**Содержание**

[Введение 2](#_Toc203501844)

[Анализ решения 3](#_Toc203501845)

[Функциональность 3](#_Toc203501846)

[Производительность 3](#_Toc203501847)

[Удобство использования (юзабилити) 4](#_Toc203501848)

[Масштабируемость 4](#_Toc203501849)

[Сопровождаемость 5](#_Toc203501850)

[Переносимость 5](#_Toc203501851)

[Качество кода 5](#_Toc203501852)

[Тестирование 6](#_Toc203501853)

[Выводы и рекомендации 7](#_Toc203501854)

# **Введение**

Для аналитического обзора проделанной работы в ходе учебной практики была выбрана и решена кейс-задача №1 на языке программирования Java:  
**"Дан одномерный массив A размерности N. Найти сумму отрицательных элементов, расположенных между максимальным и минимальным."**

Программа была разработана с использованием объектно-ориентированного подхода и проанализирована по ключевым критериям качества программного обеспечения. Программное решение проекта разделено на логические модули:

* Main – UI слой;
* TaskSolver – Бизнес-логика;
* TaskResult – Модель данных результата.

# **Анализ решения**

## **Функциональность**

Программа хорошо работает, корректно обрабатывает все входные данные, выдаёт корректный результат и выполняет следующие функции:

* Генерирует массив случайных чисел в заданном диапазоне;
* Находит минимальный и максимальный элементы;
* Суммирует отрицательные числа между ними;
* Выводит результат в консоль.

Выявленные проблемы/Рекомендации:

* Реализовать возможность ручного ввода массива значений в любом диапазоне.

## **Производительность**

В данном решении поставленной задачи не используются сложные алгоритмы, вложенные циклы, рекурсии и т.д., вместо этого используются стандартные операторы, механизмы поиска по массиву. За счёт этого программа работает очень быстро даже на большом объёме данных.

Алгоритм имеет линейную сложность O(n), так как:

* Поиск min/max требует одного прохода по массиву (количество итераций на прямую зависит от размера массива);
* Суммирование отрицательных элементов — ещё один проход в другой функции.

Итого получаем: O(n) + O(n) = O(2n) = O(n)

Выявленные проблемы/Рекомендации:

* Индексы минимального и максимального значения находятся в двух разных функциях, а значит будет каждый раз по 2 прохода по массиву. Можно объединить поиск мин. и макс. сразу в одну функцию.

## **Удобство использования (юзабилити)**

Пользователь может легко пользоваться программой за счёт гибкого интерфейса управления и понятных программных/системных сообщений и подсказок.

Сильные стороны:

* Интерактивный ввод размера массива;
* Четкие сообщения об ошибках (некорректный ввод) с возможностью повторить попытку;
* Наглядный вывод результатов с обработкой разных ситуаций.

Выявленные проблемы/Рекомендации:

* Добавить меню выбора (ручной ввод значений массива / генерация);
* Реализовать цикл выполнения без перезапуска приложения.

## **Масштабируемость**

Данное решение имеет гибкую и лёгкую возможность расширения и модификации за счёт разделения логики на UI-слой, бизнес-логику и модель данных.

Текущая реализация:

* Main.java – UI слой;
* TaskSolver.java – Бизнес-логика;
* TaskResult.java – Модель данных результата.

Выявленные проблемы/Рекомендации:

* Вынести параметры генерации (например параметры диапазона MIN\_VALUE, MAX\_VALUE) в конфигурационный файл;
* Добавить поддержку других типов данных (например, double).

## **Сопровождаемость**

Сопровождаемость программы позволяет достаточно легко понимать и обновлять программу за счёт:

* Четкого разделение кода (хорошо-структурированного) на логические модули;
* Осмысленные названия методов и переменных.

Выявленные проблемы/Рекомендации:

* Добавить описание методов и комментарии к коду к частям алгоритмов;
* Задокументировать входные/выходные параметры.

## **Переносимость**

Программа написана на стандартном Java (без внешних зависимостей), поэтому:

* Работает на любой ОС с JVM, поэтому легко может быть перенесена на другую систему;
* Может быть собрана в JAR для удобного переноса и запуска.

## **Качество кода**

Качество кода данного решения на хорошем уровне и соответствует стандартам Java за счёт:

* Правильные именования (code conventions);
* Логичное разбиение/структурирование кода в проекте;
* Использование стандартных алгоритмов поиска и обработки данных в массиве;
* Использования принципов SOLID (паттерны проектирования).

Выявленные проблемы/Рекомендации:

* Добавить описание методов и комментарии к коду к частям алгоритмов.

## **Тестирование**

Данное решение было протестировано вручную на разных случаях генерации массива – результат всегда был положительным и ожидаемым от программы.

Выявленные проблемы/Рекомендации:

* Написать unit-тесты (функциональные).

# **Выводы и рекомендации**

Программа полностью выполняет поставленную задачу, но требует доработок в:

* Оптимизации производительности;
* Улучшении пользовательского интерфейса;
* Добавлении тестов и документации.

Рекомендации по улучшению:

* Оптимизировать алгоритм (один проход на поиск мин. и макс. вместо двух);
* Реализовать unit-тесты (JUnit);
* Улучшить UX (меню выбора, повторный запуск);
* Добавить документацию (JavaDoc, README) и комментарии в коде.