**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 №2по дисциплине «Базовые компоненты интернет-технологий»

Вариант №15-В

Выполнил:

студент группы ИУ5-33Б

Рассказов Н.Д.

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Ю. Е.

2022 г.

Выданное задание

**Вариант В.**

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

|  |  |  |
| --- | --- | --- |
| **№ варианта** | **Класс 1** | **Класс 2** |
| 15 | Файл | Каталог файлов |

Разработка тестов

1) Провести рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создать модульные тесты с применением TDD - фреймворка (3 теста).

Текст программы main.py

*from* operator *import* itemgetter

class File:

*"""Файл"""*

    def \_\_init\_\_(*self*, *id*, *name*, *weight\_mb*, *catalog\_id*):

*self*.id = *id*

*self*.name = *name*

*self*.weight\_mb = *weight\_mb*

*self*.catalog\_id = *catalog\_id*

class Catalog:

*"""Каталог файлов"""*

    def \_\_init\_\_(*self*, *id*, *path*):

*self*.id = *id*

*self*.path = *path*

class FileCatalog:

*"""*

*'Файл в каталоге файлов' для реализации*

*связи многие-ко-многим*

*"""*

    def \_\_init\_\_(*self*, *catalog\_id*, *file\_id*):

*self*.catalog\_id = *catalog\_id*

*self*.file\_id = *file\_id*

def links(*catalogs*, *files*, *files\_catalogs*):

*"""Функция для создания связей"""*

*# Соединение данных один-ко-многим*

    one\_to\_many = [(f.name, f.weight\_mb, с.path)

*for* с *in* *catalogs*

*for* f *in* *files*

*if* f.catalog\_id == с.id]

*# Соединение данных многие-ко-многим*

    many\_to\_many\_temp = [(с.path, fc.catalog\_id, fc.file\_id)

*for* с *in* *catalogs*

*for* fc *in* *files\_catalogs*

*if* с.id == fc.catalog\_id]

    many\_to\_many = [(f.name, f.weight\_mb, catalog\_path)

*for* catalog\_path, catalog\_id, file\_id *in* many\_to\_many\_temp

*for* f *in* *files*

*if* f.id == file\_id]

*return* one\_to\_many, many\_to\_many

def exercise1(*catalogs*, *files*, *files\_catalogs*):

*"""Функция для задания 1"""*

    one\_to\_many, many\_to\_many = links(*catalogs*, *files*, *files\_catalogs*)

    res\_11 = list(filter(lambda *i*: *i*[0][0] == "A", one\_to\_many))

*return* res\_11

def exercise2(*catalogs*, *files*, *files\_catalogs*):

*"""Функция для задания 2"""*

    one\_to\_many, many\_to\_many = links(*catalogs*, *files*, *files\_catalogs*)

    res\_12\_unsorted = []

*# Перебираем все каталоги*

*for* c *in* *catalogs*:

*# Список файлов в каталоге*

        c\_files = list(filter(lambda *i*: *i*[2] == c.path, one\_to\_many))

*# Если каталог не пустой*

*if* len(c\_files) > 0:

*# Весы файлов в каталоге*

            c\_weights = [weight\_mb *for* \_, weight\_mb, \_ *in* c\_files]

*# Минимальный вес файла в каталоге*

            c\_weights\_min = min(c\_weights)

            res\_12\_unsorted.append((c.path, c\_weights\_min))

*# Сортировка по минимальному весу файла*

    res\_12 = sorted(res\_12\_unsorted, *key*=itemgetter(1), *reverse*=True)

*return* res\_12

def exercise3(*catalogs*, *files*, *files\_catalogs*):

*"""Функция для задания 3"""*

    one\_to\_many, many\_to\_many = links(*catalogs*, *files*, *files\_catalogs*)

    res\_13 = {}

*# Перебираем все каталоги*

*for* c *in* *catalogs*:

*# Список файлов католога*

        c\_files = list(filter(lambda *i*: *i*[2] == c.path, many\_to\_many))

*# Только имена файлов*

        c\_files\_names = [x *for* x, \_, \_ *in* c\_files]

*# Добавляем результат в словарь*

        res\_13[c.path] = sorted(c\_files\_names)

*# Сортируем словарь по файлам*

    res\_13 = sorted(res\_13.items(), *key*=itemgetter(1))

*return* res\_13

def main():

*"""Основная функция"""*

*# Каталоги файлов*

    catalogs = [

        Catalog(1, 'C:\\'),

        Catalog(2, 'C:\\Users\\'),

        Catalog(3, 'C:\\Users\\User\\'),

        Catalog(11, 'D:\\'),

        Catalog(22, 'D:\\Users\\'),

        Catalog(33, 'D:\\Users\\User\\'),

    ]

*# Файлы*

    files = [

        File(1, 'AllGuys.mp4', 14.9, 1),

        File(2, 'Valorant.mp4', 1699.84, 2),

        File(3, 'Atement.docx', 0.013, 3),

        File(4, 'Statement.pdf', 0.013, 3),

        File(5, 'Programm.zip', 100, 3),

    ]

*# Файлы в каталогах файлов*

    files\_catalogs = [

        FileCatalog(1, 1),

        FileCatalog(2, 2),

        FileCatalog(3, 3),

        FileCatalog(3, 4),

        FileCatalog(3, 5),

        FileCatalog(11, 1),

        FileCatalog(22, 2),

        FileCatalog(33, 3),

        FileCatalog(33, 4),

        FileCatalog(33, 5),

    ]

    print('Задание В1')

    print(exercise1(catalogs, files, files\_catalogs))

    print('\nЗадание В2')

    print(exercise2(catalogs, files, files\_catalogs))

    print('\nЗадание В3')

    print(exercise3(catalogs, files, files\_catalogs))

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

    main()

Текст программы tests.py

*from* main *import* \*

class TestRK:

*# Каталоги файлов*

    catalogs = [

        Catalog(1, 'C:\\'),

        Catalog(2, 'C:\\Users\\'),

        Catalog(3, 'C:\\Users\\User\\'),

        Catalog(11, 'D:\\'),

        Catalog(22, 'D:\\Users\\'),

        Catalog(33, 'D:\\Users\\User\\'),

    ]

*# Файлы*

    files = [

        File(1, 'AllGuys.mp4', 14.9, 1),

        File(2, 'Valorant.mp4', 1699.84, 2),

        File(3, 'Atement.docx', 0.013, 3),

        File(4, 'Statement.pdf', 0.013, 3),

        File(5, 'Programm.zip', 100, 3),

    ]

*# Файлы в каталогах файлов*

    files\_catalogs = [

        FileCatalog(1, 1),

        FileCatalog(2, 2),

        FileCatalog(3, 3),

        FileCatalog(3, 4),

        FileCatalog(3, 5),

        FileCatalog(11, 1),

        FileCatalog(22, 2),

        FileCatalog(33, 3),

        FileCatalog(33, 4),

        FileCatalog(33, 5),

    ]

    def test\_exercise1(*self*):

*assert* exercise1(TestRK.catalogs, TestRK.files, TestRK.files\_catalogs) == [

            ('AllGuys.mp4', 14.9, 'C:\\'), ('Atement.docx', 0.013, 'C:\\Users\\User\\')]

    def test\_exercise2(*self*):

*assert* exercise2(TestRK.catalogs, TestRK.files, TestRK.files\_catalogs) == [

            ('C:\\Users\\', 1699.84), ('C:\\', 14.9), ('C:\\Users\\User\\', 0.013)]

    def test\_exercise3(*self*):

*assert* exercise3(TestRK.catalogs, TestRK.files, TestRK.files\_catalogs) == [

            ('C:\\', ['AllGuys.mp4']), ('D:\\', ['AllGuys.mp4']), ('C:\\Users\\User\\', [

            'Atement.docx', 'Programm.zip', 'Statement.pdf']),

            ('D:\\Users\\User\\', ['Atement.docx', 'Programm.zip', 'Statement.pdf']),

            ('C:\\Users\\', ['Valorant.mp4']), ('D:\\Users\\', ['Valorant.mp4'])]

Результаты выполнения

======================= test session starts =======================

platform win32 -- Python 3.10.5, pytest-7.2.0, pluggy-1.0.0

rootdir: C:\Users\Никита\Documents\GitHub\BKIT-3Term\RK2

plugins: anyio-3.6.2, bdd-6.1.1

collected 3 items

tests.py ... [100%]

=======================3 passed in 0.02s =======================

**Скриншот с результатами выполнения**

