Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждениевысшего образования «Московский государственный технический университетимени 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

```
Testing started at 1:26 ...


Ran 1 test in 0.013s

OK
Launching unittests with arguments python -m unittest tests.TestProgramm.test_task1 in C:\Users\As
```