МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю № 2

Вариант № 31

Вариант запросов В

Выполнил:
Студент группы ИУ5-34Б
Солижонов У.И.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Условия рубежного контроля №2 по курсу ПиК ЯП

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).

Задание варианта

Классы: «Синтаксическая конструкция» и «Язык программирования»

Вариант В.

1. «Синтаксическая конструкция» и «Язык программирования» связаны соотношением один-ко-многим. Выведите список всех конструкций, у которых название начинается с буквы «А», и названия их языков программирования.

2. «Синтаксическая конструкция» и «Язык программирования» связаны соотношением один-ко-многим. Выведите список языков с минимальным временем компиляции конструкции в каждом языке программирования, отсортированный по минимальному времени компиляции.

3. «Синтаксическая конструкция» и «Язык программирования» связаны соотношением многие-ко-многим. Выведите список всех связанных конструкций и языков, отсортированный по конструкциям, сортировка по языкам программирования произвольная.

Листинг Программы

Main.py

```
from operator import itemgetter

class Table:
    def __init__(self, id, name, row_count, db_id):
        self.id = id
        self.name = name
        self.row_count = row_count
        self.db_id = db_id

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

class TableDatabase:
    def __init__(self, db_id, table_id):
        self.db_id = db_id
```

```
        self.table_id = table_id

# --- Логические функции для тестирования ---

def task_b1(tables, dbs):
    """Таблицы на 'А' и их БД (1-ко-многим)"""
    one_to_many = [(t, d) for d in dbs for t in tables if t.db_id == d.id]
    res = [(t.name, d.name) for t, d in one_to_many if t.name.startswith('A')]
    return sorted(res, key=itemgetter(0))

def task_b2(tables, dbs):
    """БД и минимальное кол-во строк в их таблицах"""
    res = []
    for d in dbs:
        d_tables = [t.row_count for t in tables if t.db_id == d.id]
        if d_tables:
            res.append((d.name, min(d_tables)))
    return sorted(res, key=itemgetter(1))

def task_b3(tables, dbs, tables_dbs):
    """Связь многие-ко-многим, сортировка по таблицам"""
    many_to_many = []
    for td in tables_dbs:
        table = next((t for t in tables if t.id == td.table_id), None)
        db = next((d for d in dbs if d.id == td.db_id), None)
        if table and db:
            many_to_many.append((table.name, db.name))
    return sorted(many_to_many, key=itemgetter(0))

# --- Данные и запуск ---

dbs = [
    Database(1, 'AnalyticsDB'),
    Database(2, 'ArchiveDB'),
    Database(3, 'AccountingDB'),
]

tables = [
    Table(1, 'Анализ_продаж', 15000, 1),
    Table(2, 'Архив_данных', 8000, 2),
    Table(3, 'Активы', 12000, 3),
    Table(4, 'Расходы', 9500, 3),
    Table(5, 'Отчеты', 7000, 2),
]

tables_dbs = [
    TableDatabase(1, 1),
    TableDatabase(2, 2),
```

```
    TableDatabase(3, 3),
    TableDatabase(3, 4),
    TableDatabase(2, 5),
]

if __name__ == '__main__':
    print('ЗАДАНИЕ B1:', task_b1(tables, dbs))
    print('ЗАДАНИЕ B2:', task_b2(tables, dbs))
    print('ЗАДАНИЕ B3:', task_b3(tables, dbs, tables_dbs))
```

test_main.py

```
import unittest

from main import Table, Database, TableDatabase, task_b1, task_b2, task_b3


class TestDatabaseLogic(unittest.TestCase):


    def setUp(self):
        """Подготовка тестовых данных"""
        self.dbs = [
            Database(1, 'DB1'),
            Database(2, 'DB2')
        ]
        self.tables = [
            Table(1, 'А_Таблица', 100, 1),
            Table(2, 'Б_Таблица', 200, 1),
            Table(3, 'А_Логи', 50, 2)
        ]
        self.links = [
            TableDatabase(1, 1),
            TableDatabase(2, 3)
        ]
```

```
    def test_task_b1(self):
        """Тест фильтрации таблиц на букву 'А'"""
        result = task_b1(self.tables, self.dbs)
        expected = [('А_Логи', 'DB2'), ('А_Таблица', 'DB1')]
        # Сортировка в b1 идет по имени таблицы
        self.assertEqual(result, expected)

    def test_task_b2(self):
        """Тест поиска минимального количества строк"""
        result = task_b2(self.tables, self.dbs)
        # DB1: min(100, 200) = 100
        # DB2: min(50) = 50
        # Сортировка по значению: (DB2, 50), (DB1, 100)
        expected = [('DB2', 50), ('DB1', 100)]
        self.assertEqual(result, expected)

    def test_task_b3(self):
        """Тест связи многие-ко-многим"""
        result = task_b3(self.tables, self.dbs, self.links)
        expected = [('А_Логи', 'DB2'), ('А_Таблица', 'DB1')]
        self.assertEqual(result, expected)

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

Запуск программы