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 «Головной учебно-исследовательский и методический центр профессиональной реабилитации лиц с ограниченными возможностями здоровья (инвалидов)»
КАФЕДРА «Системы обработки информации и управления» (ИУ5)

Отчет по рубежному контролю №2 по курсу «Парадигмы и конструкции языков программирования»

Вариант предметной области: 31
Вариант запросов: А

Студент ИУ5Ц-51Б (группа) _____ (подпись, дата) Р.Р. Чупанова (И.О. Фамилия)

Преподаватель _____ (подпись, дата) А.Н. Нардид (И.О. Фамилия)

2024 г.

Задача

1) Рефакторинг текста программы рубежного контроля №1 для модульного тестирования;
2) Для текста программы рубежного контроля №1 создать модульные тесты с использованием TDD-фреймворка.

Код решения

```
import unittest
from collections import defaultdict

class Table:
    def __init__(self, table_id, name, row_count, database_id):
        self.table_id = table_id
        self.name = name
        self.row_count = row_count
        self.database_id = database_id

class Database:
    def __init__(self, database_id, name):
        self.database_id = database_id
        self.name = name

# «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по отделам, сортировка по сотрудникам произвольная.
# "База данных" и "Таблица данных" связаны соотношением один-ко-многим. Выведен список всех связанных таблиц и баз данных, отсортированный по Таблицам, сортировка по Базам произвольная
def sort(tables, databases):
    sorted_data = sorted(tables, key=lambda x: x.database_id)
    return [
        (database.name, table.name)
        for database in databases
        for table in sorted_data
        if table.database_id == database.database_id
    ]

# «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с суммарной зарплатой сотрудников в каждом отделе, отсортированный по суммарной зарплате.
# Сортировка по числу записей в таблице данных
def counts(tables, databases):
    database_row_count = defaultdict(int)     #  defaultdict для суммирования строк по идентификаторам баз данных
    for table in tables:
        database_row_count[table.database_id] += table.row_count

    # Создаем список кортежей (название базы данных, общее количество строк)
    return sorted(
        [ (database.name, database_row_count[database.database_id]) for database in databases ],
        key=lambda x: x[1],  # Сортируем по количеству строк
        reverse=True  # В порядке убывания
    )

# «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых в названии присутствует слово «отдел», и список работающих в них сотрудников.
# Вывод всех баз данных, в названии которых есть слово Database, и список таблиц в них
def keyword(databases, tables, keyword):
```

```
    return [
        ( database.name, [table.name for table in tables if table.database_id
== database.database_id] ) for database in databases if keyword in
database.name
    ]

databases = [
    Database(database_id=1, name="Legal pluralism of the North Caucasus"),
    Database(database_id=2, name="Tradition"),
    Database(database_id=3, name="Language"),
    Database(database_id=4, name="Database Auls")
]

tables = [
    Table(table_id=1, name="AdatLaw", row_count=636, database_id=1),
    Table(table_id=2, name="Darkquante languages", row_count=6543,
database_id=3),
    Table(table_id=3, name="Interclass relations", row_count=2345,
database_id=2),
    Table(table_id=4, name="Leki language family", row_count=765,
database_id=3),
    Table(table_id=5, name="Feud", row_count=200, database_id=2),
    Table(table_id=6, name="Kurah", row_count=642, database_id=4),
    Table(table_id=7, name="Qala-Qouraish", row_count=200, database_id=4),
]

class test(unittest.TestCase):
    def setUp(self): # подготовка данныъ
        self.databases = [
            Database(database_id=1, name="Database Alpha"),
            Database(database_id=2, name="Beta"),
            Database(database_id=3, name="Database Gamma"),
        ]

        self.tables = [
            Table(table_id=1, name="Table 1", row_count=100, database_id=1),
            Table(table_id=2, name="Table 2", row_count=200, database_id=1),
            Table(table_id=3, name="Table 3", row_count=300, database_id=2),
            Table(table_id=4, name="Table 4", row_count=400, database_id=3),
        ]

    def testSort(self):
        expected = [
            ("Database Alpha", "Table 1"),
            ("Database Alpha", "Table 2"),
            ("Beta", "Table 3"),
            ("Database Gamma", "Table 4"),
        ]
        result = sort(self.tables, self.databases)
        self.assertEqual(result, expected)

    def testString(self):
        expected = [
            ("Database Gamma", 400),
            ("Database Alpha", 300),
            ("Beta", 300),
        ]
        result = counts(self.tables, self.databases)
        self.assertEqual(result, expected)

    def testKeyword(self):
        expected = [
            ("Database Alpha", ["Table 1", "Table 2"]),  # Ключ "Database"
            ("Database Gamma", ["Table 4"]),
```

```
        ]
        result = keyword(self.databases, self.tables, "Database")
        self.assertEqual(result, expected)


if __name__ == "__main__":
    unittest.main(exit=False)
```

Результат