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
«Основные конструкции языка Python»

Выполнил:
студент группы ИУ5-32Б:
Балабанов Алексей Олегович
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапонюк Ю.Е.
Подпись и дата:

Москва, 2021 г.

Постановка задачи:

Вариант №2

Вариант Б.

1. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.
2. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список отделов с количеством сотрудников в каждом отделе, отсортированный по количеству сотрудников.
3. «Класс» и «Школьник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
4. **Условия рубежного контроля №2 по курсу БКИТ**
5. Рубежный контроль представляет собой разработку тестов на языке Python.
6. 1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
7. 2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Текст кода:

Main.py

```python
# используется для сортировки
from operator import itemgetter


class Student:
    """Школьник"""

    def __init__(self, id, fio, bal, st_id):
        self.id = id
        self.fio = fio
        self.bal = bal      #Оценка по информатике
        self.st_id = st_id


class Class:
    """Класс"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class StCl:
    """
    'Учащиеся класса' для реализации
    связи многие-ко-многим
```

```
    """

    def __init__(self, st_id, class_id):
        self.class_id = class_id
        self.st_id = st_id


# Классы
clas = [
    Class(1, 'физико-математичский'),
    Class(2, 'химико-биологический'),
    Class(3, 'социально-экономический'),
    Class(4, 'информационно-технологический'),

]

# Школьники
students = [
    Student(1, 'Артамонов', 2, 1),
    Student(2, 'Петров', 3, 2),
    Student(3, 'Иваненко', 4, 3),
    Student(4, 'Иванов', 3, 3),
    Student(5, 'Иванин', 2, 3),
    Student(6, 'Балабанов', 5, 4),
]

st_cl = [
    StCl(1, 1),
    StCl(2, 2),
    StCl(3, 3),
    StCl(4, 3),
    StCl(5, 3),
    StCl(6, 4),

    StCl(3, 2),
    StCl(4, 1),
    StCl(5, 4),
    StCl(6, 3),
]

def B1(one_to_many):
    res_11 = sorted(one_to_many, key=itemgetter(0))
    return res_11

def B2(one_to_many):
    res_12_unsorted = []
    # Перебираем все классы
    for c in clas:
        # Список учеников класса
        c_student = list(filter(lambda i: i[2] == c.name, one_to_many))
        # Если класс не пустой
        if len(c_student) > 0:
            res_12_unsorted.append((c.name, len(c_student)))

    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12
def B3(many_to_many):
    res_13 = {}
    # Перебираем все отделы
    for s in students:
        if s.fio.endswith("ов"):
            c_students = list(filter(lambda i: i[0] == s.fio, many_to_many))
            c_students_names = [x[2] for x in c_students]
            # Добавляем результат в словарь
```

```
            # ключ - отдел, значение - список фамилий
            res_13[s.fio] = c_students_names

    return res_13


def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(s.fio, s.bal, c.name)
                   for c in clas
                   for s in students
                   if s.st_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, cs.class_id, cs.st_id)
                         for c in clas
                         for cs in st_cl
                         if c.id == cs.class_id]

    many_to_many = [(s.fio, s.bal, class_name)
                    for class_name, class_id, st_id in many_to_many_temp
                    for s in students if s.id == st_id]

    print('Задание Б1')
    print(B1(one_to_many))

    print('\nЗадание Б2')
    print(B2(one_to_many))

    print('\nЗадание Б3')
    print(B3(many_to_many))

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

Test_TDD.py

```
import unittest
import sys, os
sys.path.append(os.getcwd())
from main import *


class Test_B(unittest.TestCase):
    def test_B1(self):
        one_to_many = [(s.fio, s.bal, c.name)
                       for c in clas
                       for s in students
                       if s.st_id == c.id]
        self.assertEqual(B1(one_to_many), [('Артамонов', 2, 'физико-
математичский'),
                                           ('Балабанов', 5, 'информационно-
технологический'),
                                           ('Иваненко', 4, 'социально-
экономический'),
                                           ('Иванин', 2, 'социально-
экономический'),
                                           ('Иванов', 3, 'социально-
экономический'),
                                           ('Петров', 3, 'химико-
биологический')])
```

```
    def test_B2(self):
        one_to_many = [(s.fio, s.bal, c.name)
                       for c in clas
                       for s in students
                       if s.st_id == c.id]
        self.assertEqual(B2(one_to_many),  [('социально-экономический', 3),
                                            ('физико-математичский', 1),
                                            ('химико-биологический', 1),
                                            ('информационно-технологический',
1)])

    def test_B3(self):
        many_to_many_temp = [(c.name, cs.class_id, cs.st_id)
                             for c in clas
                             for cs in st_cl
                             if c.id == cs.class_id]

        many_to_many = [(s.fio, s.bal, class_name)
                        for class_name, class_id, st_id in many_to_many_temp
                        for s in students if s.id == st_id]
        self.assertEqual(B3(many_to_many), {'Артамонов': ['физико-
математичский'],
                                            'Петров': ['химико-
биологический'],
                                            'Иванов': ['физико-
математичский',
                                                       'социально-
экономический'],
                                            'Балабанов': ['социально-
экономический',
                                                          'информационно-
технологический']})

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

Тестирование:
Main.py

```
C:\Users\Acer\RK_1\venv\Scripts\python.exe C:/Users/Acer/RK_1/main.py
Задание Б1
('Артамонов', 2, 'физико-математичский')
('Балабанов', 5, 'информационно-технологический')
('Иваненко', 4, 'социально-экономический')
('Иванин', 2, 'социально-экономический')
('Иванов', 3, 'социально-экономический')
('Петров', 3, 'химико-биологический')

Задание Б2
[('социально-экономический', 3), ('физико-математичский', 1), ('химико-биологический', 1), ('информационно-технологический', 1)]

Задание Б3
Артамонов ['физико-математичский']
Петров ['химико-биологический']
Иванов ['физико-математичский', 'социально-экономический']
Балабанов ['социально-экономический', 'информационно-технологический']

Process finished with exit code 0
```

(Для более удобного отображения результата вывод Б1 и Б2 был изменён)
Test_TDD.py

```
C:\Users\Acer\RK_2\venv\Scripts\python.exe "C:\Program Files\JetBrains\PyCharm 2021.2.1\plugins\python\helpers\pycharm\_jb_unittest_runner.py" --path C:/Users/Acer/RK_2/Test_TI
Testing started at 15:50 ...


Ran 3 tests in 0.002s

OK
Launching unittests with arguments python -m unittest C:/Users/Acer/RK_2/Test_TDD.py in C:\Users\Acer\RK_2
```