Защищено:
Нардид А.Н.

"__"_____2024 г.

Демонстрация:
Нардид А.Н.

"__"_____2024 г.

Отчет по рубежному контролю № 2 по курсу Парадигмы и конструкции языков программирования ГУИМЦ

Тема работы: "Рубежный контроль представляет собой разработку тестов на языке Python."

4
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-53Б _____
(подпись)

Пронин В.К.

"__"_____2024 г.

Москва, МГТУ - 2024 г.

1. Тема и задание для выполнения рубежного контроля.

Тема работы: 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).

2. Листинг программы

Main.py

```
from operator import itemgetter

class Faculty:
    """Факультет"""
    def __init__(self, id, name, salary, uni_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.uni_id = uni_id

class University:
    """Университет"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class FacultyUniversity:
    """Факультеты университетов для связи многие-ко-многим"""
    def __init__(self, faculty_id, university_id):
        self.faculty_id = faculty_id
        self.university_id = university_id

def task_b1(one_to_many):
    """Все факультеты, где название начинается с 'И', и их университеты"""
    return [(f_name, u_name) for f_name, _, u_name in one_to_many if f_name.startswith("И")]

def task_b2(one_to_many, universities):
    """Университеты с минимальной зарплатой на каждом факультете"""
    res = []
    for u in universities:
        u_faculties = list(filter(lambda x: x[2] == u.name, one_to_many))
        if u_faculties:
            min_salary = min(sal for _, sal, _ in u_faculties)
            res.append((u.name, min_salary))
    return sorted(res, key=itemgetter(1))

def task_b3(many_to_many):
    """Список всех факультетов и университетов, отсортированный по факультетам"""
    return sorted(many_to_many, key=itemgetter(0))

def create_one_to_many(faculties, universities):
    return [(f.name, f.salary, u.name)
```

```
        for u in universities
        for f in faculties
        if f.uni_id == u.id]

def create_many_to_many(faculties, universities, fac_uni):
    many_to_many_temp = [(u.name, fu.university_id, fu.faculty_id)
                         for u in universities
                         for fu in fac_uni
                         if u.id == fu.university_id]
    return [(f.name, f.salary, uni_name)
            for uni_name, _, fac_id in many_to_many_temp
            for f in faculties if f.id == fac_id]
```

Unit_tests.py

```
import unittest
from main import create_one_to_many, create_many_to_many, task_b1, task_b2, task_b3, Faculty, University,
FacultyUniversity

class TestMainMethods(unittest.TestCase):
    def setUp(self):
        self.universities = [
            University(1, "МГТУ"),
            University(2, "МГУ"),
            University(3, "НИУ ВШЭ")
        ]

        self.faculties = [
            Faculty(1, "ИУ5", 50000, 1),
            Faculty(2, "МТ4", 45000, 1),
            Faculty(3, "ГУИМЦ", 30000, 2),
            Faculty(4, "ИУ8", 60000, 3),
            Faculty(5, "РК9", 35000, 3)
        ]

        self.fac_uni = [
            FacultyUniversity(1, 1),
            FacultyUniversity(2, 1),
            FacultyUniversity(3, 2),
            FacultyUniversity(4, 3),
            FacultyUniversity(5, 3)
        ]

        self.one_to_many = create_one_to_many(self.faculties, self.universities)
        self.many_to_many = create_many_to_many(self.faculties, self.universities, self.fac_uni)

    def test_task_b1(self):
        result = task_b1(self.one_to_many)
        reference = [('ИУ5', 'МГТУ'), ('ИУ8', 'НИУ ВШЭ')]
        self.assertEqual(result, reference)

    def test_task_b2(self):
        result = task_b2(self.one_to_many, self.universities)
        reference = [('МГУ', 30000), ('НИУ ВШЭ', 35000), ('МГТУ', 45000)]
        self.assertEqual(result, reference)

    def test_task_b3(self):
        result = task_b3(self.many_to_many)
```

```
        reference = [
            ('ГУИМЦ', 30000, 'МГУ'),
            ('ИУ5', 50000, 'МГТУ'),
            ('ИУ8', 60000, 'НИУ ВШЭ'),
            ('МТ4', 45000, 'МГТУ'),
            ('РК9', 35000, 'НИУ ВШЭ')
        ]
        self.assertEqual(result, reference)

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