**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по РК №2

Вариант запросов: B

Вариант предметной области: 24

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-33Б |  | преподаватель каф. ИУ5 |
| Хабленко Инна |  | Гапанюк Ю. Е. |
|  |  |  |

Москва, 2023 г.

**Вариант запросов B. Предметная область 24.**

# «Глава» и «Книга» связаны соотношением один-ко-многим. Выведите список всех глав, начинающихся с буквы «А».

# «Глава» и «Книга» связаны соотношением один-ко-многим. Выведите список книг с минимальным семестром, когда проходят темы этих глав, отсортированный по минимальному семестру.

# «Глава» и «Книга» связаны соотношением многие-ко-многим. Выведите список всех связанных глав и книг, отсортированный по главам, сортировка по книгам произвольная.

**Задание**

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).

**Листинг программы.**

**main.py**

from operator import itemgetter

class Chapter:

    """Глава"""

    def \_\_init\_\_(self, id, title, semester, book\_id):

        self.id = id

        self.title = title

        self.semester = semester

        self.book\_id = book\_id

class Book:

    """Книга"""

    def \_\_init\_\_(self, id, title, author):

        self.id = id

        self.title = title

        self.author = author

class ChapterBook:

    """

    'Главы книги' для реализации связи многие-ко-многим

    """

    def \_\_init\_\_(self, book\_id, chapter\_id):

        self.book\_id = book\_id

        self.chapter\_id = chapter\_id

# Книги

books = [

    Book(1, 'Математический анализ', 'В.А.Зорич'),

    Book(2, 'Курс математического анализа', 'Л.Д.Кудрявцев'),

    Book(3, 'Сборник задач для ВТУЗОВ', 'А.В.Ефимов, Б.П.Демидович'),

    Book(11, 'Ряды и кратные интегралы', 'А.А.Гусак'),

    Book(22, 'Сборник задач по векторному анализу', 'Е.Н.Кожевников'),

    Book(33, 'Сборник задач по курсу математического анализа', 'Г.Н.Берман'),

]

# Главы

chapters = [

    Chapter(1, 'Кратные интегралы', 3, 1),

    Chapter(2, 'Анализ векторный', 3, 2),

    Chapter(3, 'Предел функции', 1, 3),

    Chapter(4, 'Дифференциалы', 2, 3),

    Chapter(5, 'Ряды', 2, 3),

]

chapters\_books = [

    ChapterBook(1,1),

    ChapterBook(2,2),

    ChapterBook(3,3),

    ChapterBook(3,4),

    ChapterBook(3,5),

    ChapterBook(11,1),

    ChapterBook(22,2),

    ChapterBook(33,3),

    ChapterBook(33,4),

    ChapterBook(33,5),

]

# Соединение данных один-ко-многим

def one\_to\_many\_(chapters, books):

    return [(c.title, c.semester, b.title)

        for b in books

        for c in chapters

        if c.book\_id==b.id]

# Соединение данных многие-ко-многим

def many\_to\_many\_(chapters, chapters\_books, books):

    many\_to\_many\_temp = [(b.title, cb.book\_id, cb.chapter\_id)

                        for b in books

                        for cb in chapters\_books

                        if b.id==cb.book\_id]

    return [(c.title, c.semester, book\_title)

        for book\_title, book\_id, chapter\_id in many\_to\_many\_temp

        for c in chapters if c.id==chapter\_id]

def get\_result\_1(data):

    array = sorted(data, key=itemgetter(2))

    res\_11 = []

    for i in range(len(array)):

        if array[i][0].startswith("А"):

            res\_11.append(array[i])

    return res\_11

def get\_result\_2(data):

    res\_12\_unsorted = []

    i = 0

    for b in books:

        b\_chapter = list(filter(lambda i: i[2]==b.title, data))

        if len(b\_chapter) > 0:

            b\_semester = [page for \_,page,\_ in b\_chapter]

            b\_semester\_min = min(b\_semester)

            res\_12\_unsorted.append((b.title, b\_semester\_min))

    res\_12 = sorted(set(res\_12\_unsorted), key=itemgetter(1))

    return res\_12

def get\_result\_3 (data):

    res\_13 = sorted(data, key=itemgetter(0))

    return res\_13

def main():

    """Основная функция"""

    one\_to\_many = one\_to\_many\_(chapters, books)

    many\_to\_many = many\_to\_many\_(chapters, chapters\_books, books)

    print('Задание B1\n')

    res\_11 = get\_result\_1(one\_to\_many)

    for i in range(len(res\_11)):

        print(\*res\_11[i])

    print('\nЗадание B2\n')

    res\_12 = get\_result\_2(one\_to\_many)

    for i in range(len(res\_12)):

        print(\*res\_12[i])

    print('\nЗадание B3\n')

    res\_13 = get\_result\_3(many\_to\_many)

    for i in range(len(res\_13)):

        print(\*res\_13[i])

    print('\n')

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

    main()

**tests.py**

import unittest

from main import \*

class TestJoinOperations(unittest.TestCase):

    def test\_one\_to\_many(self):

        result = one\_to\_many\_(chapters, books)

        self.assertTrue(len(result) > 0)

    def test\_many\_to\_many(self):

        result = many\_to\_many\_(chapters, chapters\_books, books)

        self.assertTrue(len(result) > 0)

class TestTaskResults(unittest.TestCase):

    def test\_get\_result\_1(self):

        data = [("Кратные интегралы", 3, "Математический анализ"), ("Анализ векторный", 3, "Курс математического анализа"), ("Предел функции", 1, "Сборник задач для ВТУЗОВ")]

        result = get\_result\_1(data)

        self.assertEqual(result, [("Анализ векторный", 3, "Курс математического анализа")])

    def test\_get\_empty\_result\_1(self):

        data = []

        result = get\_result\_1(data)

        self.assertEqual(result, [])

    def test\_get\_result\_2(self):

        data = [("Кратные интегралы", 3, "Математический анализ"), ("Анализ векторный", 3, "Курс математического анализа"), ("Предел функции", 1, "Сборник задач для ВТУЗОВ")]

        result = get\_result\_2(data)

        self.assertEqual(result, [("Сборник задач для ВТУЗОВ", 1), ("Математический анализ", 3), ("Курс математического анализа", 3)])

    def test\_get\_result\_3(self):

        data = [

            ("Анализ векторный", 3, "Сборник задач по векторному анализу1"),

            ("Дифференциалы", 2, "Сборник задач для ВТУЗОВ"),

            ("Дифференциалы", 2, "Сборник задач по курсу математического анализа"),

            ("Кратные интегралы", 3, "Математический анализ"),

            ("Кратные интегралы", 3, "Ряды и кратные интегралы"),

            ("Предел функции", 1, "Сборник задач для ВТУЗОВ"),

            ("Предел функции", 1, "Сборник задач по курсу математического анализа"),

            ("Ряды", 2, "Сборник задач для ВТУЗОВ"),

            ("Ряды", 2, "Сборник задач по курсу математического анализа")]

        result = sorted(data, key=itemgetter(0))

        self.assertEqual(result, get\_result\_3(data))

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

    unittest.main()

**test\_runner.py**

import tests

import unittest

suite = unittest.TestLoader().loadTestsFromModule(tests)

results = unittest.TextTestRunner(verbosity=2).run(suite)

**Результат выполнения**

