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**

О Т Ч Е Т
по рубежному контролю № 2

Дисциплина: «Базовые компоненты ИТ»

Студент	ИУ5Ц-52Б		А.Н. Свинцов
	(Группа)	(дата)	(И.О. Фамилия)
Преподаватель			Ю.Е. Гапанюк
		(дата)	(И.О. Фамилия)

Москва, 2021

Описание задания

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).

Код программы

main.py

```
"""РК №1, Свинцов Артемий ИУ5Ц-52Б
Вариант Б, вариант предметной области 25"""

from operator import itemgetter

class part:
    def __init__(self, id, name_p):
        self.id = id
        self.name_p = name_p

class docum:
    def __init__(self, id, name, page_num, part_id):
        self.id = id
        self.name = name
        self.page_num = page_num
        self.part_id = part_id

class part_doc:
    def __init__(self, docum_id, part_id):
        self.docum_id = docum_id
        self.part_id = part_id

parts = [
    part(1, 'Кодекс'),
    part(2, 'Устав организации'),
    part(3, 'Федеральный закон'),
    part(4, 'Нормативный акт'),
]

docums = [
    docum(1, 'Уголовный', 15, 1),
    docum(2, 'Об образовании', 20, 3),
    docum(3, 'Организация труда', 10, 2),
    docum(4, 'Запрет продажи алкоголя', 31, 4),
    docum(5, 'О внутреннем распорядке', 16, 2),
]

part_docs = [
    part_doc(1, 1),
    part_doc(2, 3),
    part_doc(3, 2),
    part_doc(4, 1),
    part_doc(5, 2),
]

def main():
```

```
    one_to_many = [(g.name, g.page_num, c.name_p)
        for g in docums
        for c in parts
        if g.part_id == c.id]

    many_to_many_temp = [(c.name_p, gc_s.part_id, gc_s.docum_id)
        for c in parts
        for gc_s in part_docs
        if c.id == gc_s.part_id]

    many_to_many = [(g.name, g.page_num, part_name)
        for part_name, part_id, docum_id in many_to_many_temp
        for g in docums if g.id == docum_id]

    print('Задание Б1')
    print(B1(one_to_many))
    print('Задание Б2')
    print(B2(one_to_many))
    print('Задание Б3')
    print(B3(many_to_many))
def B1(one_to_many):
    res_11 = sorted(one_to_many, key=itemgetter(0))
    return res_11

def B2(one_to_many):
    res_12_unsorted = []
    for c in parts:
        parts_s = list(filter(lambda i: i[2]==c.name_p, one_to_many))
        if len(parts_s) > 0:
            parts_amount = [page_num for _,page_num,_ in parts_s]
            parts_amount_sum = sum(parts_amount)
            res_12_unsorted.append((c.name_p, parts_amount_sum))

    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def B3(many_to_many):
    res_13 = {}
    for c in parts:
        if len(c.name_p) > 0:
            part_s = list(filter(lambda i: i[2]==c.name_p, many_to_many))
            part_s_name = [x for x,_,_ in part_s]
            res_13[c.name_p] = part_s_name
    return res_13

if __name__ == '__main__':
    main()
```

TDD.py

```
import unittest
import sys, os
sys.path.append(os.getcwd())
from main import *


class Test(unittest.TestCase):
    def test_B1(self):
        one_to_many = [(g.name, g.page_num, c.name_p)
                       for g in docums
                       for c in parts
                       if g.part_id == c.id]
```

```
        self.assertEqual(B1(one_to_many), [('Запрет продажи алкоголя', 31,
'Нормативный акт'), ('О внутреннем распорядке', 16, 'Устав организации'),
('Об образовании', 20, 'Федеральный закон'), ('Организация труда', 10, 'Устав
организации'), ('Уголовный', 15, 'Кодекс')])


    def test_B2(self):
        one_to_many = [(g.name, g.page_num, c.name_p)
                       for g in docums
                       for c in parts
                       if g.part_id == c.id]
        self.assertEqual(B2(one_to_many), [('Нормативный акт', 31), ('Устав
организации', 26), ('Федеральный закон', 20), ('Кодекс', 15)])

    def test_B3(self):
        many_to_many_temp = [(c.name_p, gc_s.part_id, gc_s.docum_id)
                             for c in parts
                             for gc_s in part_docs
                             if c.id == gc_s.part_id]
        many_to_many = [(g.name, g.page_num, part_name)
                        for part_name, part_id, docum_id in many_to_many_temp
                        for g in docums if g.id == docum_id]
        self.assertEqual(B3(many_to_many),{ 'Кодекс': ['Уголовный', 'Запрет
продажи алкоголя'], 'Устав организации': ['Организация труда', 'О внутреннем
распорядке'], 'Федеральный закон': ['Об образовании'], 'Нормативный акт':
[]})

if __name__ == '__main__':
    unittest.main()
```

Результат выполнения программы

```
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK

Process finished with exit code 0
```