**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"""
```

```python
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 дисков компьютера
        hds = list(filter(lambda i: i[2] == c.OS, one_to_many))
        count = len(hds)
        res2unsorted.append((c.OS, count))

    # Сортировка 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:

```
Задание Б1
[('Seagate', 3500, 'Lunix'), ('Toshiba', 4500, 'Lunix'), ('Apple', 10000, 'Mac OS'), ('WD', 2500, 'Windows')]
Задание Б2
[('Lunix', 2), ('Windows', 1), ('Mac OS', 1)]
Задание Б3
{'Lunix': ['Seagate'], 'Mac OS': ['Apple']}

Process finished with exit code 0
```

Файл tdd.py:

```
Testing started at 16:35 ...
Launching unittests with arguments python -m unittest



Ran 3 tests in 0.003s

OK

Process finished with exit code 0
```