

**Московский государственный технический
университет им. Н. Э. Баумана**

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»
Отчет по рубежному контролю №2
«Модульное тестирование»

Выполнил:
Студент группы ИУ5И-31Б

Хамет Виам

Проверил:
Гапанюк Ю. Е.

2025 г.

Листинг программы

main.py:

```
from operator import itemgetter

# -----
# 1. Классы данных (Библиотека - Язык программирования)
# -----

class Lang:
    """Язык программирования (Many-side)"""
    def __init__(self, id, name, popularity, lib_id):
        self.id = id
        self.name = name
        self.popularity = popularity
        self.lib_id = lib_id

class Lib:
    """Библиотека (One-side)"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class LangLib:
    """
    'Языки программирования в библиотеке' для реализации
    связи многие-ко-многим
    """
    def __init__(self, lib_id, lang_id):
        self.lib_id = lib_id
        self.lang_id = lang_id

# -----
# 2. Функции для работы с данными
# -----

def get_one_to_many(libs, langs):
    """
    Создает соединение один-ко-многим

    Args:
        libs: список библиотек
    """
```

```

        langs: список языков

Returns:
    список кортежей (lang.name, lang.popularity, lib.name)
"""
return [(l.name, l.popularity, b.name)
        for b in libs
        for l in langs
        if l.lib_id == b.id]

```

```

def get_many_to_many(libs, langs, lang_libs):
    """
    Создает соединение многие-ко-многим

    Args:
        libs: список библиотек
        langs: список языков
        lang_libs: список связей многие-ко-многим

    Returns:
        список кортежей (lang.name, lang.popularity, lib.name)
    """
    many_to_many_temp = [(b.name, ll.lib_id, ll.lang_id)
                          for b in libs
                          for ll in lang_libs
                          if b.id == ll.lib_id]

    return [(l.name, l.popularity, lib_name)
            for lib_name, lib_id, lang_id in many_to_many_temp
            for l in langs if l.id == lang_id]

```

```

def filter_langs_ending_with(one_to_many, ending):
    """
    Д1: Фильтрует языки по окончанию названия

    Args:
        one_to_many: соединение один-ко-многим
        ending: строка окончания для фильтрации

    Returns:
        отсортированный список языков и их библиотек
    """
    filtered = list(filter(lambda x: x[0].endswith(ending), one_to_many))

```

```

return sorted(filtered, key=itemgetter(0))

def get_libs_avg_popularity(libs, one_to_many):
    """
    Д2: Вычисляет среднюю популярность языков для каждой библиотеки

    Args:
        libs: список библиотек
        one_to_many: соединение один-ко-многим

    Returns:
        список кортежей (lib.name, avg_popularity), отсортированный по убыванию
    """
    result = []
    for b in libs:
        b_langs = list(filter(lambda x: x[2] == b.name, one_to_many))

        if len(b_langs) > 0:
            b_popularities = [pop for _, pop, _ in b_langs]
            avg_popularity = sum(b_popularities) / len(b_popularities)
            result.append((b.name, avg_popularity))

    return sorted(result, key=itemgetter(1), reverse=True)

def get_libs_starting_with_langs(libs, many_to_many, prefix):
    """
    Д3: Получает библиотеки, начинающиеся с определенного префикса,
    и список языков в них

    Args:
        libs: список библиотек
        many_to_many: соединение многие-ко-многим
        prefix: префикс для фильтрации названий библиотек

    Returns:
        словарь {lib.name: [lang.name, ...]}
    """
    result = {}
    for b in libs:
        if b.name.startswith(prefix):
            b_langs = list(filter(lambda x: x[2] == b.name, many_to_many))
            b_lang_names = [name for name, _, _ in b_langs]
            result[b.name] = b_lang_names

```

```

        return result

# -----
# 3. Функция для получения тестовых данных
# -----

def get_test_data():
    """
    Возвращает тестовые данные

    Returns:
        кортеж (libs, langs, lang_libs)
    """
    libs = [
        Lib(1, 'Стандартная библиотека Python'),
        Lib(2, 'Библиотека для работы с данными'),
        Lib(3, 'Библиотека для веб-разработки'),
        Lib(4, 'Библиотека для машинного обучения'),
        Lib(5, 'Библиотека для научных вычислений'),
    ]

    langs = [
        Lang(1, 'Python', 10, 1),
        Lang(2, 'Java', 8, 2),
        Lang(3, 'JavaScript', 9, 3),
        Lang(4, 'C++', 7, 1),
        Lang(5, 'C#', 6, 2),
        Lang(6, 'Ruby', 4, 3),
        Lang(7, 'Go', 5, 4),
        Lang(8, 'Rust', 5, 4),
        Lang(9, 'Haskell', 2, 5),
        Lang(10, 'Scala', 3, 5),
    ]

    lang_libs = [
        LangLib(1, 1), LangLib(2, 1), LangLib(3, 1), LangLib(4, 1), LangLib(5,
1),
        LangLib(2, 2), LangLib(3, 3), LangLib(1, 4), LangLib(4, 7), LangLib(4,
8),
        LangLib(5, 9), LangLib(5, 10),
    ]

```

```

    return libs, langs, lang_libs

# -----
# 4. Основная функция
# -----

def main():
    """Основная функция"""
    libs, langs, lang_libs = get_test_data()

    one_to_many = get_one_to_many(libs, langs)
    many_to_many = get_many_to_many(libs, langs, lang_libs)

    print('Задание Д1 (Один-ко-многим)')
    res_d1 = filter_langs_ending_with(one_to_many, 'on')
    for item in res_d1:
        print(f'Язык: {item[0]}, Популярность: {item[1]}, Библиотека: {item[2]}')

    print('\nЗадание Д2 (Один-ко-многим)')
    res_d2 = get_libs_avg_popularity(libs, one_to_many)
    for lib_name, avg_pop in res_d2:
        print(f'Библиотека: {lib_name}, Средняя популярность: {avg_pop:.2f}')

    print('\nЗадание Д3 (Многие-ко-многим)')
    res_d3 = get_libs_starting_with_langs(libs, many_to_many, 'Б')
    for lib_name, lang_names in res_d3.items():
        print(f'Библиотека: {lib_name}')
        for lang_name in lang_names:
            print(f' - {lang_name}')

if __name__ == "__main__":
    main()

```

test_main.py:

```

import unittest

# -----
# Импортируем функции из основного файла
# Предполагается, что основной файл называется data_processing.py
# -----

try:
    from data_processing import (

```

```

    Lang, Lib, LangLib,
    get_one_to_many, get_many_to_many,
    filter_langs_ending_with,
    get_libs_avg_popularity,
    get_libs_starting_with_langs
)
except ImportError:
    # Если импорт не удался, определяем классы и функции здесь
    class Lang:
        def __init__(self, id, name, popularity, lib_id):
            self.id = id
            self.name = name
            self.popularity = popularity
            self.lib_id = lib_id

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

    class LangLib:
        def __init__(self, lib_id, lang_id):
            self.lib_id = lib_id
            self.lang_id = lang_id

    def get_one_to_many(libs, langs):
        return [(l.name, l.popularity, b.name)
                for b in libs
                for l in langs
                if l.lib_id == b.id]

    def get_many_to_many(libs, langs, lang_libs):
        many_to_many_temp = [(b.name, ll.lib_id, ll.lang_id)
                              for b in libs
                              for ll in lang_libs
                              if b.id == ll.lib_id]

        return [(l.name, l.popularity, lib_name)
                for lib_name, lib_id, lang_id in many_to_many_temp
                for l in langs if l.id == lang_id]

    def filter_langs_ending_with(one_to_many, ending):
        from operator import itemgetter
        filtered = list(filter(lambda x: x[0].endswith(ending), one_to_many))
        return sorted(filtered, key=itemgetter(0))

```

```

def get_libs_avg_popularity(libs, one_to_many):
    result = []
    for b in libs:
        b_langs = list(filter(lambda x: x[2] == b.name, one_to_many))

        if len(b_langs) > 0:
            b_popularities = [pop for _, pop, _ in b_langs]
            avg_popularity = sum(b_popularities) / len(b_popularities)
            result.append((b.name, avg_popularity))

    from operator import itemgetter
    return sorted(result, key=itemgetter(1), reverse=True)

def get_libs_starting_with_langs(libs, many_to_many, prefix):
    result = {}
    for b in libs:
        if b.name.startswith(prefix):
            b_langs = list(filter(lambda x: x[2] == b.name, many_to_many))
            b_lang_names = [name for name, _, _ in b_langs]
            result[b.name] = b_lang_names

    return result

# -----
# Тестовый класс
# -----

class TestLangLibSystem(unittest.TestCase):
    """Тесты для системы языков и библиотек"""

    def setUp(self):
        """Подготовка тестовых данных перед каждым тестом"""
        self.libs = [
            Lib(1, 'Стандартная библиотека Python'),
            Lib(2, 'Библиотека для работы с данными'),
            Lib(3, 'Библиотека для веб-разработки'),
        ]

        self.langs = [
            Lang(1, 'Python', 10, 1),
            Lang(2, 'Java', 8, 2),
            Lang(3, 'JavaScript', 9, 3),
            Lang(4, 'C++', 7, 1),

```

```

]

self.lang_libs = [
    LangLib(1, 1), # Python в библиотеке 1
    LangLib(2, 1), # Python в библиотеке 2
    LangLib(2, 2), # Java в библиотеке 2
    LangLib(3, 3), # JavaScript в библиотеке 3
]

def test_filter_langs_ending_with_on(self):
    """
    Тест Д1: Проверка фильтрации языков,
    заканчивающихся на 'on'
    """
    # Arrange (Подготовка)
    one_to_many = get_one_to_many(self.libs, self.langs)

    # Act (Действие)
    result = filter_langs_ending_with(one_to_many, 'on')

    # Assert (Проверка)
    # Должен вернуться только Python
    self.assertEqual(len(result), 1)
    self.assertEqual(result[0][0], 'Python')
    self.assertEqual(result[0][1], 10)
    self.assertEqual(result[0][2], 'Стандартная библиотека Python')

    # Проверяем, что результат отсортирован
    lang_names = [item[0] for item in result]
    self.assertEqual(lang_names, sorted(lang_names))

def test_get_libs_avg_popularity(self):
    """
    Тест Д2: Проверка вычисления средней популярности
    языков в библиотеках
    """
    # Arrange
    one_to_many = get_one_to_many(self.libs, self.langs)

    # Act
    result = get_libs_avg_popularity(self.libs, one_to_many)

    # Assert
    # Проверяем количество библиотек с языками
    self.assertEqual(len(result), 3)

```

```

# Проверяем сортировку по убыванию популярности
popularities = [item[1] for item in result]
self.assertEqual(popularities, sorted(popularities, reverse=True))

# Проверяем конкретные значения
# Библиотека 1: (Python=10 + C++=7) / 2 = 8.5
# Библиотека 2: Java=8
# Библиотека 3: JavaScript=9
lib_dict = {name: pop for name, pop in result}

self.assertEqual(
    lib_dict['Стандартная библиотека Python'],
    8.5,
    places=2
)
self.assertEqual(
    lib_dict['Библиотека для работы с данными'],
    8.0,
    places=2
)
self.assertEqual(
    lib_dict['Библиотека для веб-разработки'],
    9.0,
    places=2
)

def test_get_libs_starting_with_prefix(self):
    """
    Тест ДЗ: Проверка получения библиотек по префиксу
    и списка языков в них
    """

    # Arrange
    many_to_many = get_many_to_many(self.libs, self.langs, self.lang_libs)

    # Act
    result = get_libs_starting_with_langs(self.libs, many_to_many, 'Б')

    # Assert
    # Должны быть две библиотеки, начинающиеся с 'Б'
    self.assertEqual(len(result), 2)
    self.assertIn('Библиотека для работы с данными', result)
    self.assertIn('Библиотека для веб-разработки', result)

    # Проверяем языки в библиотеке для работы с данными

```

```
data_lib_langs = result['Библиотека для работы с данными']
self.assertIn('Python', data_lib_langs)
self.assertIn('Java', data_lib_langs)

# Проверяем языки в библиотеке для веб-разработки
web_lib_langs = result['Библиотека для веб-разработки']
self.assertIn('JavaScript', web_lib_langs)
```

```
class TestEdgeCases(unittest.TestCase):
    """Дополнительные тесты для проверки граничных случаев"""

    def test_empty_data(self):
        """Тест с пустыми данными"""
        empty_libs = []
        empty_langs = []

        one_to_many = get_one_to_many(empty_libs, empty_langs)
        result = filter_langs_ending_with(one_to_many, 'on')

        self.assertEqual(len(result), 0)

    def test_no_matching_langs(self):
        """Тест когда нет языков с заданным окончанием"""
        libs = [Lib(1, 'Test Lib')]
        langs = [Lang(1, 'Java', 8, 1)]

        one_to_many = get_one_to_many(libs, langs)
        result = filter_langs_ending_with(one_to_many, 'on')

        self.assertEqual(len(result), 0)

    def test_no_matching_prefix(self):
        """Тест когда нет библиотек с заданным префиксом"""
        libs = [Lib(1, 'Test Library')]
        langs = [Lang(1, 'Python', 10, 1)]
        lang_libs = [LangLib(1, 1)]

        many_to_many = get_many_to_many(libs, langs, lang_libs)
        result = get_libs_starting_with_langs(libs, many_to_many, 'Б')

        self.assertEqual(len(result), 0)

if __name__ == '__main__':
```

```
unittest.main(verbosity=2)
```

Результат выполнения:

```
• (venv) wiame@HP:~/Wiame-Pradigm-2025/RK2$ python3 test_main.py
test_empty_data (__main__.TestEdgeCases.test_empty_data)
Тест с пустыми данными ... ok
test_no_matching_langs (__main__.TestEdgeCases.test_no_matching_langs)
Тест когда нет языков с заданным окончанием ... ok
test_no_matching_prefix (__main__.TestEdgeCases.test_no_matching_prefix)
Тест когда нет библиотек с заданным префиксом ... ok
test_filter_langs_ending_with_on (__main__.TestLangLibSystem.test_filter_langs_ending_with_on)
Тест Д1: Проверка фильтрации языков, ... ok
test_get_libs_avg_popularity (__main__.TestLangLibSystem.test_get_libs_avg_popularity)
Тест Д2: Проверка вычисления средней популярности ... ok
test_get_libs_starting_with_prefix (__main__.TestLangLibSystem.test_get_libs_starting_with_prefix)
Тест Д3: Проверка получения библиотек по префиксу ... ok
```

```
-----
Ran 6 tests in 0.002s
```

```
OK
```