|  |  |
| --- | --- |
|  | **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 «ГУИМЦ»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Парадигмы и конструкции языков программирования»»

ОТЧЕТ

ПО РУБЕЖНОМУ КОНТРОЛЮ №2

Вариант предметной области 27

Вариант запросов: Б

Студент: Мефодьев И.Н., группа ИУ5Ц-52Б

Преподаватель: Гапанюк Ю. Е.

Москва, 2024 г.

**Задание**

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).

**Листинг программы**

*# файл prod.py*

*# Мефодьев ИУ5Ц-52Б вариант 27 задание Б*

*# классы Преподаватель, учебный курс*

*# код был изменён для проведения тестов.*

class Prepod:   *#Преподаватель*

    def \_\_init\_\_(self, id: int, FIO: str, salary: float):

        self.id = id

        self.FIO = FIO

        self.salary = salary

class Course:   *# учебный курс*

    def \_\_init\_\_(self, id: int, name: str, prepod\_id: int):

        self.id = id

        self.name = name

        self.prepod\_id = prepod\_id

class Prep\_course:  *# курсы у преподователя (для реализации связи многие-ко-многим)*

    def \_\_init\_\_(self, prep\_id: int, course\_id: int):

        self.prep\_id = prep\_id

        self.course\_id = course\_id

Prepods = [

    Prepod(1, "Большаков Сергей Алексеевич", 45000.00),

    Prepod(2, "Крылов Алексей Олегович", 55000.00),

    Prepod(3, "Яковишена Светлана Георгиевна", 40000.00),

    Prepod(4, "Маслеников Константин Юрьевич", 65000.00),

    Prepod(5, "Чепик Елена Чеславовна", 30000.00),

]

Courses = [

    Course(1, "Основы программирования", 1),

    Course(2, "Системное программирование", 1),

    Course(3, "История", 2),

    Course(4, "Английский язык", 3),

    Course(5, "Модели данных", 4),

    Course(6, "Базы данных", 4),

    Course(7, "Оперативный анализ данных", 4),

    Course(8, "Инженерная графика", 5)

]

Cour\_preps = [

    Prep\_course(1, 1),

    Prep\_course(1, 2),

    Prep\_course(2, 3),

    Prep\_course(3, 4),

    Prep\_course(4, 5),

    Prep\_course(4, 6),

    Prep\_course(4, 7),

    Prep\_course(5, 8),

]

def task1(prepods: list[Prepod], courses: list[Course]):

    Presult = [(p, c) *# формируем связь один ко многим*

        for p in prepods

        for c in courses

        if p.id == c.prepod\_id

    ]

*# сортируем по преподавателям*

    Presult.sort(key = lambda el: el[0].FIO)

    return Presult

def task2(prepods: list[Prepod], courses: list[Course]):

    result = []

    for p in prepods:

        tmp\_res = (p, [])

        for c in courses:

            if p.id == c.prepod\_id:

                tmp\_res[1].append(c)

        result.append(tmp\_res)

*#сортировка по количеству курсов у преподавателей*

    result.sort(key = lambda el: len(el[1]), reverse=True)

    return result

def task3(prepods: list[Prepod], courses: list[Course], prep\_cours: list[Prep\_course]):

    result = []

    curPrepID = 0

    index = -1

    for el in prep\_cours:

        if(prepods[el.prep\_id - 1].FIO.split()[0][-2:] == "ов" and curPrepID != el.prep\_id):

*# второе условие - чтоб избежать повторов*

            curPrepID = el.prep\_id

            index+=1

            result.append((Prepods[curPrepID-1], []))

        if(curPrepID == el.prep\_id):

            result[index][1].append(courses[el.course\_id - 1]) *# тут был баг из-за отсутствия -1*

    return result

res1 = task1(Prepods, Courses)

print("Запрос 1")

for (p, c) in res1:

    print(p.FIO, "\t - \t", c.name)

res2 = task2(Prepods, Courses)

print("Запрос 2")

for (p, c) in res2:

    print(p.FIO, "\t - \t", len(c))

res3 = task3(Prepods, Courses, Cour\_preps)

print("Запрос 3")

print(res3)

for el in res3:

    print(el[0].FIO, end=":\n")

    for elC in el[1]:

        print("\t",elC.name, end=";\n")

    print("\n")

*#файл tests.py*

from prod import \*

def test\_task1():

    PreResult1 = task1(Prepods, Courses)

    result\_test\_1 = True

    result\_expect\_1 = [

        (Prepods[0].FIO, Courses[0].name),

        (Prepods[0].FIO, Courses[1].name),

        (Prepods[1].FIO, Courses[2].name),

        (Prepods[3].FIO, Courses[4].name),

        (Prepods[3].FIO, Courses[5].name),

        (Prepods[3].FIO, Courses[6].name),

        (Prepods[4].FIO, Courses[7].name),

        (Prepods[2].FIO, Courses[3].name)

    ]

    print(result\_expect\_1)

    i = 0

    for (p, c) in PreResult1:

        print(p.FIO, "\t - \t", c.name)

        if(p.FIO != result\_expect\_1[i][0] or c.name != result\_expect\_1[i][1]):

            result\_test\_1 = False

            break

        i+=1

*# не можем сравнить в лоб из-за особенностей объектов в python (например просто распечатать объект нельзя)*

    assert result\_test\_1

def test\_task2():

    PreResult2 = task2(Prepods, Courses)

    result\_test\_2 = True

    result\_expect\_2 = [

        (Prepods[3].FIO, 3),

        (Prepods[0].FIO, 2),

        (Prepods[1].FIO, 1),

        (Prepods[2].FIO, 1),

        (Prepods[4].FIO, 1),

    ]

    i = 0

    for (p, c) in PreResult2:

        print(p.FIO, "\t - \t", len(c))

        if(p.FIO != result\_expect\_2[i][0] or len(c) != result\_expect\_2[i][1]):

            result\_test\_2 = False

            break

        i+=1

    assert result\_test\_2

def test\_task3():

    PreResult3 = task3(Prepods, Courses, Cour\_preps)

    result\_test\_3 = True

    result\_expect\_3 = [

        (Prepods[0], [Courses[0], Courses[1]]),

        (Prepods[1], [Courses[2]]),

        (Prepods[3], [Courses[4],Courses[5],Courses[6]])

    ]

    print(result\_expect\_3)

    i = 0

    for el in PreResult3:

        if(result\_expect\_3[i][0]==el[0] and result\_test\_3):

            j = 0

            for elC in el[1]:

                print(elC.name)

                if(elC!=PreResult3[i][1][j]):

                    result\_test\_3 = False *# Одновременно флаг для внешнего цикла*

                    break

                j+=1

        else:

            result\_test\_3 = False

            break

        i+=1

    assert result\_test\_3

**Результат работы программы**

