```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

/**
 *
 * @author Rasmus Bartholin og Mads Mikael Keinicke
 * Rasmus: rbart17
 * Mads: makei17
 */
public class Encode {

    public static void main(String[] args)
    {
        encode(args[0], args[1]);
    }

    public static void encode(String inputFile, String compPath)
    {
        // Create the HuffTree Object
        HuffTree huff = new HuffTree();

        // Create list of frequencies, ordered by unicode number
        int[] freqs = HuffTree.getFrq(inputFile);

        // Create a heap of Elements with frquencies as key, and ASCII number as data
        PQHeap heapFreq = huff.createHeap(freqs);

        // retrieve the Element containing the created Huffman Tree
        Element tmp = huff.HuffUnify(heapFreq);

        // cast the Tree root from the Element to HuffNode
        HuffNode root = (HuffNode) tmp.getData();

        // Acquire the Array of bitcodes for each ASCII character
        String[] bitCode = huff.findCode(root);

        // Write the proper output to the path for the compressed file
        writeOutput(inputFile, compPath, bitCode, freqs);

    }




    /**
     * The method that writes the Output, from the given parameters gotten in the main method
     *
     * @param filePath
     * @param compPath
     * @param bitCode
     * @param freqs
     */
    public static void writeOutput(String filePath, String compPath, String[] bitCode, int[] freqs)
    {
        // Try statement in case file does not exist
        try {
            // Creation of inputStream, from the given original file
```

```
            FileInputStream fin = new FileInputStream(filePath);

            // Try-with resources statement for the outputstream, in case path cannot be found
            try(FileOutputStream output = new FileOutputStream(compPath)) {

                // creation of bitstream
                BitOutputStream bitStream = new BitOutputStream(output);

                // Write the frequencies of the ASCII characters, for the usage of decoding
                for(int x : freqs)
                {
                    bitStream.writeInt(x);
                }

                // loop for writing ASCII characters as bits, using our bitCodes
                while(fin.available() > 0)
                {
                    // getting the next byte to convert
                    int nextByte = fin.read();

                    // Get the bitcode, corresponding to the equivalent ASCII btyecode
                    String bits = bitCode[nextByte];

                    // Convert the string to an array of strings, each indice is one bit
                    String[] bitArray = bits.split("");

                    for(String x : bitArray)
                    {
                        // convert String of bit to an int
                        int tmp = Integer.parseInt(x);


                        // write int as a bit to the bitstream
                        bitStream.writeBit(tmp);
                    }
                }

                // Close the streams, as we are done using them
                bitStream.close();
                output.close();
            }
            catch(NullPointerException e)
            {
                System.out.println(e);
            }
        } catch (IOException e) {
            System.out.println(e);
        }
    }


}
```