Rodrigo Ajmac - 22279
Adrian Arimany - 211063

Hoja trabajo 9 Informe Sobre ASD

The code for the Huffman-based compression and decompression tool is structured around a single App class that orchestrates three main workflows: automatic end-to-end execution (when no arguments are provided), explicit compression (-c argument), and explicit decompression (-d argument). At its core, the createTestFile method ensures that a valid input file exists, populating it with random sample text if necessary. The handleCompression and handleDecompression methods encapsulate the logic for invoking the HuffmanCompressor and HuffmanDecompressor classes, respectively, generating or consuming the .hufftree (serialized code tree) and .huff (bitstream) files. This separation of concerns results in clear, readable code that follows the Single Responsibility Principle, making each part of the pipeline easily testable and maintainable.

The design decisions reflect the classical data compression theory, particularly the use of the Huffman algorithm to achieve minimum-redundancy prefix codes. The code implemented for the Huffman Algorithm is a refractor from GeeksforGeeks (2025). By serializing the code tree into a .hufftree file, the implementation preserves the exact mapping between symbols and bit patterns required for lossless reconstruction. This approach mirrors the method introduced by Huffman (1952), who proved that his algorithm produces optimal prefix codes for a given set of symbol frequencies. Packaging the bitstream separately in the .huff file minimizes storage space by eliminating overhead, while maintaining the integrity of the original message. The code also implements Java's I/O streams and object serialization to handle the tree structure efficiently, ensuring cross-platform compatibility and ease of extension.

The accompanying JUnit tests are designed to validate both the compressor and decompressor in isolation and in combination. The HuffmanCompressorTest verifies that running a full compression–decompression round trip on a known sample yields an output identical to the original input, ensuring that no data is lost or corrupted. It also checks for the existence of the expected output files (.hufftree and .huff). The HuffmanDecompressorTest focuses on the decompressor alone by first generating compressed artifacts with the compressor and then verifying that the decompressor correctly reconstructs the original text. Together, these tests provide high confidence in the correctness and reliability of the implementation, catching regressions in both tree serialization and bitstream decoding logic.

Reference:
GeeksforGeeks. (2025, April 22). *Huffman Coding | Greedy Algo-3*. GeeksforGeeks. https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

Huffman, D. A. (1952). A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9), 1098–1101. https://doi.org/10.1109/JRPROC.1952.273898