Защищено:
Гапанюк Ю.Е.

"__"_____2024 г.

Демонстрация:
Гапанюк Ю.Е.

"__"_____2024 г.

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

Вариант запросов: В
Вариант предметной области: 20

6
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-53Б _____
(подпись)

Ахмеров Д.И.

"5" декабря 2024 г.

Москва, МГТУ – 2024

Вариант запросов: В. Вариант предметной области: 20

1. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список всех деталей, у которых название начинается с буквы «Т», и названия их поставщиков.
2. «Поставщик» и «Деталь» связаны соотношением один-ко-многим. Выведите список поставщиков с минимальной ценой их деталей, отсортированный по минимальной цене.
3. «Поставщик» и «Деталь» связаны соотношением многие-ко-многим. Выведите список всех связанных деталей и поставщиков, отсортированный по деталям, сортировка по поставщикам произвольная.

Условия рубежного контроля №2 по курсу ПиКЯП:

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).

Текст программы

main.py

```
from operator import itemgetter


class Detail:
    def __init__(self, id, name, price, id_supplier):
        self.id = id
        self.name = name
        self.price = price
        self.id_supplier = id_supplier


class Supplier:
    def __init__(self, id, name):
        self.id = id
        self.name = name
```

```
class DetailSupplier:
    def __init__(self, id_detail, id_supplier):
        self.id_detail = id_detail
        self.id_supplier = id_supplier

def one_to_many(details, suppliers):
    return [(d.name, d.price, s.name)
            for s in suppliers
            for d in details
            if d.id_supplier == s.id]

def many_to_many(details, suppliers, details_suppliers):
    temp = [(s.name, ds.id_supplier, ds.id_detail)
            for s in suppliers
            for ds in details_suppliers
            if s.id == ds.id_supplier]

    return [(d.name, d.price, supplier_name)
            for supplier_name, _, id_detail in temp
            for d in details if d.id == id_detail]

def task1(one_to_many_data):
    return sorted(
        [(name, supplier) for name, _, supplier in one_to_many_data if name.startswith('T')],
        key=itemgetter(1)
    )

def task2(one_to_many_data, suppliers):
```

```
    result = []
    for s in suppliers:
        s_details = list(filter(lambda i: i[2] == s.name, one_to_many_data))
        if s_details:
            min_price = min([price for _, price, _ in s_details])
            result.append((s.name, min_price))
    return sorted(result, key=itemgetter(1))

def task3(many_to_many_data, suppliers):
    result = {}
    for s in suppliers:
        s_details = list(filter(lambda i: i[2] == s.name, many_to_many_data))
        s_details_names = [x for x, _, _ in s_details]
        result[s.name] = sorted(s_details_names)
    return result
```

test_main.py

```
import unittest
from main import *

class Test(unittest.TestCase):

    def setUp(self):
        self.details = [
            Detail(1, 'Тормозная колодка', 1000, 1),
            Detail(2, 'Свеча зажигания', 4000, 3),
            Detail(4, 'Решетка радиатора', 8000, 2),
            Detail(3, 'Подшипник', 2500, 2),
            Detail(5, 'Топливный бак', 5000, 3),
```

```
        ]

        self.suppliers = [
            Supplier(1, 'Автотрейд'),
            Supplier(2, 'Берг'),
            Supplier(3, 'Автопитер')
        ]

        self.details_suppliers = [
            DetailSupplier(1, 1),
            DetailSupplier(2, 3),
            DetailSupplier(4, 2),
            DetailSupplier(3, 2),
            DetailSupplier(5, 3),
        ]

    def test_task1(self):
        one_to_many_data = one_to_many(self.details, self.suppliers)
        result = task1(one_to_many_data)
        expected = [('Топливный бак', 'Автопитер'), ('Тормозная колодка', 'Автотрейд')]
        self.assertEqual(result, expected)

    def test_task2(self):
        one_to_many_data = one_to_many(self.details, self.suppliers)
        result = task2(one_to_many_data, self.suppliers)
        expected = [('Автотрейд', 1000), ('Берг', 2500), ('Автопитер', 4000)]
        self.assertEqual(result, expected)

    def test_task3(self):
```

```
        many_to_many_data = many_to_many(self.details, self.suppliers, self.details_suppliers)
        result = task3(many_to_many_data, self.suppliers)
        expected = {
            'Автотрейд': ['Тормозная колодка'],
            'Берг': ['Подшипник', 'Решетка радиатора'],
            'Автопитер': ['Свеча зажигания', 'Топливный бак']
        }
        self.assertEqual(result, expected)


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

Результат выполнения

```
C:\Users\Akhme\OneDrive\文档\GitHub\programming-paradigm\RK2>python -m unittest -v
test_task1 (test_main.Test.test_task1) ... ok
test_task2 (test_main.Test.test_task2) ... ok
test_task3 (test_main.Test.test_task3) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s
```