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 Г31

Выполнил:
Студент группы ИУ5-36Б
Галенко А.В.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

Листинг кода

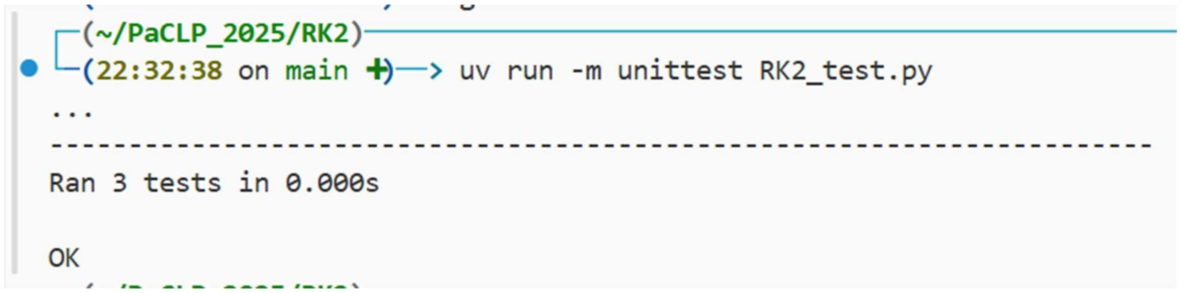
RK2.py

```
from collections import defaultdict

class Database:
    def __init__(self, databaseId, databaseName):
        self.databaseId = databaseId
        self.databaseName = databaseName
    def __eq__(self, other):
        return self.databaseId == other.databaseId and
self.databaseName == other.databaseName
    def __hash__(self):
        return hash((self.databaseId,
self.databaseName))

class DataTable:
    def __init__(self, tableId, tableName, rowCount,
databaseId):
        self.tableId = tableId
        self.tableName = tableName
        self.rowCount = rowCount
        self.databaseId = databaseId

class TableDatabaseLink:
    def __init__(self, tableId, databaseId):
        self.tableId = tableId
        self.databaseId = databaseId

def get_dbs_starting_with_a(databases, data_tables):
    result = []
    dbs_with_a = [db for db in databases if
db.databaseName.startswith('A')]

    for db in dbs_with_a:
        related_tables = [tbl for tbl in data_tables if
tbl.databaseId == db.databaseId]
        result.append((db.databaseName, [(tbl.tableName,
tbl.rowCount) for tbl in related_tables]))

    return result

def get_dbs_sorted_by_max_rows(databases, data_tables):
    tables_by_database = defaultdict(list)
    for tbl in data_tables:
        tables_by_database[tbl.databaseId].append(tbl)

    db_max_rows = []
    for db_id, tables in tables_by_database.items():
        if tables:
            max_rows = max(tbl.rowCount for tbl in
tables)
            db_name = next((db.databaseName for db in
databases if db.databaseId == db_id), None)
            if db_name:
                db_max_rows.append((db_name, max_rows))

    db_max_rows.sort(key=lambda x: x[1], reverse=True)
    return db_max_rows

def get_many_to_many_links_sorted(databases,
data_tables, links):
    db_dict = {db.databaseId: db for db in databases}
    tbl_dict = {tbl.tableId: tbl for tbl in data_tables}

    links_by_database = defaultdict(list)

    for link in links:
        db = db_dict.get(link.databaseId)
        tbl = tbl_dict.get(link.tableId)
        if db and tbl:
            links_by_database[db].append(tbl)

    sorted_dbs = sorted(links_by_database.keys(),
key=lambda d: d.databaseName)

    result = []
    for db in sorted_dbs:
        sorted_tables = sorted(links_by_database[db],
key=lambda t: t.tableName)
        result.append((db.databaseName, [(tbl.tableName,
tbl.rowCount) for tbl in sorted_tables]))

    return result
def main():
    databases = [
        Database(1, "Аналитика"), Database(2, "Архив"),
Database(3, "Бухгалтерия"),
        Database(4, "Аудит"), Database(5, "Тестовая"),
    ]
    data_tables = [
        DataTable(1, "Users", 500, 1), DataTable(2,
"Logs", 3500, 2), DataTable(3, "Invoices", 200, 3),
        DataTable(4, "AuditTrail", 1000, 4),
DataTable(5, "TempData", 150, 5), DataTable(6,
"AnalyticsData", 10000, 1),
    ]
    table_database_links = [
        TableDatabaseLink(1, 1), TableDatabaseLink(2,
2), TableDatabaseLink(3, 3),
        TableDatabaseLink(4, 4), TableDatabaseLink(5,
5), TableDatabaseLink(6, 1),
        TableDatabaseLink(2, 1), TableDatabaseLink(3,
4),
    ]
    print("\nБазы данных с именем на 'А' (один-ко-
многим):")
    dbs_with_a = get_dbs_starting_with_a(databases,
data_tables)
    for db_name, tables in dbs_with_a:
        print(f"\nБаза данных: {db_name}")
        if tables:
            for tbl_name, row_count in tables:
                print(f"  - Таблица: {tbl_name}
({row_count} строк)")
        else:
            print("  Нет связанных таблиц")

    print("\nБазы данных по максимальному числу строк в
таблицах:")
    sorted_dbs = get_dbs_sorted_by_max_rows(databases,
data_tables)
    for db_name, max_rows in sorted_dbs:
        print(f"{db_name}: макс. число строк =
{max_rows}")

    print("\nСвязанные таблицы и базы (многие-ко-
многим):")
    many_to_many_links =
get_many_to_many_links_sorted(databases, data_tables,
table_database_links)
    for db_name, tables in many_to_many_links:
        print(f"\nБаза данных: {db_name}")
        for tbl_name, row_count in tables:
            print(f"  - Таблица: {tbl_name} ({row_count}
строк)")

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

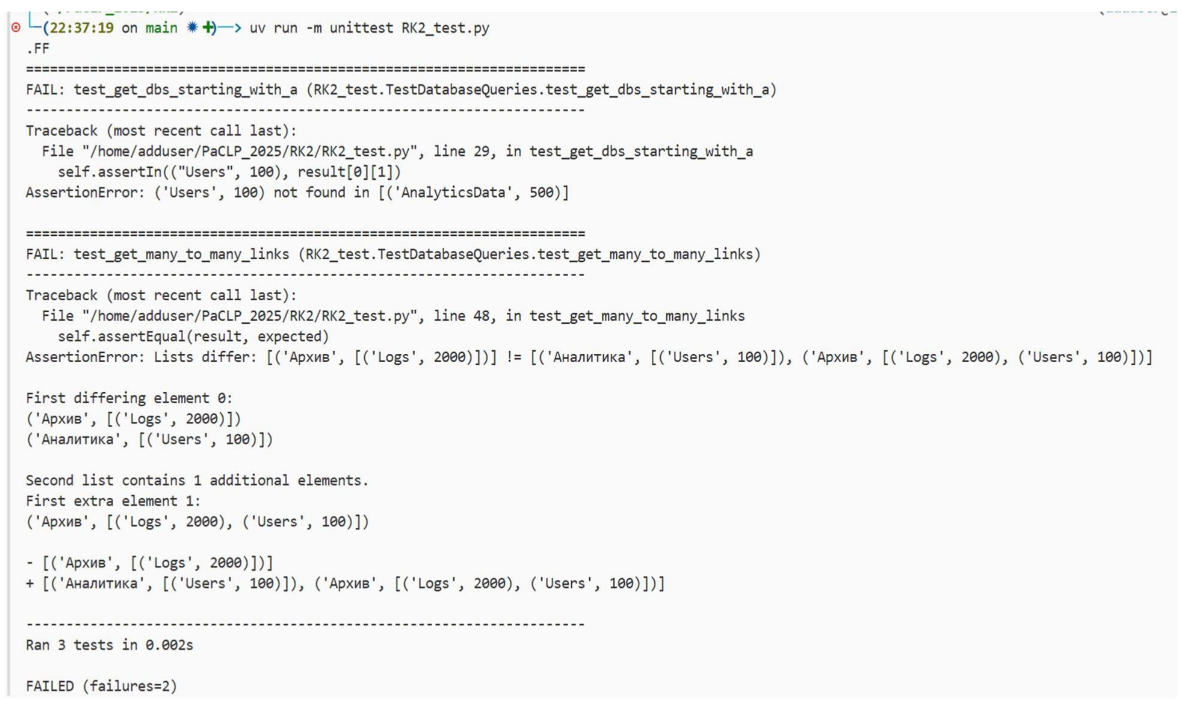
RK2_test.py

```
import unittest
from RK2 import Database, DataTable, TableDatabaseLink,
\
    get_dbs_starting_with_a, get_dbs_sorted_by_max_rows,
get_many_to_many_links_sorted

class TestDatabaseQueries(unittest.TestCase):
    def setUp(self):
        self.databases = [
            Database(1, "Аналитика"),
            Database(2, "Архив"),
            Database(3, "Продажи"),
        ]
        self.tables = [
            DataTable(1, "Users", 100, 1),
            DataTable(2, "Logs", 2000, 2),
            DataTable(3, "AnalyticsData", 500, 1),
            DataTable(4, "OldRecords", 50, 2)
        ]
        self.links = [
            TableDatabaseLink(1, 1),
            TableDatabaseLink(2, 2),
            TableDatabaseLink(1, 2)
        ]

    def test_get_dbs_starting_with_a(self):
        result = get_dbs_starting_with_a(self.databases,
self.tables)

        self.assertEqual(len(result), 2)

        self.assertEqual(result[0][0], "Аналитика")
        self.assertIn(("Users", 100), result[0][1])
        self.assertIn(("AnalyticsData", 500),
result[0][1])

    def test_get_dbs_sorted_by_max_rows(self):
        result =
get_dbs_sorted_by_max_rows(self.databases, self.tables)

        expected = [
            ("Архив", 2000),
            ("Аналитика", 500),
        ]
        self.assertEqual(result, expected)

    def test_get_many_to_many_links(self):
        result =
get_many_to_many_links_sorted(self.databases,
self.tables, self.links)

        expected = [
            ('Аналитика', [('Users', 100)]),
            ('Архив', [('Logs', 2000), ('Users', 100)]),
        ]
        self.assertEqual(result, expected)

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

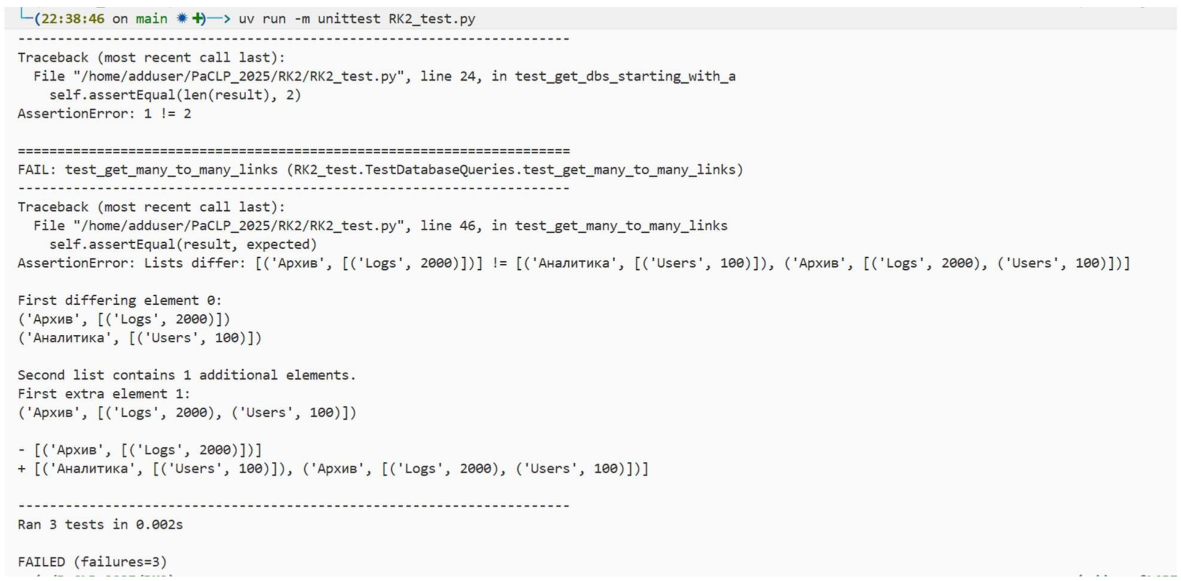
Скриншоты работы программы



```
┌(~/PaCLP_2025/RK2)──────────────────────────────────────────────
└(22:32:38 on main ✚)──> uv run -m unittest RK2_test.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```



```
└(22:37:19 on main ✱✚)──> uv run -m unittest RK2_test.py
.FF
======================================================================
FAIL: test_get_dbs_starting_with_a (RK2_test.TestDatabaseQueries.test_get_dbs_starting_with_a)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/adduser/PaCLP_2025/RK2/RK2_test.py", line 29, in test_get_dbs_starting_with_a
    self.assertIn(("Users", 100), result[0][1])
AssertionError: ('Users', 100) not found in [('AnalyticsData', 500)]

======================================================================
FAIL: test_get_many_to_many_links (RK2_test.TestDatabaseQueries.test_get_many_to_many_links)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/adduser/PaCLP_2025/RK2/RK2_test.py", line 48, in test_get_many_to_many_links
    self.assertEqual(result, expected)
AssertionError: Lists differ: [('Архив', [('Logs', 2000)])] != [('Аналитика', [('Users', 100)]), ('Архив', [('Logs', 2000), ('Users', 100)])]

First differing element 0:
('Архив', [('Logs', 2000)])
('Аналитика', [('Users', 100)])

Second list contains 1 additional elements.
First extra element 1:
('Архив', [('Logs', 2000), ('Users', 100)])

- [('Архив', [('Logs', 2000)])]
+ [('Аналитика', [('Users', 100)]), ('Архив', [('Logs', 2000), ('Users', 100)])]

----------------------------------------------------------------------
Ran 3 tests in 0.002s

FAILED (failures=2)
```



```
└(22:38:46 on main ✱✚)──> uv run -m unittest RK2_test.py
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/adduser/PaCLP_2025/RK2/RK2_test.py", line 24, in test_get_dbs_starting_with_a
    self.assertEqual(len(result), 2)
AssertionError: 1 != 2

======================================================================
FAIL: test_get_many_to_many_links (RK2_test.TestDatabaseQueries.test_get_many_to_many_links)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/adduser/PaCLP_2025/RK2/RK2_test.py", line 46, in test_get_many_to_many_links
    self.assertEqual(result, expected)
AssertionError: Lists differ: [('Архив', [('Logs', 2000)])] != [('Аналитика', [('Users', 100)]), ('Архив', [('Logs', 2000), ('Users', 100)])]

First differing element 0:
('Архив', [('Logs', 2000)])
('Аналитика', [('Users', 100)])

Second list contains 1 additional elements.
First extra element 1:
('Архив', [('Logs', 2000), ('Users', 100)])

- [('Архив', [('Logs', 2000)])]
+ [('Аналитика', [('Users', 100)]), ('Архив', [('Logs', 2000), ('Users', 100)])]

----------------------------------------------------------------------
Ran 3 tests in 0.002s

FAILED (failures=3)
```