Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

ДИСЦИПЛИНА: «БКИТ»

Рубежная контроль № 2

Студент Герасимов Андрей ИУ5-35Б _____

(И.О. Фамилия) (Группа) (Подпись, дата)

Преподаватель Гапанюк Ю.Е. _____

(И.О. Фамилия) (Подпись, дата)

Москва, 2021г.

Полученное 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).

Задание РК1:

1. **Вариант Д.**
   1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (*отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений*).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Вариант:

2	Школьник	Класс

Текст программы:

1. Файл students.py

```
class Student:
    """Ученик"""

    def __init__(self, num, fio, mark, ClassDep_id):
        self.id = num
        self.fio = fio
        self.mark = mark
        self.ClassDep_id = ClassDep_id
```

2. Файл Departament.py

```
class ClassDep:
    """Класс"""

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

3. Файл StudentsInClass.py

```
class StudentInClass:
    """ 'Ученики класса' для реализации связи многие-ко-многим"""

    def __init__(self, ClassDep_id, student_id):
        self.ClassDep_id = ClassDep_id
        self.student_id = student_id
```

4. Файл DataOfClasses

```
from students import Student
from Departament import ClassDep
from StudentsInClass import StudentInClass

# Классы

ClassDeps = [
    ClassDep(1, '7А'),
    ClassDep(2, '8Б'),
    ClassDep(3, '9В'),
    ClassDep(4, '10А'),
    ClassDep(5, '11Б'),
]

# Ученики
Students = [
    Student(1, 'Герасимов', 78, 1),
    Student(2, 'Ищенко', 97, 2),
    Student(3, 'Акулова', 45, 3),
    Student(4, 'Троцук', 0, 4),
    Student(5, 'Иванов', 29, 5),
    Student(6, 'Макаров', 83, 1),
    Student(7, 'Сидоров', 65, 2),
    Student(8, 'Сыса', 66, 3),
    Student(9, 'Морозов', 48, 4),
    Student(10, 'Артёменко', 77, 5),
]
# Распределение по классам
StudentInClasses = [
    StudentInClass(1, 1),
    StudentInClass(2, 2),
    StudentInClass(3, 3),
    StudentInClass(4, 4),
    StudentInClass(5, 5),
    StudentInClass(1, 6),
    StudentInClass(2, 7),
    StudentInClass(3, 8),
    StudentInClass(4, 9),
    StudentInClass(5, 10),
]
```

5. Файл Main.py

```
import TDD
from operator import itemgetter
from DataOfClasses import ClassDeps, StudentInClasses,Students


def taskD1():
    many_to_many_temp = [(c.name, pc.ClassDep_id, pc.student_id)
                         for c in ClassDeps
                         for pc in StudentInClasses
                         if c.id == pc.ClassDep_id]

    many_to_many = [(p.fio, p.mark, ClassDep)
                    for ClassDep, ClassDep_id, student_id in many_to_many_temp
                    for p in Students if p.id == student_id]
    # «Класс» и «Ученик» связаны соотношением один-ко-многим. Выведите список
всех учеников, у которых фамилия заканчивается на «ов», и названия их классов.

    answer_1 = []
    b = [j for j in many_to_many if j[0][-1:] == 'в' and j[0][-2] == 'о']
    answer_1 = {j[2]: [i[0] for i in b if i[2] == j[2]] for j in b}
    return answer_1


def taskD2():
```

```
    one_to_many = [(p.fio, p.mark, c.name)
                   for c in ClassDeps
                   for p in Students
                   if p.ClassDep_id == c.id]
    # «Класс» и «Ученик» связаны соотношением один-ко-многим. Выведите список
классов со средней оценкой учеников в каждом отделе, отсортированный по средней
оценке

    answer_2NS = []
    # Перебираем все классы

    for c in ClassDeps:
        # Список учеников класса
        list_students = list(filter(lambda i: i[2] == c.name, one_to_many))
        # Если класс не пустой
        if len(list_students) > 0:
            # Оценки учеников класса
            mark = [mark for _, mark, _ in list_students]
            # Среднее значение оценок учеников класса
            mark_sum = (round((sum(mark)) / (len(list_students)), 3))
            answer_2NS.append((c.name, mark_sum))

    # Сортировка по среднему значению оценки
    answer_2 = sorted(answer_2NS, key=itemgetter(1), reverse=True)

    return answer_2

    # «Класс» и «Ученик» связаны соотношением многие-ко-многим.
    # Выведите список всех классов, у которых название начинается с буквы
«А»(или присутсвует),
    # и список классов в них учеников.


def taskD3(ClassDeps):
    many_to_many_temp = [(c.name, pc.ClassDep_id, pc.student_id)
                         for c in ClassDeps
                         for pc in StudentInClasses
                         if c.id == pc.ClassDep_id]

    many_to_many = [(p.fio, p.mark, ClassDep)
                    for ClassDep, ClassDep_id, student_id in many_to_many_temp
                    for p in Students if p.id == student_id]

    answer_3 = {}
    # Перебираем все классы
    for c in ClassDeps:
        if 'A' in c.name:
            # Список учеников класса
            list_students = list(filter(lambda i: i[2] == c.name, many_to_many))
            list_students_names = [x for x, _, _ in list_students]
            answer_3[c.name] = list_students_names

    return answer_3



if __name__ == '__main__':
    # Соединение данных один-ко-многим
    one_to_many = [(p.fio, p.mark, c.name)
                   for c in ClassDeps
                   for p in Students
                   if p.ClassDep_id == c.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(c.name, pc.ClassDep_id, pc.student_id)
                         for c in ClassDeps
                         for pc in StudentInClasses
                         if c.id == pc.ClassDep_id]
```

```
    many_to_many = [(p.fio, p.mark, ClassDep)
                    for ClassDep, ClassDep_id, student_id in many_to_many_temp
                    for p in Students if p.id == student_id]

    print('Задание D1:')
    print(taskD1())
    print('Задание D2:')
    print(taskD2())
    print('Задание D3:')
    print(taskD3(ClassDeps))
```

6. Файл TDD.py

```
import unittest
from main import taskD1, taskD2, taskD3, ClassDeps


class TestD(unittest.TestCase):

    def test_d1(self):
        self.assertEqual(taskD1(), {'7А': ['Герасимов', 'Макаров'], '8Б':
['Сидоров'], '10А': ['Морозов'], '11Б': ['Иванов']})

    def test_d2(self):
        self.assertEqual(taskD2(), [('8Б', 81.0), ('7А', 80.5), ('9В', 55.5),
('11Б', 53.0), ('10А', 24.0)])

    def test_d3(self):
        self.assertEqual(taskD3(ClassDeps), {'7А': ['Герасимов', 'Макаров'],
'10А': ['Троцук', 'Морозов']})


suite = unittest.TestLoader().loadTestsFromTestCase(TestD)
unittest.TextTestRunner(verbosity=2).run(suite)
```

Результат выполнения программы:

```
D:\Gerandden\RK2BKID\Scripts\python.exe D:/Gerandden/RK2BKID/main.py
test_d1 (TDD.TestD) ... ok
test_d2 (TDD.TestD) ... ok
test_d3 (TDD.TestD) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
Задание D1:
{'7А': ['Герасимов', 'Макаров'], '8Б': ['Сидоров'], '10А': ['Морозов'], '11Б': ['Иванов']}
Задание D2:
[('8Б', 81.0), ('7А', 80.5), ('9В', 55.5), ('11Б', 53.0), ('10А', 24.0)]
Задание D3:
{'7А': ['Герасимов', 'Макаров'], '10А': ['Троцук', 'Морозов']}

Process finished with exit code 0
```