1. Introduction

File compression and decompression are essential operations in computing, allowing efficient storage, faster file transfers, and reduced bandwidth usage. This project, **File Compressor and Decompressor**, is designed as a desktop application built using **Java** and **Swing**. It leverages the `java.util.zip` package to provide users with a simple yet powerful interface for compressing multiple files into a single archive and extracting them when needed.

The tool was developed as a mini-project to demonstrate practical implementation of **file handling, GUI design, and data stream manipulation** in Java.

2. Objectives

- To design a user-friendly desktop tool for compressing and decompressing files.
- To integrate **file selection dialogs** for choosing input files and output destinations.
- To implement compression using **`ZipOutputStream`** and extraction using **`ZipInputStream`**.
- To provide a **progress bar** for long operations.
- To display **logs and compression statistics**, including file size reduction and compression ratio.
- To package the project into an **executable JAR file** for portability.

3. Tools and Technologies

- **Programming Language**: Java (JDK 17)
- **GUI Framework**: Swing (JavaFX can also be used)
- **Core Library**: `java.util.zip` for compression and decompression
- **IDE**: Visual Studio Code (with Java Extension Pack)
- **Deliverables**: Source code, executable JAR, demo video

4. System Design and Implementation

4.1 User Interface

The application interface consists of three main panels:

1. **Compression Panel** – Allows users to select multiple files, choose an output ZIP location, and initiate compression.
2. **Decompression Panel** – Provides options to select an existing ZIP file and an extraction directory.
3. **Log Panel** – Displays progress messages and compression statistics such as total input size, compressed size, and efficiency ratio.

4.2 Compression Logic

- Files selected by the user are read in streams.
- Each file is added as a `ZipEntry` into the `ZipOutputStream`.
- The stream writes compressed data block by block, updating the progress bar.
- Compression statistics are calculated after completion.

4.3 Decompression Logic

- A `ZipInputStream` reads each entry from the chosen ZIP archive.
- Files are recreated in the destination directory.
- The progress bar reflects the number of files extracted.

5. Features

- Multiple file compression into a single archive.
- One-click decompression with folder selection.
- Real-time progress tracking for large files.
- Compression ratio and file statistics logging.
- User-friendly Swing interface.
- Executable JAR for easy distribution and usage.

6. Results

The project successfully demonstrates a complete workflow for compressing and decompressing files. The application is lightweight, portable, and performs efficiently for files of varying sizes. The GUI is intuitive, and logs provide users with transparency about the process.

Testing confirmed that:

- Small and large files are compressed without errors.
- Multiple files are handled in a single ZIP.
- Extracted files match the original data.

7. Conclusion

The File Compressor and Decompressor project achieves its objective of providing a simple yet functional desktop tool for managing compressed files. It showcases the practical use of Java’s file handling and GUI frameworks in a real-world context.

In the future, the project could be enhanced with features like password-protected ZIP files, support for additional formats (RAR, 7z), and integration with cloud storage.