|  |  |  |
| --- | --- | --- |
| 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.

# 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).

# 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()