**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 «Системы обработки информации и управления»**

**Рубежный контроль №2.**

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-31Б

Целуйко Ульяна

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

**Условия РК №1**

**Вариант А. Предметная область 25**

1. «Документ» и «Раздел» связаны соотношением один-ко-многим. Выведите список всех связанных разделов и документов, отсортированный по документам, сортировка по разделам произвольная.
2. «Документ» и «Раздел» связаны соотношением один-ко-многим. Выведите список документов с суммарным количеством символов в каждом документе, отсортированный по количеству символов.
3. «Документ» и «Раздел» связаны соотношением многие-ко-многим. Выведите список всех документов, у которых в названии присутствует слово «Книга», и список разделов в нем.

**Условия РК №2**

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**

from operator import itemgetter  
  
class Sec:  
 def \_\_init\_\_(self, Id, Names, Symbols, Id\_Sec):  
 self.id =Id  
 self.names =Names  
 self.symb=Symbols  
 self.id\_sec=Id\_Sec  
   
class Doc:  
 def \_\_init\_\_(self, Id, Named):  
 self.id=Id  
 self.named=Named  
   
class SecDoc:  
 def \_\_init\_\_(self, Id\_Doc, Id\_Sec):  
 self.id\_doc=Id\_Doc  
 self.id\_sec=Id\_Sec  
   
# Разделы  
secs = [  
 Sec(1, 'От автора', 2300,1 ),  
 Sec(2, 'Введение', 3400,2 ),  
 Sec(3, 'О зачислении', 9000,3 ),  
 Sec(4, 'Глава 1', 4000,3 ),  
 Sec(5, 'Заключение', 2700,3 ),  
]  
   
# Документы  
docs = [  
 Doc(1, 'Книга'),  
 Doc(2, 'Курсовая'),  
 Doc(3, 'Приказ'),  
 Doc(4, 'Заметки'),  
 Doc(5, 'Книга1'),  
 Doc(6, 'Сборник указов '),  
]   
   
secs\_docs = [  
 SecDoc(1,1),  
 SecDoc(2,2),  
 SecDoc(3,3),  
 SecDoc(3,4),  
 SecDoc(3,5),  
 SecDoc(4,1),  
 SecDoc(5,2),  
 SecDoc(6,3),  
 SecDoc(6,4),  
 SecDoc(6,5),  
]   
  
one\_to\_many = [(s.id\_doc, s.id\_sec, d.named)  
 for d in docs  
 for s in secs\_docs  
 if s.id\_doc==d.id]  
  
many\_to\_many\_temp = [(d.named, sd.id\_doc, sd.id\_sec)  
 for d in docs  
 for sd in secs\_docs  
 if d.id == sd.id\_doc]  
  
many\_to\_many = [(s.names, s.symb, doc\_name)  
 for doc\_name, Id\_Doc, Id\_Sec in many\_to\_many\_temp  
 for s in secs if s.id == Id\_Sec]  
  
def task\_1(one\_to\_many):  
 *""" ЗАДАНИЕ №1.  
 Выведите список всех связанных разделов и документов, отсортированный по документам, сортировка по разделам произвольная  
 """* return sorted(one\_to\_many, key=itemgetter(2))  
  
def task\_2(one\_to\_many):  
 *""" ЗАДАНИЕ №2.  
 Выведите список документов с суммарным количеством символов в каждом документе, отсортированный по количеству символов.  
 """* res\_12\_unsorted = []  
   
 for d in docs:  
   
 d\_secs = list(filter(lambda i: i[2]==d.named, one\_to\_many))  
 if len(d\_secs) > 0:  
 d\_symbs = [ Symbols for \_,Symbols,\_ in d\_secs]  
   
 d\_symbs\_sum = sum(d\_symbs)  
 res\_12\_unsorted.append((d.named, d\_symbs\_sum))  
   
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)   
 return res\_12  
  
  
def task\_3(many\_to\_many):  
 *""" ЗАДАНИЕ №3.  
 Выведите список всех документов, у которых в названии присутствует слово «Книга», и список разделов в нем.  
 """* res\_13 = {}  
   
 for d in docs:  
 if 'Книга' in d.named:  
   
 d\_secs = list(filter(lambda i: i[2]==d.named, many\_to\_many))  
   
 d\_secs\_names = [x for x,\_,\_ in d\_secs]  
   
 res\_13[d.named] = d\_secs\_names  
  
 return res\_13  
  
   
def main():  
 *"""Основная функция"""* print(f'{"-" \* 10} Задание №1. {"-" \* 10}')  
 print(\*task\_1(one\_to\_many), sep='\n', end='\n\n')  
  
 print(f'{"-" \* 10} Задание №2. {"-" \* 10}')  
 print(\*task\_2(one\_to\_many), sep='\n', end='\n\n')  
  
 print(f'{"-" \* 10} Задание №3. {"-" \* 10}')  
 print(\*task\_3(many\_to\_many), sep='\n', end='\n\n')  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**tests.py**

import unittest  
from main import one\_to\_many, many\_to\_many  
from main import task\_1  
from main import task\_2  
from main import task\_3  
  
  
class TestProgramm(unittest.TestCase):  
  
 def test\_task1(self):  
 result = [  
 (4, 1, 'Заметки'),  
 (1, 1, 'Книга'),  
 (5, 2, 'Книга1'),  
 (2, 2, 'Курсовая'),  
 (3, 3, 'Приказ'),  
 (3, 4, 'Приказ'),  
 (3, 5, 'Приказ'),  
 (6, 3, 'Сборник указов '),  
 (6, 4, 'Сборник указов '),  
 (6, 5, 'Сборник указов ')  
 ]  
  
 self.assertEqual(task\_1(one\_to\_many), result)  
  
 def test\_task2(self):  
 result = [  
 ('Приказ', 12),  
 ('Сборник указов ', 12),  
 ('Курсовая', 2),  
 ('Книга1', 2),  
 ('Книга', 1),  
 ('Заметки', 1)  
 ]  
  
 self.assertEqual(task\_2(one\_to\_many), result)  
  
  
 def test\_task3(self):  
 result = {  
 'Книга': ['От автора'],   
 'Книга1': ['Введение']  
 }  
  
 self.assertEqual(task\_3(many\_to\_many), result)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Результат выполнения**

