```java
import java.io.IOException;

public class HuffmanTree {

    private HNode root;

    /**
     * build a Huffman tree using the given characters and corresponding frequencies
     * @param frequencies the corresponding frequencies of given character
     */
    public HuffmanTree(TreeMap<Character, Integer> frequencies){
        PriorityQueue<HNode> pq = new PriorityQueue<HNode>(frequencies.size(), new HNodeComparator<HNode>());
        for (Entry<Character, Integer> entry : frequencies.entrySet()){
            HNode leaf = new HNode(entry.getKey(), entry.getValue());
            pq.add(leaf);
        }
        int pqSize = pq.size();
        for (int i = 1; i < pqSize; i++){
            HNode h1 = (HNode) pq.poll();
            HNode h2 = (HNode) pq.poll();
            HNode h3 = new HNode(h1, h2);
            pq.add(h3);
        }
        root = (HNode) pq.poll();
    }

    /**
     * binary encoding of the given symbol using binary characters '0' and '1'
     * @param symbol the given symbol
     * @return the string of binary encoding
     */
    public String encodeLoop(char symbol){
        String encoded = "";
        HNode curr = root;
        while (!curr.isLeaf()){
            if (curr.leftChild.contains(symbol)){
                encoded = encoded + "0";
                curr = curr.leftChild;
            }
            else if (curr.rightChild.contains(symbol)){
                encoded = encoded + "1";
                curr = curr.rightChild;
            }
        }
        return encoded;
    }


    /**
     * recursive method
     * binary encoding of the given symbol using binary characters '0' and '1'
     * @param symbol the given symbol
     * @return the string of binary encoding
     */
    public String encode(char symbol){
        return encode(symbol, root);
    }


    /**
     * recursive method
     * binary encoding of the given symbol using binary characters '0' and '1' from HNode node
     * @param symbol the given symbol
     * @param node the node to start encoding
     * @return the string of binary encoding
     */
    private String encode(char symbol, HNode node){
        if (node.isLeaf()){
            return "";
        }
        else if (node.leftChild.contains(symbol)){
            return "0" + encode(symbol, node.leftChild);
        }
        else if (node.rightChild.contains(symbol)){
            return "1" + encode(symbol, node.rightChild);
        }
        else{
            throw new NoSuchElementException();
        }
    }

    /**
     * decode the code
     * @param code the given code
     * @return the symbol of corresponding to the given code
     */
    public char decode(String code){
        HNode curr = root;
        for (char i : code.toCharArray()){
            if (i == '0'){
                if (curr.leftChild != null){
                    curr = curr.leftChild;
                }
                else{
                    return '\0';
                }
            }
            if (i == '1'){
                if (curr.rightChild != null){
                    curr = curr.rightChild;
                }
                else{
                    return '\0';
                }
            }
        }
        if (curr.symbolSet.length() == 1){
            return curr.symbolSet.charAt(0);
        }
        return '\0';
    }

    /**
     * write the individual bits of the given symbol using encoding to bitOutputStream
     * @param symbol the given symbol
     * @param stream the output stream
     * @return return true if written success
     */
    public boolean writeCode(char symbol, BitOutputStream stream){
        HNode curr = root;
        while (!curr.isLeaf()){
            if (curr.leftChild.contains(symbol)){
                try {
                    stream.writeBit(1);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                curr = curr.leftChild;
            }
            else if (curr.rightChild.contains(symbol)){
                try {
                    stream.writeBit(0);
                } catch (IOException e) {
                    e.printStackTrace();
                }
                curr = curr.rightChild;
            }
        }
        return true;
    }


    /**
     * read the next symbol of binary encoding individual bits from BitInputStream
     * and return the corresponding characters
     * @param stream the input stream
     * @return the corresponding character
     */
    public char readCode(BitInputStream stream){
        HNode curr = root;
        int b = -1;
        try {
            b = stream.readBit();
        } catch (IOException e) {
            e.printStackTrace();
        }
        while(b != -1){
            if (b == '0'){
                if (curr.leftChild != null){
                    curr = curr.leftChild;
                }
                else{
                    return '\0';
                }
            }
            if (b == '1'){
                if (curr.rightChild != null){
                    curr = curr.rightChild;
                }
                else{
                    return '\0';
                }
            }
        }
        if (curr.symbolSet.length() == 1){
            return curr.symbolSet.charAt(0);
```

```java
        }
        return '\0';
    }

    public HNode getRoot(){
        return root;
    }

}
```