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

Выполнил:
Студент группы ИУ5-36Б
Гаврилик Я. С.
Преподаватель:
Гапанюк Ю. Е.

Москва 2025

Листинг кода:
rk2.py:

```
from operator import itemgetter


class Row:
    def __init__(self, id, name, value, table_id):
        self.id = id
        self.name = name
        self.value = value
        self.table_id = table_id


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


class RowTable:
    def __init__(self, table_id, row_id):
        self.table_id = table_id
        self.row_id = row_id


def create_test_data():
    tables = [
        Table(1, 'Аналитические данные'),
        Table(2, 'Сырые данные'),
        Table(3, 'Архивные записи'),
        Table(4, 'Административные таблицы'),
        Table(5, 'Временные данные'),
    ]

    rows = [
        Row(1, 'Строка1', 150, 1),
        Row(2, 'Строка2', 200, 2),
        Row(3, 'Строка3', 180, 1),
        Row(4, 'Строка4', 220, 3),
```

```
        Row(5, 'Строка5', 190, 4),
        Row(6, 'Строка6', 210, 1),
        Row(7, 'Строка7', 170, 4),
    ]

    rows_tables = [
        RowTable(1, 1),
        RowTable(1, 3),
        RowTable(1, 6),
        RowTable(2, 2),
        RowTable(3, 4),
        RowTable(4, 5),
        RowTable(4, 7),
        RowTable(5, 1),
        RowTable(5, 2),
    ]

    return tables, rows, rows_tables


def create_one_to_many(tables, rows):
    return [(r.name, r.value, t.name)
            for t in tables
            for r in rows
            if r.table_id == t.id]


def create_many_to_many(tables, rows, rows_tables):
    many_to_many_temp = [(t.name, rt.table_id, rt.row_id)
                         for t in tables
                         for rt in rows_tables
                         if t.id == rt.table_id]

    return [(r.name, r.value, table_name)
            for table_name, table_id, row_id in many_to_many_temp
            for r in rows if r.id == row_id]


def task_g1(one_to_many, tables):
```

```
    result = {}
    for t in tables:
        if t.name.startswith('A'):
            table_rows = [row_name for row_name, _, table_name in one_to_many
                          if table_name == t.name]
            result[t.name] = table_rows
    return result


def task_g2(one_to_many, tables):
    result_unsorted = []
    for t in tables:
        table_values = [value for _, value, table_name in one_to_many
                        if table_name == t.name]
        if table_values:
            max_value = max(table_values)
            result_unsorted.append((t.name, max_value))

    return sorted(result_unsorted, key=itemgetter(1))


def task_g3(many_to_many, tables):
    table_groups = {}
    for row_name, _, table_name in many_to_many:
        if table_name not in table_groups:
            table_groups[table_name] = []
        if row_name not in table_groups[table_name]:
            table_groups[table_name].append(row_name)

    return dict(sorted(table_groups.items(), key=itemgetter(0)))


def main():
    tables, rows, rows_tables = create_test_data()
    one_to_many = create_one_to_many(tables, rows)
    many_to_many = create_many_to_many(tables, rows, rows_tables)

    print('Задание Г1')
    print(task_g1(one_to_many, tables))
```

```
    print('\nЗадание Г2')
    print(task_g2(one_to_many, tables))

    print('\nЗадание Г3')
    print(task_g3(many_to_many, tables))


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

Test_RK2.py:

```
import unittest
from rk2 import (
    Row, Table, RowTable,
    create_test_data, create_one_to_many, create_many_to_many,
    task_g1, task_g2, task_g3
)


class TestProgram(unittest.TestCase):

    def setUp(self):
        self.tables, self.rows, self.rows_tables = create_test_data()
        self.one_to_many = create_one_to_many(self.tables, self.rows)
        self.many_to_many = create_many_to_many(self.tables, self.rows,
self.rows_tables)

    def test_task_g1(self):
        result = task_g1(self.one_to_many, self.tables)

        expected_tables = ['Аналитические данные', 'Архивные записи',
'Административные таблицы']
        for table_name in expected_tables:
            self.assertIn(table_name, result)

        self.assertEqual(len(result['Аналитические данные']), 3)
```

```
        self.assertEqual(result['Аналитические данные'], ['Строка1', 'Строка3',
'Строка6'])
        self.assertEqual(result['Архивные записи'], ['Строка4'])
        self.assertEqual(result['Административные таблицы'], ['Строка5',
'Строка7'])

    def test_task_g2(self):
        result = task_g2(self.one_to_many, self.tables)

        expected_result = [
            ('Административные таблицы', 190),
            ('Сырые данные', 200),
            ('Аналитические данные', 210),
            ('Архивные записи', 220)
        ]

        self.assertEqual(len(result), 4)
        self.assertEqual(result, expected_result)

        sorted_values = [item[1] for item in result]
        self.assertEqual(sorted_values, sorted(sorted_values))

    def test_task_g3(self):
        result = task_g3(self.many_to_many, self.tables)

        expected_keys = ['Административные таблицы', 'Аналитические данные',
                         'Архивные записи', 'Временные данные', 'Сырые данные']

        self.assertEqual(list(result.keys()), sorted(expected_keys))

        self.assertEqual(len(result['Административные таблицы']), 2)
        self.assertEqual(result['Административные таблицы'], ['Строка5',
'Строка7'])
        self.assertEqual(result['Аналитические данные'], ['Строка1', 'Строка3',
'Строка6'])


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

Результаты программы:

Launching unittests with arguments python -m unittest C:\Users\YARRS\vkprog\pythonProject2\RK2\Test_RK2.py in C:\Users\YARRS\vkprog\pythonProject2\RK2

Ran 3 tests in 0.002s

OK