|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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 «Информатика и системы управления»

ДИСЦИПЛИНА:

«Базовые компоненты ИТ»

**Рубежная контроль № 2**

Студент Макеев В. А. ИУ5Ц-54Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(И.О. Фамилия) (Группа) (Подпись, дата)

Преподаватель Гапанюк Ю.Е.  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(И.О. Фамилия) (Подпись, дата)

Москва, 2021г.

**Описание задания**

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).

**main.py**

# используется для сортировки

from operator import itemgetter

"""Деталь"""

class Detail:

    def \_\_init\_\_(self, id, name1, price, provider\_id):

        self.id=id

        self.name1=name1

        self.price=price

        self.provider\_id=provider\_id

"""Поставщик"""

class Provider:

    def \_\_init\_\_(self, id, name):

        self.id=id

        self.name=name

"""Детали поставщика"""

class DetailProvider:

    def \_\_init\_\_(self, provider\_id, detail\_id):

        self.provider\_id=provider\_id

        self.detail\_id=detail\_id

#поставщики (id поставщика, название поставщика)

providers=[Provider(1,'АвтоСпейс'),

Provider(2,'Favorit-auto'),

Provider(3,'Автотрейд'),

Provider(11,'Next-auto'),

Provider(22,'Гарант-Авто'),

Provider(33,'Forum-Auto'),

]

#детали (id детали, название детали, цена, id поставщика)

details=[Detail(1,'сцепление',4000,1),

Detail(2,'маховик',2000,3),

Detail(3,'поршень',15000,3),

Detail(4,'колодка',1500,2),

Detail(5,'подвеска',14000,1),

]

#детали поставщика (id поставщика,id детали)

details\_providers=[DetailProvider(1,1),

DetailProvider(2,2),

DetailProvider(3,3),

DetailProvider(4,4),

DetailProvider(5,5),

DetailProvider(22,1),

DetailProvider(11,2),

DetailProvider(33,3),

DetailProvider(33,4),

DetailProvider(11,5),

]

def sort\_name(detail):

    res\_11 = [(p.name,list(name1 for name1,\_,name in detail if name == p.name)) for p in providers if p.name[0] == 'А']

    return res\_11

def sort\_price(detail, providers):

    res\_12\_unsorted = []

 # Перебираем все поставщики

    for s in providers:

 # Список деталей поставщика

        d\_details = list(filter(lambda i: i[2] == s.name, detail))

 # Если поставщик не пустой

        if len(d\_details) > 0:

            res\_12\_unsorted.append((s.name, max(d\_details, key=lambda x: x[1])[1]))

    return sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)

def sort\_provider(detail):

    res\_13 = []

 # Перебираем все детали

    return sorted(detail, key=lambda entry: entry[2])

def main():

    one\_to\_many=[(d.name1,d.price,p.name)

        for p in providers

        for d in details

        if d.provider\_id == p.id

    ]

    many\_to\_many\_temp=[(p.name, dp.provider\_id,dp.detail\_id)

        for p in providers

        for dp in details\_providers

        if p.id==dp.provider\_id

    ]

    many\_to\_many=[(d.name1,d.price,provider\_name)

        for provider\_name, provider\_id, detail\_id in many\_to\_many\_temp

        for d in details if d.id == detail\_id

    ]

    print('\nЗадание Г1\n',sort\_name(one\_to\_many))

    print('\nЗадание Г2\n', sort\_price(one\_to\_many, providers))

    print('\nЗадание Г3\n', sort\_provider(many\_to\_many))

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**tests.py**

import unittest

import sys, os

sys.path.append(os.getcwd())

from main import \*

one\_to\_many=[(d.name1,d.price,p.name)

        for p in providers

        for d in details

        if d.provider\_id == p.id]

many\_to\_many\_temp=[(p.name, dp.provider\_id,dp.detail\_id)

        for p in providers

        for dp in details\_providers

        if p.id==dp.provider\_id]

many\_to\_many=[(d.name1,d.price,provider\_name)

        for provider\_name, provider\_id, detail\_id in many\_to\_many\_temp

        for d in details if d.id == detail\_id]

class TestCost(unittest.TestCase):

    def test\_sort\_name(self):

        self.assertEqual(sort\_name(one\_to\_many),  [('АвтоСпейс', ['сцепление', 'подвеска']), ('Автотрейд', ['маховик', 'поршень'])])

    def test\_sort\_price(self):

        self.assertEqual(sort\_price(one\_to\_many, providers), [('Автотрейд', 15000), ('АвтоСпейс', 14000), ('Favorit-auto', 1500)])

    def test\_sort\_provider(self):

        self.assertEqual(sort\_provider(many\_to\_many), [('маховик', 2000, 'Favorit-auto'), ('поршень', 15000, 'Forum-Auto'),

                                                       ('колодка', 1500, 'Forum-Auto'), ('маховик', 2000, 'Next-auto'),

                                                       ('подвеска', 14000, 'Next-auto'), ('сцепление', 4000, 'АвтоСпейс'),

                                                       ('поршень', 15000, 'Автотрейд'), ('сцепление', 4000, 'Гарант-Авто')])

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

**Результат выполнения программы:**

