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] == 'A']
    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: