**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики» Рефакторинг баз данных и приложений**

Лабораторная работа №2

Выполнил: Студент группы

БПИ2301

Быков Максим   
Эдуардович

Москва

2025

**Цели лабораторной работы**

1. Изучить основные принципы и задачи модульного тестирования.
2. Научиться разрабатывать модульные тесты для проверки функциональности отдельных частей приложения.
3. Применить модульное тестирование для контроля качества кода в процессе рефакторинга.
4. Освоить использование инструментов для написания и запуска модульных тестов.
5. **Задание**
6. Ознакомьтесь с теоретическими основами модульного тестирования, его задачами и целями. Изучите, как модульное тестирование помогает проверять функциональность отдельных частей программы (модулей) и выявлять ошибки на ранних этапах разработки.
7. Выберите небольшой проект или модуль, который вы ранее разработали, либо используйте предоставленный преподавателем пример.
8. Определите ключевые функции или классы, которые требуют тестирования. Выявите участки кода, которые могут быть улучшены в процессе рефакторинга.
9. Напишите модульные тесты для выбранных функций или классов. Используйте подходы, описанные в теории, такие как тестирование граничных условий, проверка исключений и тестирование на корректность данных. Примените инструменты для модульного тестирования, такие как JUnit (для Java), pytest (для Python) или аналогичные для вашего языка программирования.
10. Проведите рефакторинг выбранного кода, улучшая его структуру, читаемость и производительность. После каждого изменения запускайте модульные тесты, чтобы убедиться, что функциональность программы не нарушена.
11. Подготовьте отчет, включающий описание исходного состояния кода, список написанных модульных тестов с пояснениями, внесенные изменения в код и их обоснование, результаты выполнения тестов до и после рефакторинга.

# Ход работы

# Код, в который были добавлены тесты был взят из прошлой лабораторной работы

# Перенос проекта на maven

# Изначально проект находился на обычной build system – intellij. Но для удобства проведения тестов с помощью JUnit был перенесен на maven.

# Была создана новая директория и структура кода, добавлен pom.xml файл и необходимые для начала зависимости

# До:

# 

# После:

# 

* 1. **Добавление тестов для логики игры**

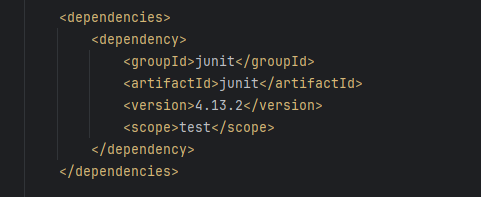
Был написан класс GameLogicTest, где проверяются следующие функции:

generateUniqueNumbers — проверяется генерация уникальных чисел и их правильность (16 чисел в пределах заданного диапазона и без повторений).

isValidNext — проверяется, что следующий выбор пользователя правильный в контексте текущей последовательности.

**1.3 Добавление зависимостей для тестирования:**

В pom.xml был добавлен JUnit для написания и выполнения тестов.



# 1.4. Пример тестов

# Изображение выглядит как текст, снимок экрана, число, календарь Контент, сгенерированный ИИ, может содержать ошибки.

# Изображение выглядит как текст, снимок экрана, программное обеспечение Контент, сгенерированный ИИ, может содержать ошибки.

# 1.5. Этот код — это GUI-приложение на AWT, и тестировать напрямую такие приложения непросто обычными JUnit-тестами. Поэтому раздеялем логику и интерфейс, чтобы было удобно тестировать.

# 

# 1.6. Результаты тестов

# 

# 1.7 Изменение кода. Был взят не отформатированный код из прошлой лабораторной работы и созданы для него тесты.

# 

# ССЫЛКА НА РЕПОЗИТОРИЙ

https://github.com/A1e0nd/refactoring2

# ВЫВОД

* Тесты позволили убедиться, что генерация уникальных чисел работает корректно, и что проверка правильности следующего числа также правильно выполняется.
* **Удобство добавления новых тестов**: сейчас можно без проблем добавлять тесты для других методов класса GameLogic.
* Код стал более устойчивым к ошибкам и легче поддерживаемым.