

Рубежный контроль по БКИТ №2

Вариант запросов: Б

Вариант предметной области: 3

Запросы:

1. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список всех связанных школьников и классов, отсортированный по школьникам, сортировка по классам произвольная.
2. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список классов с количеством школьников в каждом классе, отсортированный по количеству школьников.
3. «Класс» и «Школьник» связаны соотношением многие-ко-многим. Выведите список всех школьников, у которых фамилия заканчивается на «г», и названия их классов.

Задания:

- 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

school.py

```
from operator import itemgetter
from collections import Counter
from prettytable import PrettyTable
```

```
class Schoolboy:
    """Школьник"""
    def init(self, id, fio, old, class_id):
        self.id = id
        self.fio = fio
        self.old = old
        self.class_id = class_id
```

```
class SchoolClass:
```

```

    """Класс"""
    def init(self, id, name):
        self.id = id
        self.name = name

class SchoolboyClass:
    """
    'Школьники' для реализации
    связи многие-ко-многим
    """
    def init(self, schoolboy_id, class_id):
        self.schoolboy_id = schoolboy_id
        self.class_id = class_id

# Классы
list_class = [
    SchoolClass(1, "A"),
    SchoolClass(2, "B"),
    SchoolClass(3, "C"),
    SchoolClass(4, "D"),
    SchoolClass(5, "E")
]

# Школьники
schoolboys = [
    Schoolboy(1, "Ivan", 7, 3),
    Schoolboy(2, "Vadim", 1, 1),
    Schoolboy(3, "Andrey", 10, 3),
    Schoolboy(4, "Boris", 12, 2),
    Schoolboy(5, "Egor", 8, 2),
    Schoolboy(6, "Petr", 7, 1),
    Schoolboy(7, "Timur", 17, 5),
    Schoolboy(8, "Fedor", 12, 3),
    Schoolboy(9, "Dmitry", 11, 3),
    Schoolboy(10, "Oleg", 15, 3)
]

schoolboy_class = [
    SchoolboyClass(5, 3),
    SchoolboyClass(5, 4),
    SchoolboyClass(8, 5),
    SchoolboyClass(7, 3),
    SchoolboyClass(2, 1),
    SchoolboyClass(2, 2),
    SchoolboyClass(3, 4),
    SchoolboyClass(7, 1),
    SchoolboyClass(4, 5),
    SchoolboyClass(5, 1)
]

```

```

def task1(schoolboys, list_class):
    return sorted(map(lambda x: x[:-1], [(scb.fio, scb.old, cls.name, cls.id)
                                         for scb in schoolboys
                                         for cls in list_class
                                         if scb.class_id == cls.id]),
                  key=itemgetter(0))

def task2(schoolboys, list_class):
    schoolboy_in_class = Counter((class_id, class_name)
                                for _, _, class_name, class_id
                                in [(scb.fio, scb.old, cls.name, cls.id)
                                    for scb in schoolboys
                                    for cls in list_class
                                    if scb.class_id == cls.id])
    return sorted([(cls[1], cou_scb)
                   for cls, cou_scb in schoolboy_in_class.items()],
                  key=itemgetter(1))

def task3(schoolboys, schoolboy_class, list_class):
    res = {}
    for scb in schoolboys:
        if scb.fio[-1] == "r":
            list_class_for_scb = [cls.name for mm in schoolboy_class
                                  for cls in list_class
                                  if (cls.id == mm.class_id and
                                      scb.id == mm.schoolboy_id)]
            if len(list_class_for_scb) > 0:
                res[scb.fio] = list_class_for_scb
    return [(scb, ", ".join(lst)) for scb, lst in res.items()]

def draw(name_of_fields, data):
    table = PrettyTable(name_of_fields)
    table.align = "l"
    table.add_rows(data)
    print(table)

def main():
    print("Задание Б1")
    res1 = task1(schoolboys, list_class)
    draw(("ФИО", "Возраст", "Название класса"), res1)

    print("\nЗадание Б2")
    res2 = task2(schoolboys, list_class)
    draw(("Название класса", "Количество учеников"), res2)

```

```
print("\nЗадание Б3")
res3 = task3(schoolboys, schoolboy_class, list_class)
draw(("ФИО", "Названия классов"), res3)
```

```
if name == "main":
    main()
```

tests.py

```
import unittest
```

```
from school import task1, task2, task3, Schoolboy, SchoolClass, SchoolboyClass
```

```
class TestSchool(unittest.TestCase):
```

```
    def setUp(self):
```

```
        self.list_class = [
```

```
            SchoolClass(1, "A"),
```

```
            SchoolClass(2, "B"),
```

```
            SchoolClass(3, "C"),
```

```
            SchoolClass(4, "D"),
```

```
            SchoolClass(5, "E")
```

```
        ]
```

```
        self.schoolboys = [
```

```
            Schoolboy(1, "Ivan", 7, 3),
```

```
            Schoolboy(2, "Vadim", 1, 1),
```

```
            Schoolboy(3, "Andrey", 10, 3),
```

```
            Schoolboy(4, "Boris", 12, 2),
```

```
            Schoolboy(5, "Egor", 8, 2),
```

```
            Schoolboy(6, "Petr", 7, 1),
```

```
            Schoolboy(7, "Timur", 17, 5),
```

```
            Schoolboy(8, "Fedor", 12, 3),
```

```
            Schoolboy(9, "Dmitry", 11, 3),
```

```
Schoolboy(10, "Oleg", 15, 3)
]
```

```
self.schoolboy_class = [
    SchoolboyClass(5, 3),
    SchoolboyClass(5, 4),
    SchoolboyClass(8, 5),
    SchoolboyClass(7, 3),
    SchoolboyClass(2, 1),
    SchoolboyClass(2, 2),
    SchoolboyClass(3, 4),
    SchoolboyClass(7, 1),
    SchoolboyClass(4, 5),
    SchoolboyClass(5, 1)
]
```

```
def test_task1(self):
    res = task1(self.schoolboys, self.list_class)
    self.assertEqual(res, [
        ('Andrey', 10, 'C'),
        ('Boris', 12, 'B'),
        ('Dmitry', 11, 'C'),
        ('Egor', 8, 'B'),
        ('Fedor', 12, 'C'),
        ('Ivan', 7, 'C'),
        ('Oleg', 15, 'C'),
        ('Petr', 7, 'A'),
        ('Timur', 17, 'E'),
        ('Vadim', 1, 'A')])
```

```
def test_task2(self):

    res = task2(self.schoolboys, self.list_class)

    self.assertListEqual(res, [('E', 1), ('A', 2), ('B', 2), ('C', 5)])


def test_task3(self):

    res = task3(self.schoolboys, self.schoolboy_class, self.list_class)

    self.assertListEqual(res, [

        ('Egor', 'C, D, A'),

        ('Timur', 'C, A'),

        ('Fedor', 'E')])


if name == "main":

    unittest.main()
```

Результат выполнения программы

Задание Б1

ФИО	Возраст	Название класса
Andrey	10	C
Boris	12	B
Dmitry	11	C
Egor	8	B
Fedor	12	C
Ivan	7	C
Oleg	15	C
Petr	7	A
Timur	17	E
Vadim	1	A

Задание Б2

Название класса	Количество учеников
E	1
A	2
B	2
C	5

Задание Б3

ФИО	Названия классов
Egor	C, D, A
Timur	C, A
Fedor	E

```
• (.venv) artem@artem-MS-7B53:~/Study/Course_BKIT_3sem/rk2$ python3 tests.py
...
-----
Ran 3 tests in 0.001s

OK
○ (.venv) artem@artem-MS-7B53:~/Study/Course_BKIT_3sem/rk2$ █
```