Network Programming – Java NIO Report

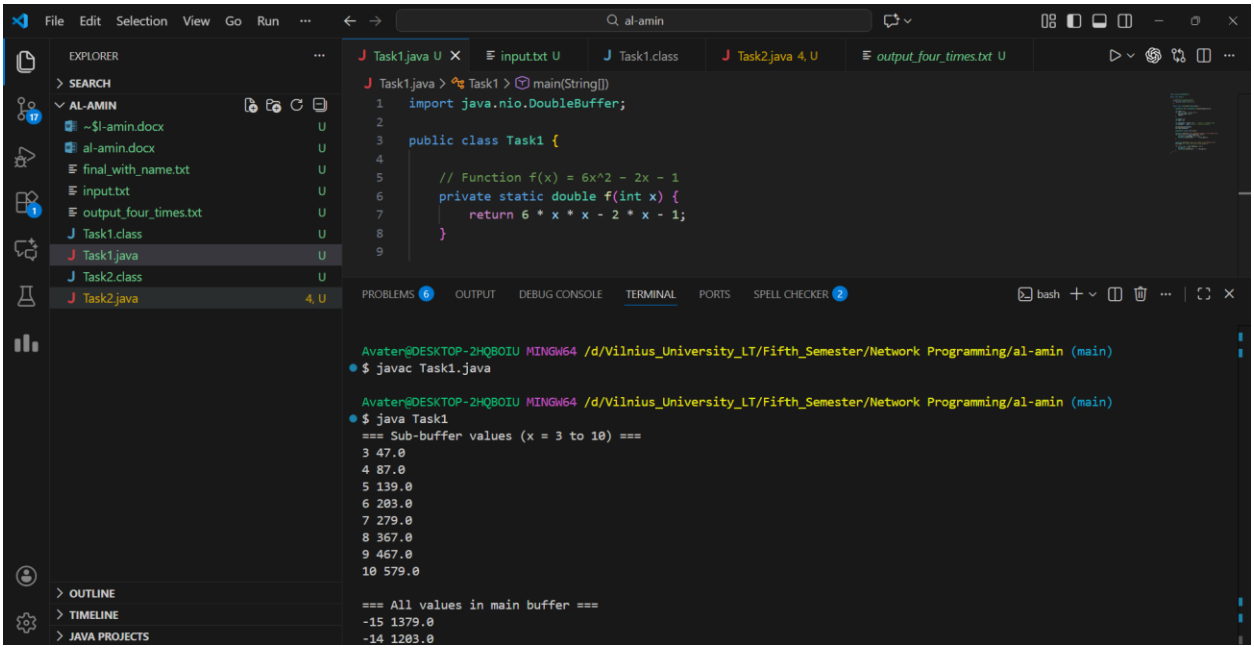
1. Introduction

This report provides a structured explanation of two practical programming tasks completed using Java New I/O (NIO). These tasks focus on buffer manipulation, mathematical computation, sub-buffer slicing, high-performance file copying, and random-access writing. The purpose of the assignment is to reinforce understanding of NIO components such as DoubleBuffer, FileChannel, ByteBuffer, and the use of position/limit/capacity in buffer handling.

2. Task 1 – DoubleBuffer Computation and Slicing

The first task required computing function values of $f(x)=6x^2-2x-1$ for all integers between −15 and 15. A DoubleBuffer with capacity 31 was used to store each computed value. After this, the program extracted a specific range (x = 3 to x = 10) using NIO's slice() method. This demonstrates correct control over buffer state, including setting the position and limit to isolate the relevant portion of data.

The sliced buffer allowed printing the required sub-range, while the full buffer was later reset with clear() to show all values again. This confirms proper usage of NIO buffer methods such as put(), get(), clear(), position(), limit(), and slice().

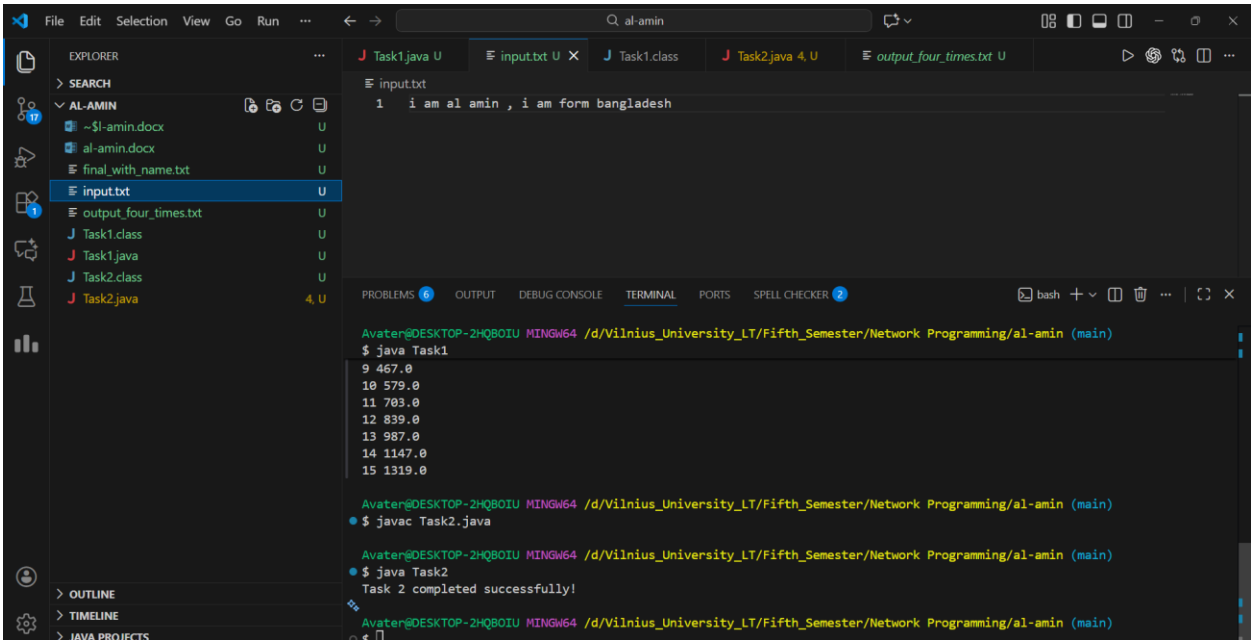


3. Task 2 – File Copying and Writing at Offset

The second task involved two stages: copying the content of input.txt four times into a new file, and then creating a second file where the copied content is duplicated and additional

text is inserted at byte position 200. This was implemented using FileChannel's transferTo() method, which provides high-performance file copying by leveraging the operating system's internal buffer mechanisms.

For the random-access write, channel.position(200) was used to jump to the specified byte offset, after which a ByteBuffer containing the student's name and country was written. When writing beyond the current file size, Java fills the gap with zero bytes (NULL), which is expected behavior in NIO.



4. Discussion

These tasks strengthen understanding of Java NIO through practical usage. Task 1 emphasizes mathematical computation in conjunction with precise memory and buffer operations. Task 2 focuses on file-level operations, including efficient copying and absolute-position writing. These concepts are crucial for applications involving high-speed data processing, networking, and systems programming.

The behavior observed when writing beyond EOF (NULL padding) confirms correct file expansion handling, which is an expected part of working with FileChannel and must be understood when dealing with random-access I/O.

5. Conclusion

Both tasks were completed successfully with correct outputs. Through buffer slicing, mathematical storage, file duplication, and random-access insertion, the assignment

demonstrates a solid grasp of Java NIO fundamentals. These skills form an essential foundation for advanced network programming and file-oriented system design.