ИУ5-32Б
Носков А. А.

Рубежный контроль № 2

Условие:

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).

Текст программы

main.py

```
from operator import itemgetter

# Файл - Каталог файлов

class File:
    def __init__(self, file_id, name, type, size, catalog_id):
        self.id = file_id
        self.name = name
        self.type = type
        self.size = size
        self.catalog_id = catalog_id

class Catalog:
    def __init__(self, catalog_id, name):
        self.id = catalog_id
        self.name = name

class CatalogFile:
    def __init__(self, file_id, catalog_id):
        self.file_id = file_id
        self.catalog_id = catalog_id

files = [
    File(1, "some_text", "txt", 5, 1),
```

```
    File(2, "more_text", "txt", 6, 1),
    File(3, "first", "py", 3, 2),
    File(4, "second", "py", 3, 2),
    File(5, "third", "py", 5, 2)
]

catalogs = [
    Catalog(1, "Texts"),
    Catalog(2, "PythonCode")
]

catalog_files = [
    CatalogFile(1, 2),
    CatalogFile(2, 1),
    CatalogFile(3, 2),
    CatalogFile(4, 2),
    CatalogFile(5, 2),
    CatalogFile(1, 1)
]

def first_task(lst):
    return sorted(lst, key=lambda x: x[0])

def second_task(lst):
    d = {}
    for file, file_id, catalog in lst:
        if catalog in d:
            d[catalog] += 1
        else:
            d[catalog] = 1
    return sorted(d.items(), key=lambda x: x[1], reverse=True)

def third_task(lst, end):
    return [(file_name, catalog_name) for file_name, _, catalog_name in lst
if file_name.endswith(end)]

def main():
    one_to_many = [(file.name + "." + file.type, file.id, catalog.name)
                   for catalog in catalogs
```

```
                for file in files
                if file.catalog_id == catalog.id]

    many_to_many = [(file.name + '.' + file.type, catalog_file.file_id,
catalog.name)
                for file in files
                for catalog in catalogs
                for catalog_file in catalog_files
                if file.id == catalog_file.file_id and catalog.id ==
catalog_file.catalog_id]
    print(many_to_many)

    print("\nЗадание 1")
    print(first_task(one_to_many))

    print("\nЗадание 2")
    print(second_task(one_to_many))

    print("\nЗадание 3")
    print(third_task(many_to_many, "txt"))

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

tests.py

```
import main
import unittest


class Tests(unittest.TestCase):
    def first_test(self):
        test_data = [("test1", "test1", "test1"), ("test2", "test2", "test2")]
        result = main.first_task(test_data)
        good_result = sorted(test_data, key=lambda x: x[0])
        self.assertEqual(result, good_result)

    def second_test(self):
        test_data = [('some_text.txt', 1, 'Texts'), ('more_text.txt', 2, 'Texts'),
('first.py', 3, 'PythonCode'), ('second.py', 4, 'PythonCode'), ('third.py', 5,
'PythonCode')]
```

```
        result = main.second_task(test_data)
        good_result = [('PythonCode', 3), ('Texts', 2)]
        self.assertEqual(result, good_result)

    def third_test(self):
        test_data = [('some_text.txt', 1, 'Texts'), ('some_text.txt', 1,
'PythonCode'), ('more_text.txt', 2, 'Texts'), ('first.py', 3, 'PythonCode'),
('second.py', 4, 'PythonCode'), ('third.py', 5, 'PythonCode')]
        result = main.third_task(test_data, "txt")
        good_result = [('some_text.txt', 'Texts'), ('some_text.txt',
'PythonCode'), ('more_text.txt', 'Texts')]
        self.assertEqual(result, good_result)

t = Tests()

t.first_test()
t.second_test()
t.third_test()
```