```java
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 Decode {

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

    public static void decode(String cmprsdFile, String outputFile)
    {
        try {

            // creation of input streams
            FileInputStream fin = new FileInputStream(cmprsdFile);
            BitInputStream inpStream = new BitInputStream(fin);

            // try-with resources for the ouput stream
            try(FileOutputStream output = new FileOutputStream(outputFile)) {

                // Retrieve the frequency list, used to encode the file
                int[] freqs = new int[256];
                int d = 0;
                for(int x : freqs)
                {
                    freqs[d] = inpStream.readInt();
                    d++;
                }

                // int to keep track of missing characters.
                int charsLeft = 0;

                // loop to get the amount of characters to be inserted, from the list of frequencies
                for(int x : freqs)
                {

                    charsLeft+= x;
                }

                //Create HuffTree Object
                HuffTree huff = new HuffTree();

                // 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 tmpEl = huff.HuffUnify(heapFreq);

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

                // While loop that stops once all characters has been written
                while(charsLeft > 0)
                {

                    // intialize the next bit integer, that contains the next bit
                    int nextBit = inpStream.readBit();

                    // initialize the node that describes the position in the tree, starting in the root
                    HuffNode tmpNode = root;

                    // A loop that iterates through the tree, until finds a character, within a node
                    while(true)
                    {
                        // if the next bit it 0, go to the left child
                        if(nextBit == 0)
                        {
                            tmpNode = tmpNode.getLchild();
                        }
                        // else if the next bit it 1, go to the right child
                        else if(nextBit == 1)
                        {
                            tmpNode = tmpNode.getRchild();
                        }

                        // If the current node contains data, break the loop
                        if(tmpNode.getData() != null)
                        {
                            break;
                        }

                        // If there is no more bits left, break the loop;
                        else if(nextBit == -1)
                        {
                            break;
                        }

                        // Get the next bit
                        nextBit = inpStream.readBit();
                    }

                    // Retrive the number of the data, which is a int referencing the proper ASCII character, as a byte
                    int outPut = (int) tmpNode.getData();

                    // Write the int, as a byte
                    output.write(outPut);

                    // Subtract number of chars by one.
                    charsLeft --;
                }

                // Close the two streams.
                inpStream.close();
                output.close();
            }
            catch(NullPointerException e)
            {
                System.out.println(e);
            }



        } catch (IOException e) {
            System.out.println(e);

        }
    }
}
```