**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»**

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю №2

Вариант №В-15

Выполнила:

студентка группы ИУ5-33Б

Румак Д.П.

Проверил:

Гапанюк Ю.Е.

2021 г.

**Задание**

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**

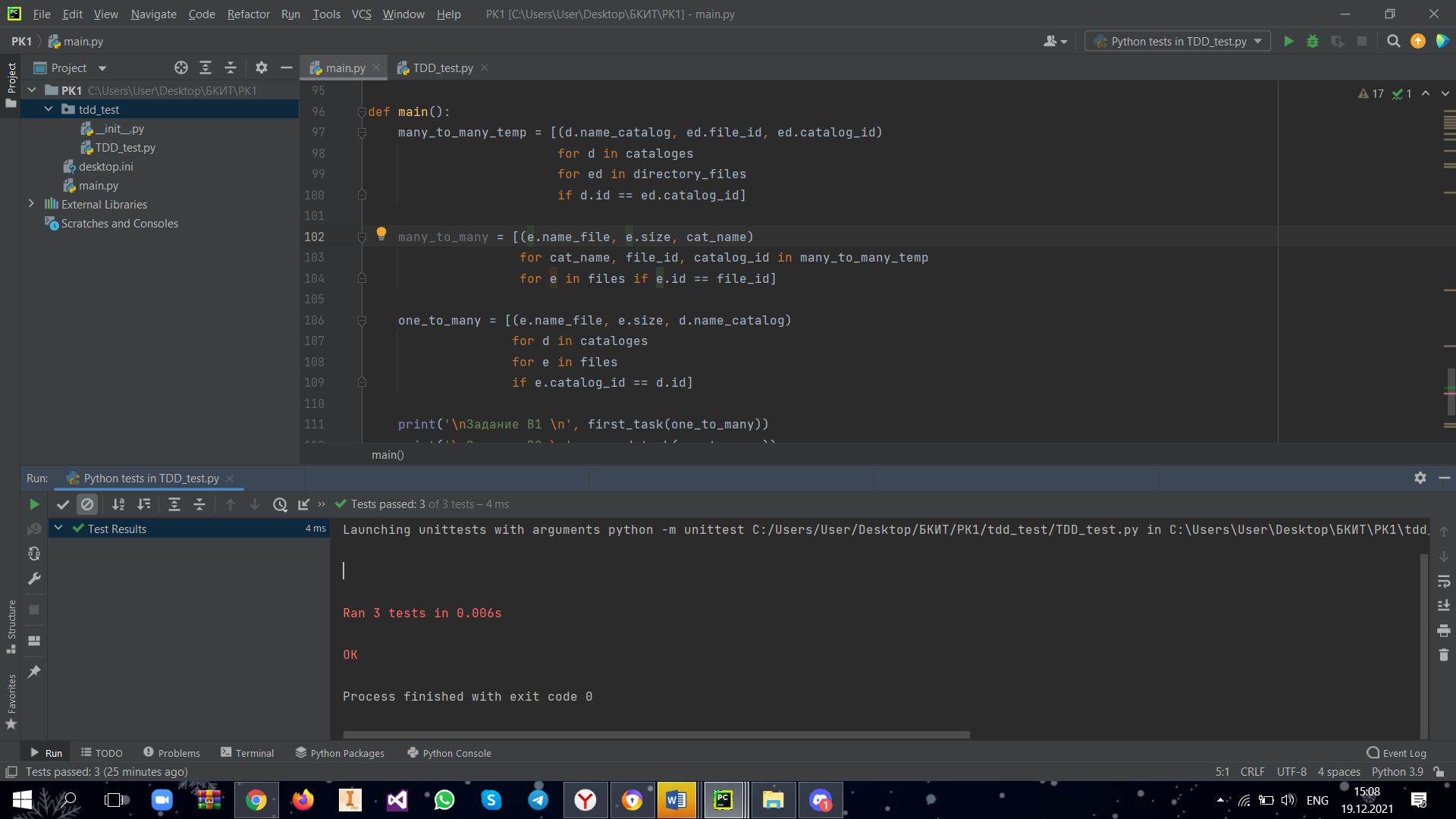
**main.py**

from operator import itemgetter  
  
class File:  
 *"""Файл"""* def \_\_init\_\_(self, id, name\_file, catalog\_id, size):  
 self.id = id  
 self.name\_file = name\_file  
 self.catalog\_id = catalog\_id  
 self.size = size  
  
class Catalog:  
 *"""Каталог"""* def \_\_init\_\_(self, id, name\_catalog):  
 self.id = id  
 self.name\_catalog = name\_catalog  
  
class DirectoryFiles:  
 *"""Файлы каталога для связи М:М"""* def \_\_init\_\_(self, catalog\_id, file\_id):  
 self.catalog\_id = catalog\_id  
 self.file\_id = file\_id  
#Файлы  
files = [  
 File(1, 'photo.pdf', 2, 4.5),  
 File(2, 'image.pdf', 3, 2.1),  
 File(3, 'image.jpg', 4, 3.25),  
 File(4, 'image0.jpg', 4, 1.7),  
 File(5, 'my\_work.pdf', 1, 1.3),  
 File(6, 'dz1\_oop.jpg', 3, 2.4)  
]  
  
#Каталоги  
cataloges = [  
 Catalog(1, 'Рабочий стол'),  
 Catalog(2, 'Панель управления'),  
 Catalog(3, 'Папка1'),  
 Catalog(4, 'Папка2'),  
 Catalog(5, 'Работы'),  
 Catalog(6, 'Домашнее задание'),  
]  
  
directory\_files = [  
 DirectoryFiles(2, 1),  
 DirectoryFiles(3, 2),  
 DirectoryFiles(4, 3),  
 DirectoryFiles(4, 4),  
 DirectoryFiles(1, 5),  
 DirectoryFiles(3, 6)  
]  
  
  
def first\_task(one\_to\_many):  
 *"""  
 "Файл" и "Каталог" связаны соотношением один-ко-многим.   
 Вывести список всех файлов, у которых название начинается на "i", и названия их каталогов.  
 """* res\_1 = []  
 for name\_file, size, name\_catalog in one\_to\_many:  
 if name\_file[0] == 'i':  
 res\_1.append((name\_file, name\_catalog))  
 return res\_1  
  
  
def second\_task(one\_to\_many):  
 *"""  
 "Файл" и "Каталог" связаны соотношением один-ко-многим.   
 Вывести список каталогов с минимальным размером файлов в каждом каталоге, отстортированный по минимальному размеру.  
 P.S. размер файлов указан в Мб.  
 """* res\_21 = []  
 # Список каталогов  
 for d in cataloges:  
 d\_files = list(filter(lambda i: i[2] == d.name\_catalog, one\_to\_many))  
 # Если каталог не пустой  
 if len(d\_files) > 0:  
 # Размеры файлов  
 d\_sizes = [size for \_, size, \_ in d\_files]  
 min\_size = min(d\_sizes)  
 res\_21.append((d.name\_catalog, min\_size))  
 res\_2 = sorted(res\_21, key=itemgetter(1))  
 return res\_2  
  
  
def third\_task(many\_to\_many):  
 *"""  
 "Файл" и "Каталог" связаны соотношением многие-ко-многим.   
 Вывести список всех связанных каталогов и файлов, отстортированный по файлам, сортровка по каталогам произвольная.  
 """* res\_31 = []  
 for name\_file, size, name\_catalog in many\_to\_many:  
 res\_31.append((name\_file, name\_catalog))  
 res\_3 = list(sorted(res\_31, key=itemgetter(0)))  
 return res\_3  
  
  
def main():  
 many\_to\_many\_temp = [(d.name\_catalog, ed.file\_id, ed.catalog\_id)  
 for d in cataloges  
 for ed in directory\_files  
 if d.id == ed.catalog\_id]  
  
 many\_to\_many = [(e.name\_file, e.size, cat\_name)  
 for cat\_name, file\_id, catalog\_id in many\_to\_many\_temp  
 for e in files if e.id == file\_id]  
  
 one\_to\_many = [(e.name\_file, e.size, d.name\_catalog)  
 for d in cataloges  
 for e in files  
 if e.catalog\_id == d.id]  
  
 print('\nЗадание B1 \n', first\_task(one\_to\_many))  
 print('\nЗадание B2 \n', second\_task(one\_to\_many))  
 print('\nЗадание B3 \n', third\_task(one\_to\_many))  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**TDD\_test.py**

import unittest  
import sys, os  
  
sys.path.append(os.getcwd())  
from main import \*  
  
class TddTest(unittest.TestCase):  
 def test\_1(self):  
 one\_to\_many = [(e.name\_file, e.size, d.name\_catalog)  
 for d in cataloges  
 for e in files  
 if e.catalog\_id == d.id]  
 self.assertEqual(first\_task(one\_to\_many), [('image.pdf', 'Папка1'), ('image.jpg', 'Папка2'), ('image0.jpg', 'Папка2')])  
  
 def test\_2(self):  
 one\_to\_many = [(e.name\_file, e.size, d.name\_catalog)  
 for d in cataloges  
 for e in files  
 if e.catalog\_id == d.id]  
 self.assertEqual(second\_task(one\_to\_many), [('Рабочий стол', 1.3), ('Папка2', 1.7), ('Папка1', 2.1), ('Панель управления', 4.5)])  
  
 def test\_3(self):  
 many\_to\_many\_temp = [(d.name\_catalog, ed.file\_id, ed.catalog\_id)  
 for d in cataloges  
 for ed in directory\_files  
 if d.id == ed.catalog\_id]  
  
 many\_to\_many = [(e.name\_file, e.size, cat\_name)  
 for cat\_name, file\_id, catalog\_id in many\_to\_many\_temp  
 for e in files if e.id == file\_id]  
  
 self.assertEqual(third\_task(many\_to\_many),  
 [('dz1\_oop.jpg', 'Папка1'), ('image.jpg', 'Папка2'), ('image.pdf', 'Папка1'), ('image0.jpg', 'Папка2'),  
 ('my\_work.pdf', 'Рабочий стол'), ('photo.pdf', 'Панель управления')])  
  
  
if \_\_name\_\_=='\_\_main\_\_':  
 unittest.main()

**Результаты выполнения программы**

****