

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение*  *высшего образования*  ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных

технологий (МОСИТ)

ПРАКТИЧЕСКАЯ РАБОТА №2

по дисциплине

**«Тестирование и верификация программного обеспечения»**

Тема: **«Тестирование черного ящика»**

**Студент группы** ИКБО-20-21

Сидоров С.Д.

(подпись студента)

**Принял руководитель работы** Овчинникова М.А.

(подпись руководителя)

Практические работы выполнены « » 2023 г.

Зачтено « » 2023 г.

Содержание

[1. Цель работы 3](#_Toc145500964)

[1.1 Задание для выполнения 3](#_Toc145500965)

[1.2 Материальная часть 3](#_Toc145500966)

[1.2.1 Динамический анализ 3](#_Toc145500967)

[1.2.2 Статический анализ 4](#_Toc145500968)

[1.2.3 Тестирование методом черного ящика 4](#_Toc145500969)

[2. Инструменты 4](#_Toc145500970)

[2.1. Используемое ПО 4](#_Toc145500971)

[3. Выполнение задания 5](#_Toc145500972)

[3.1. Техническое задание 5](#_Toc145500973)

[3.2. Документация 8](#_Toc145500974)

[3.3. Анализ и уточнение требований 15](#_Toc145500975)

[3.2 Планирование тестирования 15](#_Toc145500976)

[3.3 Разработка тестов 22](#_Toc145500977)

[3.4 Выполнение тестов 22](#_Toc145500978)

[4. Выводы 27](#_Toc145500979)

[5. Список использованных источников 28](#_Toc145500980)

[6. Дополнения и приложения 28](#_Toc145500981)

# Цель работы

## 1.1 Задание для выполнения

Составление документации к программному продукту, анализ документации другого участница группы, написание модульных тестов в соответствии с спецификацией AAA ( Arrange, Act, Assert).

## 1.2 Материальная часть

## 1.2.1 Модульное тестирование

Модульное тестирование -метод тестирования, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки.

## 1.2.3 Спецификация AAA

Спецификация AAA ( Arrange, Act, Assert) - (входные данные, действие, ожидаемый результат.

# Инструменты

## Используемое ПО

Для проведения тестирования программного продукта написанного на языке Java использовалась библиотека JUnit.

1. **Выполнение задания**
   1. **Составление документации к собственному программному продукту.**

Программный продукт состоит из класса Main, содержащего три метода.

Метод 1 - main, сгенерированный метод для запуска программы и отладки в ручном режиме.

**Метод 2 - solveQuadraticEquation**.

Функционал: вычислений корней квадратного уравнения.

Входные данные:

Int a - коэффициент перед неизвестным во второй степени;

Int b - коэффициент перед неизвестным в первой степени;

Int c - свободный член уравнения;

Ограничения на входные данные: отсутствуют

Выходные данные:

Строка типа String;

Возможные варианты выходных данных:

“none” - уравнение не имеет корней;

“корень” - уравнение с одним корнем;

“корень/корень” - уравнение с двумя корнями;

**Метод 3 - solveCubicEquation.**

Функционал: вычисление корней кубического уравнения.

Входные данные:

Int a - коэффициент перед неизвестным в третьей степени;

Int b - коэффициент перед неизвестным во второй степени;

Int c - коэффициент перед неизвестным в первой степени;

Int d - свободный член уравнения;

Ограничения на входные данные: отсутствуют.

Выходные данные:

Строка типа String;

Возможные варианты выходных данных

“none” - уравнение не имеет корней;

“корень” - уравнение с одним корнем;

“корень/корень” - уравнение с двумя корнями;

“корень/корень/корень” - уравнение с тремя корнями;

* 1. **Документация к программному продукту другого участника группы**
  2. **Тестирование программного продукта другого участника группы**
     1. **План тестирования.**

Выполнение практической работы подразумевает проведение следующих этапов.

* Проведение модульного тестирования
  + **Тестирование метода NOD**
    - Тестирование корректной работы при стандартных данных
    - Тестирование корректной работы при крайних случаях
  + **Тестирование метода NOK**
    - Тестирование корректной работы при стандартных данных
    - Тестирование корректной работы при крайних случаях
    1. **Создание модульных тестов**
       - **Тесты метода NOD**

Для тестирования метода NOD были проверены четыре различных тестовых случая.

Случай первый с стандартным набором данных при существовании отличного от 1 НОД продемонстрирован на рисунке 1.

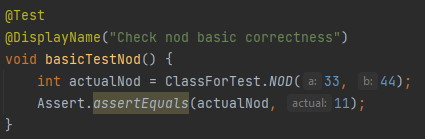


Рисунок 1 - Тесты на стандартных значениях

Случай второй с одинаковыми входными параметрами продемонстрирован на рисунке 2.

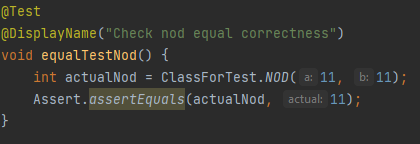


Рисунок 2 - Тесты метода NOD с одинаковым входными параметрами

Случай третий с измененным порядком входных данных представлен на рисунке 3.

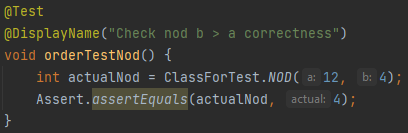


Рисунок 3 - Тесты метода NOD с входными данными в обратном порядке

Случай четвёртый с стандартным набором данным предполагающим отсутствием НОД не равным 1 представлено на рисунке 4.

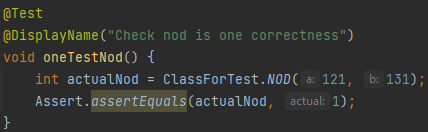


Рисунок 4 - Тесты метода NOD с крайними данными

* + - * **Тесты метода NOK**

Для тестирования метода NOK были проверены 8 тестовых случаев.

Первые два случая со стандартными входными данными в различном порядке представлены на рисунке 5.

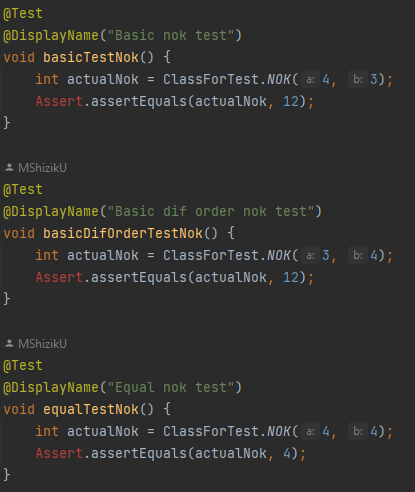


Рисунок 5 - Тесты метода NOK на стандартных значениях

Также тестирование было проведено с отрицательными входными данными (рис. 6).

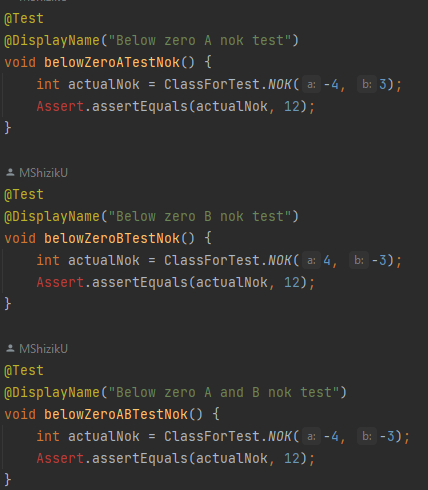


Рисунок 6 - Тесты метода NOK на отрицательных значениях

В ходе тестирования также были проведены тесты с нулевыми значения в качестве входных параметров (рис. 7).



Рисунок 7 - Тесты метода NOK на нулевых значениях

* + 1. **Проведение тестирования**
       - **Тестирование метода NOD**

Таблица 1 - результаты проведения тестирования метода NOD

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Полученный результат** |
| 33 44 | 11 | 11 |
| 11 11 | 11 | 11 |
| 12 4 | 4 | 4 |
| 121 131 | 1 | 1 |

* + - * **Тестирование метода NOK**

Таблица 2 - Результаты тестирования метода NOK

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Полученный результат** |
| 4 3 | 12 | 12 |
| 3 4 | 12 | 12 |
| 4 4 | 4 | 4 |
| -4 3 | 12 | 12 |
| 4 -3 | 12 | 12 |
| -4 -3 | 12 | 12 |
| 0 3 | 0 | 0 |
| 3 0 | 0 | 0 |
| 0 0 | 0 | 0 |

Подтверждение описанных ранее результатов представлено на рисунках 8,9.

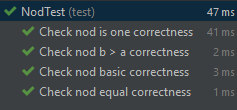


Рисунок 8 - результаты тестирования метода NOD.

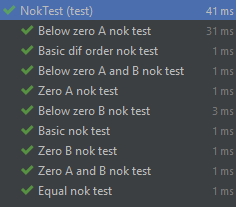


Рисунок 9 - результаты тестирования метода NOK.

* 1. **Оценка**

В ходе анализа документации ошибок не было выявлено. В ходе проведённого тестирования не было выявлено ошибок, код отрабатывает во всех ожидаемых случаях.

# Выводы

В ходе данной практической работы были продемонстрированы возможности тестирования программного продукта с помощью модульного тестирования согласно патерну AAA.

# Список использованных источников

1. Статический анализ кода [Электронный ресурс] – <https://www.jetbrains.com/ru-ru/resharper/features/code_analysis.html>
2. Динамический анализ кода [Электронный ресурс] – <https://habr.com/ru/companies/pvs-studio/articles/580196/>
3. WebStrom – анализ кода [Электронный ресурс] – <https://www.jetbrains.com/ru-ru/resharper/features/code_analysis.html>
4. Использование статического и динамического анализа для повышения качества продукции и эффективности разработки [Электронный ресурс] – <https://www.swd.ru/print.php3?pid=828>

# Дополнения и приложения