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 рубежному контролю №2

Выполнил:
студент группы ИУ5-31Б
Ашуров Г. В.

Проверил:
Гапанюк Е.Ю.

2022

Условия рубежного контроля №2 по курсу БКИТ

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).

Измененный код РК1:

```
from operator import itemgetter


class Pupil:
    def __init__(self, Id, Name, Rating, IdGr):
        self.Id = Id
        self.Name = Name
        self.Rating = Rating
        self.IdGr = IdGr


class Grade:
    def __init__(self, Id, Name):
        self.Id = Id
        self.Name = Name


class PupilAndGrade:
    def __init__(self, IdGr, IdP):
        self.IdP = IdP
        self.IdGr = IdGr


pupils = [
    Pupil(1, "Artemov", 4.6, 1),
    Pupil(2, "Berezov", 4, 1),
    Pupil(3, "Rugalina", 3.2, 2),
    Pupil(4, "Kirov", 4.8, 2),
    Pupil(5, "Belova", 4.5, 3),
    Pupil(6, "Lesov", 4.3, 3),
    Pupil(7, "Alekseev", 5, 4),
    Pupil(8, "Petrov", 4.6, 4),
]

grades = [
    Grade(1, "10A"),
    Grade(2, "11A"),
    Grade(3, "11B"),
    Grade(4, "11C")
```

```
]

pupilofgrade = [
    PupilAndGrade(1, 1),
    PupilAndGrade(1, 2),
    PupilAndGrade(2, 3),
    PupilAndGrade(2, 4),
    PupilAndGrade(3, 5),
    PupilAndGrade(3, 6),
    PupilAndGrade(4, 7),
    PupilAndGrade(4, 8),

]


def one_to_many(grades, pupils):
    return [(p.Name, p.Rating, g.Name)
            for g in pupils
            for p in grades
            if p.IdGr == g.Id]


def many_to_many(grades, pupils):
    many_to_many_temp = [(g.Name, pg.IdGr, pg.IdP)
                         for g in grades
                         for pg in pupilofgrade
                         if g.Id == pg.IdGr]
    return [(p.Id, IdGr)
            for name, IdGr, IdP in many_to_many_temp
            for p in pupils if p.Id == IdP]


def many_to_many(grades, pupils):
    many_to_many_temp = [(g.Name, pg.IdGr, pg.IdP)
                         for g in grades
                         for pg in pupilofgrade
                         if g.Id == pg.IdGr]
    return [(p.Name, p.Rating, gradeName)
            for gradeName, gradeId, pupilId in many_to_many_temp
            for p in pupils
            if p.Id == pupilId]


def A1(grades, pupils) -> list:
    res1 = sorted(one_to_many(pupils, grades), key=itemgetter(2))
    return list(res1)


def A2(grades, pupils) -> list:
    res2_unsorted = []
    for p in grades:
```

```
        p_grades = list(
            filter(lambda i: i[2] == p.Name, one_to_many(pupils, grades)))
        if len(p_grades) > 0:
            p_sizes = [sal for _, sal, _ in p_grades]
            p_sizes_sum = sum(p_sizes)/2
            res2_unsorted.append((p.Name, p_sizes_sum))
    return sorted(res2_unsorted, key=itemgetter(1), reverse=True)

def A3(grades, pupils):
    res3 = {}
    for g in grades:
        if '11' in g.Name:
            p_grades = list(
                filter(lambda i: i[2] == g.Name, many_to_many(grades, pupils)))
            p_grades_names = [x for x, _, _ in p_grades]
            res3[g.Name] = p_grades_names
    return res3


if __name__ == '__main__':
    print('Задание A1')
    print(A1(grades, pupils))
    print('Задание A2')
    print(A2(grades, pupils))
    print('Задание A3')
    print(A3(grades, pupils))
```

Тестирование:

```
import unittest
from rk1 import *


class rk_test(unittest.TestCase):

    def setUp(self):
        self.grades = [
            Grade(1, "10A"),
            Grade(2, "11A"),
            Grade(3, "11B"),
            Grade(4, "11C")
        ]
        self.pupils = [
            Pupil(1, "Artemov", 4.6, 1),
            Pupil(2, "Berezov", 4, 1),
            Pupil(3, "Rugalina", 3.2, 2),
            Pupil(4, "Kirov", 4.8, 2),
            Pupil(5, "Belova", 4.5, 3),
            Pupil(6, "Lesov", 4.3, 3),
            Pupil(7, "Alekseev", 5, 4),
```

```
            Pupil(8, "Petrov", 4.6, 4)
        ]
        self.pupils_grades = [
            PupilAndGrade(1, 1),
            PupilAndGrade(1, 2),
            PupilAndGrade(2, 3),
            PupilAndGrade(2, 4),
            PupilAndGrade(3, 5),
            PupilAndGrade(3, 6),
            PupilAndGrade(4, 7),
            PupilAndGrade(4, 8)
        ]

    def test_A1(self):
        expected_result = [
            ('Artemov', 4.6, '10A'),
            ('Berezov', 4, '10A'),
            ('Rugalina', 3.2, '11A'),
            ('Kirov', 4.8, '11A'),
            ('Belova', 4.5, '11B'),
            ('Lesov', 4.3, '11B'),
            ('Alekseev', 5, '11C'),
            ('Petrov', 4.6, '11C')
        ]
        result = A1(self.grades, self.pupils)
        self.assertEqual(result, expected_result)

    def test_A2(self):
        expected_result = [
            ('11C', 4.8),
            ('11B', 4.4),
            ('10A', 4.3),
            ('11A', 4.0)
        ]
        result = A2(self.grades, self.pupils)
        self.assertEqual(result, expected_result)

    def test_A3(self):
        expected_result = {
            '11A': ['Rugalina', 'Kirov'],
            '11B': ['Belova', 'Lesov'],
            '11C': ['Alekseev', 'Petrov']
        }
        result = A3(self.grades, self.pupils)
        self.assertEqual(result, expected_result)


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

Результат тестирования:

```
Ran 3 tests in 0.001s

OK
PS C:\Users\Георгий\Desktop\r>
```