**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»

Выполнил:

студент группы ИУ5-33Б Носкин С.А.

Проверил: Гапанюк Ю.Е.

2021 г.

**Задание:**

Рубежный контроль представляет собой разработку тестов на языке Python.

1. Проведите рефакторинг текста программы рубежного контроля №1 такимобразом, чтобы он был пригоден для модульного тестирования.
2. Для текста программы рубежного контроля №1 создайте модульные тестыс применением TDD - фреймворка (3 теста).

**Текст программы:**

main.py

from operator import itemgetter  
  
class Browser:  
  
 *"""Браузер"""* def \_\_init\_\_(self, id, naz, kol\_str, kol\_zap):  
 self.id = id  
 self.naz = naz  
 self.kol\_str = kol\_str  
 self.kol\_zap = kol\_zap  
  
  
class Computer:  
 *""" Компьютеры """* def \_\_init\_\_(self, id, mode):  
 self.id = id  
 self.mode = mode # виды компьютеров  
  
  
class ComputerBrowser:  
  
 def \_\_init\_\_(self, browsers\_id, computers\_id):  
 self.browsers\_id = browsers\_id  
 self.computers\_id = computers\_id  
  
  
# Компьютеры  
computers = [  
 Computer(1, 'Асус'),  
 Computer(2, 'МСИ'),  
 Computer(3, 'Асер'),  
 Computer(4, 'Леново'),  
]  
  
# Браузеры  
browsers = [  
 Browser(1, 'Мозилла', 50, 5),  
 Browser(2, 'Яндекс', 30, 3),  
 Browser(3, 'Гугл', 45, 4),  
 Browser(4, 'Амиго', 44, 4),  
 Browser(5, 'Ёхо', 45, 2),  
  
]  
  
computers\_browsers = [  
 ComputerBrowser(1, 4),  
 ComputerBrowser(3, 2),  
 ComputerBrowser(3, 3),  
 ComputerBrowser(4, 5),  
 ComputerBrowser(2, 1),  
]  
def Task1(one\_to\_many):  
 print('Задание В1')  
 res\_11 = []  
 for naz, kol\_str, computers\_name in one\_to\_many:  
 if 'А' in naz[0]:  
 res\_11.append((naz, computers\_name))  
 return res\_11  
def Task2(one\_to\_many):  
 print('Задание В2')  
 buff = []  
 for c in computers:  
 # список видов компьютеров  
 c\_modes = list(filter(lambda i: i[2] == c.mode, one\_to\_many))  
 if len(c\_modes) > 0:  
 c\_kol\_str = [kol\_str for \_, kol\_str, \_ in c\_modes]  
 min\_kol\_str = min(c\_kol\_str)  
 buff.append((c.mode, min\_kol\_str))  
 res\_12 = sorted(buff, key=itemgetter(1))  
 return res\_12  
def Task3(many\_to\_many):  
 print('Задание В3')  
 buff = []  
 for naz, kol\_str, computers\_name in many\_to\_many:  
 buff.append((naz, computers\_name))  
 res\_13 = list(sorted(buff, key=itemgetter(0)))  
 return res\_13  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(b.naz, b.kol\_str, c.mode)  
 for c in computers  
 for b in browsers  
 if b.kol\_zap == c.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(c.mode, cb.computers\_id, cb.browsers\_id)  
 for c in computers  
 for cb in computers\_browsers  
 if c.id == cb.computers\_id]  
  
 many\_to\_many = [(b.naz, b.kol\_str, computers\_name)  
 for computers\_name, computers\_id, browsers\_id in many\_to\_many\_temp  
 for b in browsers if b.id == browsers\_id]  
  
 print('Test') # вывод списков со связями 1-м, м-м  
 res\_0 = (one\_to\_many)  
 print(res\_0)  
 res\_01 = (many\_to\_many)  
 print(res\_01)  
 Task1(one\_to\_many)  
 Task2(one\_to\_many)  
 Task3(many\_to\_many)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

test.py

import unittest  
import sys, os  
  
sys.path.append(os.getcwd())  
from main import \*  
  
  
class Test\_task\_1(unittest.TestCase):  
 def test\_task\_1(self):  
 one\_to\_many = [(b.naz, b.kol\_str, c.mode)  
 for c in computers  
 for b in browsers  
 if b.kol\_zap == c.id]  
  
 self.assertEqual(Task1(one\_to\_many), [('Амиго', 'Леново')])  
  
class Test\_task\_2(unittest.TestCase):  
 def test\_task\_2(self):  
 one\_to\_many = [(b.naz, b.kol\_str, c.mode)  
 for c in computers  
 for b in browsers  
 if b.kol\_zap == c.id]  
  
 self.assertEqual(Task2(one\_to\_many),[('Асер', 30), ('Леново', 44), ('МСИ', 45)])  
  
  
class Test\_task\_3(unittest.TestCase):  
 def test\_task\_3(self):  
 many\_to\_many\_temp = [(c.mode, cb.computers\_id, cb.browsers\_id)  
 for c in computers  
 for cb in computers\_browsers  
 if c.id == cb.computers\_id]  
  
 many\_to\_many = [(b.naz, b.kol\_str, computers\_name)  
 for computers\_name, computers\_id, browsers\_id in many\_to\_many\_temp  
 for b in browsers if b.id == browsers\_id]  
 self.assertEqual(Task3(many\_to\_many),  
 [('Гугл', 'МСИ'), ('Гугл', 'Асер'), ('Мозилла', 'Леново'), ('Яндекс', 'Асус')])  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

Результат выполнения:

