**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»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю №2

Выполнил:

студент группы ИУ5-33Б

Требуков Д.А.

Проверил:

Преподаватель каф. ИУ5

Гапанюк Юрий Евгеньевич

Москва, 2021 г.

***Задание:***

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:***

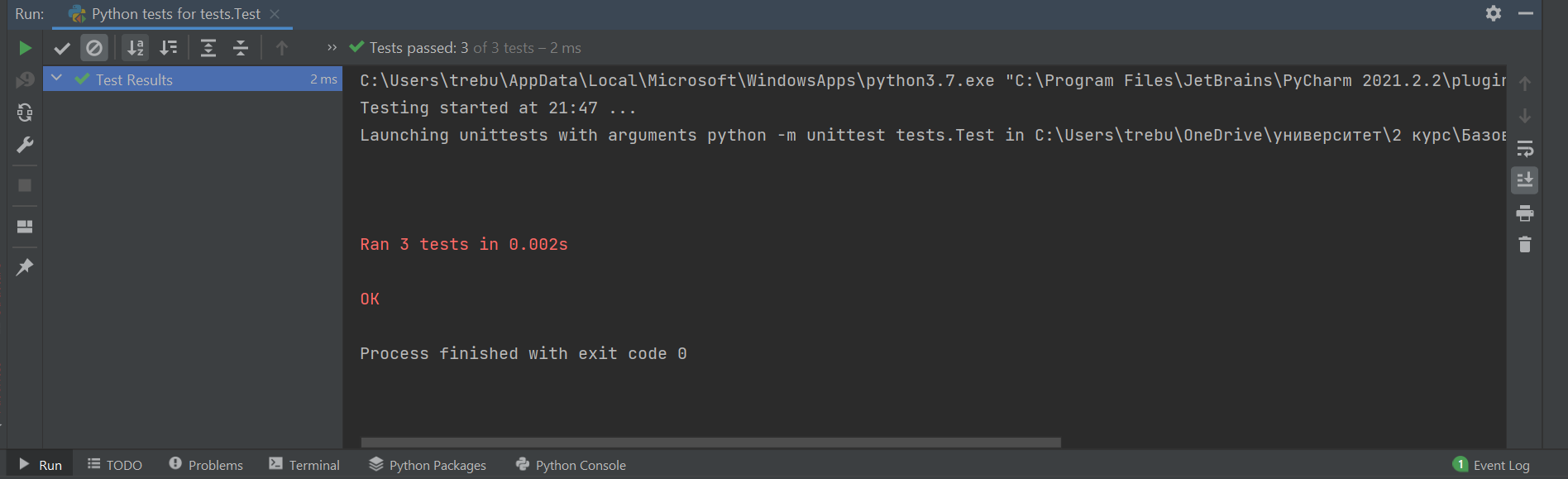
**main.py**

from operator import itemgetter  
  
  
class Detail:  
 *"""Детали"""* def \_\_init\_\_ (self, id, name, value, manufacturer\_id):  
 self.id = id  
 self.name = name  
 self.value = value  
 self.manufacturer\_id = manufacturer\_id  
  
class Manufacturer:  
 *"""Производители"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
class ManufactDetail:  
 *"""Детали от производителей"""* def \_\_init\_\_(self, manufacturer\_id, detail\_id):  
 self.manufacturer\_id = manufacturer\_id  
 self.detail\_id = detail\_id  
  
#Производители  
manufacturers = [  
 Manufacturer(1, 'Schlieckmann'),  
 Manufacturer(2, 'Klokkerholm'),  
 Manufacturer(3, 'Signeda'),  
  
 Manufacturer(11, 'Sigma'),  
 Manufacturer(22, 'Tyg'),  
 Manufacturer(33, 'Kito')  
]  
#Детали  
details = [  
 Detail(1, 'Кузов', 15000, 1),  
 Detail(2, 'Поршни', 20000, 1),  
 Detail(3, 'Ремень ГРМ', 5000, 2),  
 Detail(4, 'Подушки двигателя', 25000, 3),  
 Detail(5, 'Система охлаждения', 7000, 3)  
]  
  
detail\_manufacs = [  
 ManufactDetail(1, 1),  
 ManufactDetail(1, 2),  
 ManufactDetail(2, 3),  
 ManufactDetail(3, 4),  
 ManufactDetail(3, 5),  
  
 ManufactDetail(11, 1),  
 ManufactDetail(22, 2),  
 ManufactDetail(22, 3),  
 ManufactDetail(33, 4),  
 ManufactDetail(33, 5)  
]  
  
def res\_1(arr):  
 answer\_list = []  
 for detail, value, manufacturer in arr:  
 if detail[0] == 'П':  
 answer\_list.append([detail, manufacturer])  
 return answer\_list  
  
def res\_2(arr):  
 answer\_list = [[arr[0][2], arr[0][1]]]  
 for detail, value, manufacturer in arr:  
 if manufacturer == answer\_list[len(answer\_list)-1][0]:  
 if value < answer\_list[len(answer\_list)-1][1]:  
 answer\_list[len(answer\_list)-1][1]= value  
 else:  
 answer\_list.append([manufacturer, value])  
 return sorted(answer\_list, key=itemgetter(1))  
  
def res\_3(arr):  
 answer\_list = []  
 for detial, value, manufacturer in arr:  
 answer\_list.append([detial, manufacturer])  
 return sorted(answer\_list, key=itemgetter(0))  
def main():  
 # Соединение данных один-ко-многим  
 one\_to\_many = [(d.name, d.value, m.name)  
 for m in manufacturers  
 for d in details  
 if d.manufacturer\_id == m.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(m.name, md.manufacturer\_id, md.detail\_id)  
 for m in manufacturers  
 for md in detail\_manufacs  
 if m.id == md.manufacturer\_id]  
  
 many\_to\_many = [(d.name, d.value, m\_name)  
 for m\_name, m\_id, d\_id in many\_to\_many\_temp  
 for d in details if d.id == d\_id]  
  
 print('Задание В1')  
 """  
 «Производитель» и «Деталь» связаны соотношением один-ко-многим.   
 Выведите список всех деталей, у которых названия   
 начинаются с буквы «П», и названия их производителей.  
 """  
 print(res\_1(one\_to\_many))  
  
 print('Задание В2')  
 """   
 «Производитель» и «Деталь» связаны соотношением один-ко-многим.   
 Выведите список производителей с минимальной стоимостью деталей   
 у каждого производителя, отсортированный по минимальной стоимости.  
 """  
 print(res\_2(one\_to\_many))  
  
 print('Задание В3')  
 """  
 «Производитель» и «Деталь» связаны соотношением многие-ко-многим.   
 Выведите список всех связанных производителей и деталей,   
 отсортированный по деталям, сортировка по производителям произвольная.   
 """  
 print(res\_3(many\_to\_many))  
  
if \_\_name\_\_=="\_\_main\_\_":  
 main()

**tests.py**

from main import Detail, Manufacturer, ManufactDetail, res\_1, res\_2, res\_3  
from unittest import TestCase  
  
class Test(TestCase):  
 def setUp(self) -> None:  
 # Производители  
 self.manufacturers = [  
 Manufacturer(1, 'Schlieckmann'),  
 Manufacturer(2, 'Klokkerholm'),  
 Manufacturer(3, 'Signeda'),  
  
 Manufacturer(11, 'Sigma'),  
 Manufacturer(22, 'Tyg'),  
 Manufacturer(33, 'Kito')  
 ]  
 # Детали  
 self.details = [  
 Detail(1, 'Кузов', 15000, 1),  
 Detail(2, 'Поршни', 20000, 1),  
 Detail(3, 'Ремень ГРМ', 5000, 2),  
 Detail(4, 'Подушки двигателя', 25000, 3),  
 Detail(5, 'Система охлаждения', 7000, 3)  
 ]  
 self.detail\_manufacs = [  
 ManufactDetail(1, 1),  
 ManufactDetail(1, 2),  
 ManufactDetail(2, 3),  
 ManufactDetail(3, 4),  
 ManufactDetail(3, 5),  
  
 ManufactDetail(11, 1),  
 ManufactDetail(22, 2),  
 ManufactDetail(22, 3),  
 ManufactDetail(33, 4),  
 ManufactDetail(33, 5)  
 ]  
 self.one\_to\_many = [(d.name, d.value, m.name)  
 for m in self.manufacturers  
 for d in self.details  
 if d.manufacturer\_id == m.id]  
 self.many\_to\_many\_temp = [(m.name, md.manufacturer\_id, md.detail\_id)  
 for m in self.manufacturers  
 for md in self.detail\_manufacs  
 if m.id == md.manufacturer\_id]  
 self.many\_to\_many = [(d.name, d.value, m\_name)  
 for m\_name, m\_id, d\_id in self.many\_to\_many\_temp  
 for d in self.details if d.id == d\_id]  
  
 def test1(self):  
 result = res\_1(self.one\_to\_many)  
 desired = [['Поршни', 'Schlieckmann'], ['Подушки двигателя', 'Signeda']]  
 self.assertEqual(result, desired)  
  
 def test2(self):  
 result = res\_2(self.one\_to\_many)  
 desired = [['Klokkerholm', 5000], ['Signeda', 7000], ['Schlieckmann', 15000]]  
 self.assertEqual(result, desired)  
  
 def test3(self):  
 result = res\_3(self.many\_to\_many)  
 desired = [['Кузов', 'Schlieckmann'], ['Кузов', 'Sigma'],  
 ['Подушки двигателя', 'Signeda'], ['Подушки двигателя', 'Kito'],  
 ['Поршни', 'Schlieckmann'], ['Поршни', 'Tyg'], ['Ремень ГРМ', 'Klokkerholm'],  
 ['Ремень ГРМ', 'Tyg'], ['Система охлаждения', 'Signeda'], ['Система охлаждения', 'Kito']]  
 self.assertEqual(result, desired)

***Результаты программы:***

******