**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 ИУ5-32Б

Ким А.М.

2021 г.

# Задание рубежного контроля №2:

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).

# Задание рубежного контроля № 1:

**Вариант Б.**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с количеством сотрудников в каждом отделе, отсортированный по количеству сотрудников.

3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.

**Вариант предметной области** 8.

Жесткий диск - Компьютер

# Программа:

Файл main.py:

**from** operator **import** itemgetter

**class** hd:

*"""Жесткий диск"""*

**def** \_\_init\_\_(self, id, mf, cost, comp\_id):

       self.id = id

       self.mf = mf

       self.cost = cost

       self.comp\_id = comp\_id

**class** comp:

*"""Компьютер"""*

**def** \_\_init\_\_(self, id, OS):

       self.id = id

       self.OS = OS

**class** hdcomp:

*"""*

*'Жесткие диски компютеров' для реализации*

*связи многие-ко-многим*

*"""*

**def** \_\_init\_\_(self, disk\_id, comp\_id):

       self.disk\_id = disk\_id

       self.comp\_id = comp\_id

comps = [

   comp(1, **'Lunix'**),

   comp(2, **'Windows'**),

   comp(3, **'Mac OS'**),

]

hds = [

   hd(1, **'Seagate'**, 3500, 1),

   hd(2, **'WD'**, 2500, 2),

   hd(3, **'Toshiba'**, 4500, 1),

   hd(4, **'Apple'**, 10000, 3),

]

hdcomps = [

   hdcomp(1, 1),

   hdcomp(2, 2),

   hdcomp(3, 1),

   hdcomp(4, 3),

   hdcomp(2, 3),

   hdcomp(3, 2),

]

**"""Основная функция"""**

**def** one\_to\_many\_connection():

*# Соединение данных один-ко-многим*

one\_to\_many = [(e.mf, e.cost, d.OS)

**for** d **in** comps

**for** e **in** hds

**if** e.comp\_id == d.id]

**return** one\_to\_many

**def** many\_to\_many\_connection():

*# Соединение данных многие-ко-многим*

many\_to\_many\_temp = [(d.OS, ed.comp\_id, ed.disk\_id)

**for** d **in** comps

**for** ed **in** hdcomps

**if** d.id == ed.comp\_id]

   many\_to\_many = [(e.mf, e.cost, OS)

**for** OS, comp\_id, disk\_id **in** many\_to\_many\_temp

**for** e **in** hds **if** e.id == disk\_id]

**return** many\_to\_many

**def** task1(one\_to\_many):

   res\_11 = sorted(one\_to\_many, key=itemgetter(2))

**return** res\_11

**def** task2(one\_to\_many):

   res2unsorted = []

**for** c **in** comps:

*# Список диском компьютера*

hds = list(filter(**lambda** i: i[2] == c.OS, one\_to\_many))

       count = len(hds)

       res2unsorted.append((c.OS, count))

*# Cортировка по кол-ву дисков*

res2 = sorted(res2unsorted, key=itemgetter(1), reverse=**True**)

**return** res2

**def** task3(many\_to\_many):

   b = [j **for** j **in** many\_to\_many **if** j[0][-1:] == **'e'**]

   res\_13 = {j[2]: [i[0] **for** i **in** b **if** i[2] == j[2]] **for** j **in** b}

**return** res\_13

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

   one\_to\_many = one\_to\_many\_connection()

   many\_to\_many = many\_to\_many\_connection()

   print(**'Задание Б1\n{}'**.format(task1(one\_to\_many)))

   print(**'Задание Б2\n{}'**.format(task2(one\_to\_many)))

   print(**'Задание Б3\n{}'**.format(task3(many\_to\_many)))

Файл tdd.py:

**import** unittest

**import** main

**class** TestCompsHDs(unittest.TestCase):

**def** test\_task1(self):

       relations = main.one\_to\_many\_connection()

       expected\_result = [(**'Seagate'**, 3500, **'Lunix'**), (**'Toshiba'**, 4500, **'Lunix'**), (**'Apple'**, 10000, **'Mac OS'**), (**'WD'**, 2500, **'Windows'**)]

       self.assertEqual(main.task1(relations), expected\_result)

**def** test\_task2(self):

       relations = main.one\_to\_many\_connection()

       expected\_result = [(**'Lunix'**, 2), (**'Windows'**, 1), (**'Mac OS'**, 1)]

       self.assertEqual(main.task2(relations), expected\_result)

**def** test\_task3(self):

       relations = main.many\_to\_many\_connection()

       expected\_result = {**'Lunix'**: [**'Seagate'**], **'Mac OS'**: [**'Apple'**]}

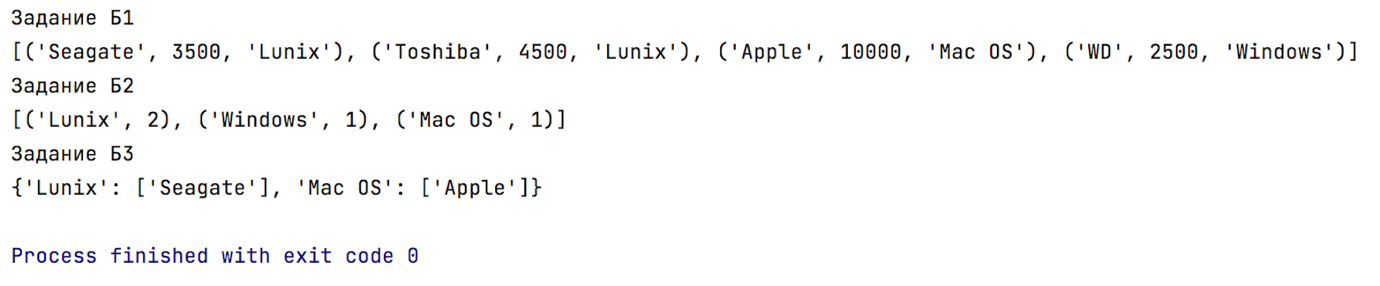
       self.assertEqual(main.task3(relations), expected\_result)

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

   unittest.main()

# Результаты выполнения программы:

Файл main.py:



Файл tdd.py:

