Министерство науки и высшего образования РФ Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по дисциплине

“Тестирование программного

обеспечения”

Выполнил:

Студент группы P3330,

Тарасов Иван Сергеевич

Проверил:

Райла Мартин

 Санкт-Петербург

2025

**Основное задание. Вариант 7755**

**Изображение выглядит как текст, Шрифт, рукописный текст, линия

Контент, сгенерированный ИИ, может содержать ошибки.**

1. Все составляющие систему функции (как тригонометрические, так и логарифмические) должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).
2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции sin(x)):
3. Обе "базовые" функции (в примере выше - sin(x) и ln(x)) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.
5. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в сsv файл вида «X, Результаты модуля (X)», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

Код: [Software-Testing/lab2 at migration · IoannTar2004/Software-Testing](https://github.com/IoannTar2004/Software-Testing/tree/migration/lab2)

**Диаграмма классов**

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, 3D-моделирование

Контент, сгенерированный ИИ, может содержать ошибки.

**Тестовые покрытия**

Натуральный логарифм

- небольшие числа (2-10)

- аргумент «1», при котором значение логарифма равно «0».

- числа в пределах 0 < x < 1

- большие числа, при которых значение моего логарифма отличается от реального с большой погрешностью

- Отрицательные числа.

Обычный логарифм

- небольшие числа

- основание равно 0 или 1

- аргумент меньше или равен 0

- все остальные случаи, как у натурального логарифма, так как обычный логарифм реализован на основе натурального.

Синус:

- табличные значения (0, pi / 4, -pi / 4, pi / 2)

- большие числа

Косинус:

- табличные значения (0, pi / 4, -pi / 4, pi / 2)

- большие числа

Тангенс:

- табличные значения (0, pi / 4, -pi / 4)

- числа, близкие к +-pi / 2 + 2pi\*k.

Тангенс:

- табличные значения (pi / 2, pi / 3, -pi / 3)

- числа, близкие к 0 и pi + 2pi\*k.

Косеканс:

- табличные значения

- числа, близкие к 0 и pi + 2pi\*k.

**Тестирование асинхронных процессов**

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

**Пояснение**: первый тест проверяет то, что метод createGame выполнится не больше, чем за 5 секунд. CreateGame выполняется асинхронно в течение случайного промежутка времени (от 1 до 4 секунд). Второй тест проверяет корректность отмены создания игры до того, как игра будет создана.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Пояснение**: имеется 3 асинхронных метода у сервиса: addPunishmentMap, incrementTime, removeFromPunishmentMap. Первый добавляет игрока в punishmentMap, где ключ – является ник игрока, а значение количество секунд, которое он пробыл в наказании. Второй метод увеличивает количество секунд у каждого игрока в этой map каждую секунду. Последний метод удаляет игрока из map, если его время нахождения превышает аргумент этого метода. Данный тест проверяет зависимость между 3 этими асинхронными методами.

Основные стратегии тестирования асинхронных методов

1️) **Ожидание завершения** – блокирующее ожидание результата перед проверкой (get(), join()).

2) **Polling (опрос состояния)** – периодическая проверка результата в течение заданного времени.

3) **Callbacks & Listeners –** подписка на завершение выполнения**.**

4) **Использование специальных тестовых инструментов** – например, Awaitility.

Возможные проблемы при тестировании асинхронного кода

1) **Гонки**

2) **Тесты проходят случайно**

3) **Утечки потоков** – не завершенные потоки, работающие в фоне

4) **Зависание тестов** – ожидание завершения задачи может привести к бесконечному ожиданию.

Ограничения при тестировании

1. Неопределенность времени выполнения
2. Сложность отладки
3. Ограниченная предсказуемость поведения

**Интеграционное тестирование базы данных через Testcontainers и Liquibase**

Стратегии тестирования

1. Подготовка тестов и тестового окружения (инициализация Docker контейнера и базы данных в нём)
2. Применение миграций
3. Проверка структуры таблиц
4. Проверка откатов миграций (при необходимости)
5. Тестирование CRUD-операций в ожидаемое время

Анализ возможных проблем

1. Медленный запуск контейнера
2. Проблемы с консистентностью данных - если тесты не сбрасывают данные, это может привести к непредсказуемому поведению.
3. Проблемы совместимости среды. Например, различия версий PostgreSQL в контейнере и в продакшене.