**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»**

Рубежный контроль №2

по дисциплине «Базовые компоненты интернет-технологий»

Выполнила:

студентка группы ИУ5-33Б

Жамнова М. С.

Проверил:

Гапанюк Ю. Е.

2021 г.

**Условия рубежного контроля №2 по курсу БКИТ**

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

# используется для сортировки  
from operator import itemgetter  
  
  
class Comp:  
 *"""Компьютер"""* def \_\_init\_\_(self, id, brand, price, disp\_cls\_id):  
 self.id = id  
 self.brand = brand  
 self.price = price  
 self.disp\_cls\_id = disp\_cls\_id  
  
  
class Disp\_cls:  
 *"""Дисплейный класс"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class CompDisp\_cls:  
 *"""  
 'Компьютеры дисплейного класса' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, disp\_cls\_id, comp\_id):  
 self.disp\_cls\_id = disp\_cls\_id  
 self.comp\_id = comp\_id  
  
  
# Дисплейные классы  
disp\_classes = [  
 Disp\_cls(1, 'А-класс'),  
 Disp\_cls(2, 'Б-класс'),  
 Disp\_cls(3, 'В-класс'),  
 Disp\_cls(11, 'Г-класс'),  
 Disp\_cls(22, 'Д-класс'),  
 Disp\_cls(33, 'Е-класс'),  
  
]  
  
# Компьютеры  
comps = [  
 Comp(1, 'HP', 1168390, 1),  
 Comp(2, 'Asus', 55390, 1),  
 Comp(3, 'ThinkPad',78940, 3),  
 Comp(4, 'Acer', 97450, 2),  
 Comp(5, 'Lenovo', 86980, 3),  
]  
  
# Компьютеры и Дисплейные классы для связи многие-ко-многим  
comps\_disp\_classs = [  
 CompDisp\_cls(1, 1),  
 CompDisp\_cls(1, 2),  
 CompDisp\_cls(3, 3),  
 CompDisp\_cls(2, 4),  
 CompDisp\_cls(3, 5),  
  
 CompDisp\_cls(33, 1),  
 CompDisp\_cls(22, 2),  
 CompDisp\_cls(11, 3),  
 CompDisp\_cls(33, 4),  
 CompDisp\_cls(22, 5),  
]  
  
def first\_task(one\_to\_many):  
 task1 = []  
 for brand, price, name in one\_to\_many:  
 if brand[0] == "A":  
 task1.append((brand, name))  
 return task1  
  
  
def second\_task(one\_to\_many):  
 task2\_uns = []  
 for c in disp\_classes:  
 # все компьютеры  
 d\_disp\_cls = list(filter(lambda i: i[2] == c.name, one\_to\_many))  
 if len(d\_disp\_cls) > 0:  
 c\_price = [price for \_, price, \_ in d\_disp\_cls]  
 c\_minPrice = min(c\_price)  
 task2\_uns.append((c.name, c\_minPrice))  
 task2 = sorted(task2\_uns, key=itemgetter(1))  
 return task2  
  
  
def third\_task(many\_to\_many):  
 task3\_uns = []  
 for brand, price, name in many\_to\_many:  
 task3\_uns.append((brand, name))  
  
 task3 = list(sorted(task3\_uns, key=itemgetter(0)))  
 return task3  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(c.brand, c.price, d.name)  
 for d in disp\_classes  
 for c in comps  
 if c.disp\_cls\_id == d.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(d.name, dc.disp\_cls\_id, dc.comp\_id)  
 for d in disp\_classes  
 for dc in comps\_disp\_classs  
 if d.id == dc.disp\_cls\_id]  
  
 many\_to\_many = [(c.brand, c.price, disp\_class\_name)  
 for disp\_class\_name, disp\_cls\_id, comp\_id in many\_to\_many\_temp  
 for c in comps if c.id == comp\_id]  
  
 print('\nЗадание 1 \n', first\_task(one\_to\_many))  
 print('\nЗадание 2 \n', second\_task(one\_to\_many))  
 print('\nЗадание 3 \n', third\_task(one\_to\_many))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

test\_tdd.py

import unittest  
import sys, os  
  
sys.path.append(os.getcwd())  
from main import \*  
  
  
class TddTest(unittest.TestCase):  
 def test\_first\_task(self):  
 one\_to\_many = [(c.brand, c.price, d.name)  
 for d in disp\_classes  
 for c in comps  
 if c.disp\_cls\_id == d.id]  
 self.assertEqual(first\_task(one\_to\_many), [('Asus', 'А-класс'), ('Acer', 'Б-класс')])  
  
 def test\_2(self):  
 one\_to\_many = [(c.brand, c.price, d.name)  
 for d in disp\_classes  
 for c in comps  
 if c.disp\_cls\_id == d.id]  
 self.assertEqual(second\_task(one\_to\_many), [('А-класс', 55390), ('В-класс', 78940), ('Б-класс', 97450)])  
  
 def test\_3(self):  
 many\_to\_many\_temp = [(d.name, dc.disp\_cls\_id, dc.comp\_id)  
 for d in disp\_classes  
 for dc in comps\_disp\_classs  
 if d.id == dc.disp\_cls\_id]  
 many\_to\_many = [(c.brand, c.price, disp\_class\_name)  
 for disp\_class\_name, disp\_cls\_id, comp\_id in many\_to\_many\_temp  
 for c in comps if c.id == comp\_id]  
 self.assertEqual(third\_task(many\_to\_many),  
 [('Acer', 'Б-класс'), ('Acer', 'Е-класс'), ('Asus', 'А-класс'), ('Asus', 'Д-класс'),  
 ('HP', 'А-класс'), ('HP', 'Е-класс'), ('Lenovo', 'В-класс'), ('Lenovo', 'Д-класс'),  
 ('ThinkPad', 'В-класс'), ('ThinkPad', 'Г-класс')])

**Скрины результата работы:**

