If the file you want to compress is not a text file but rather a binary file or any other file format, you can still use Java to read the file by treating it as a stream of bytes. Here's how you can modify the steps to handle non-text files:

1. Read the Input File as Bytes: Instead of reading the file as text, use a **java.io.FileInputStream** or **java.nio.file.Files** to read the file as a stream of bytes. This way, you can read the binary content of the file.
2. Analyze the Frequency of Bytes: Instead of analyzing the frequency of characters, you will analyze the frequency of bytes. Create a frequency table or map to store the count of each byte present in the file. Iterate through the file content as bytes and update the frequency table accordingly.
3. Build the Huffman Tree: The steps to build the Huffman tree remain the same, but you will build the tree based on the frequency of bytes rather than characters.
4. Generate Huffman Codes: Generate Huffman codes for each byte in the same way as before. The generated Huffman codes will represent the binary representation of each byte.
5. Compress the File: Instead of replacing characters with Huffman codes, you will replace bytes with their corresponding Huffman codes. Pack the binary representation of the Huffman codes into a compact format. Write the compressed data, which consists of the encoded bytes, to a new file or the same file.
6. Write the Huffman Tree to the Compressed File: Include the necessary information about the Huffman tree in the compressed file, similar to the previous step. This information will be used during decompression.

When decompressing the file, you will perform the reverse steps, including reading the compressed file as bytes, rebuilding the Huffman tree from the metadata, and decoding the compressed data using the Huffman codes to restore the original file content as bytes.

Remember that during compression and decompression of binary files, you need to handle file read/write operations using appropriate binary I/O methods provided by Java libraries.