycharmProjects\RK1

============================ test session starts ============================
collecting ... collected 3 items

testing.py::test_task_1::testtask_1 PASSED                            [ 33%]
testing.py::test_task_2::testtask_2 PASSED                            [ 66%]
testing.py::test_task_3::testtask_3 PASSED                            [100%]

============================ 3 passed in 0.02s ============================

Process finished with exit code 0
```