```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.lang.reflect.Array;
import java.util.*;
import java.util.Map.Entry;


public class Decision_Tree
{

static int columb_count=0;
static int row_count=0;
static private BitSet columns;
static private BitSet rows;


static String targetValue;
static HashMap<String , Integer> tvalue= new HashMap<String, Integer>();

/*
static String value1=null;
static String value2=null;
static int val1_count=0;
static int val2_count=0;
*/
static File file=new File("C:\\Users\\Tanvi\\Desktop\\MCA IV sem\\Data 
Mining\\Decision_Tree\\data.txt");
static double Targetentropy=0.0;



    public static void main(String[] args) throws IOException
{

// TODO Auto-generated method stub
// fetch total number of rows and columbs in table.(done)
// get target class name, value1, value2 and entropy of that class.(done)
        FileReader fileReader = new FileReader(file);
        BufferedReader breader = new BufferedReader(fileReader);
        String line;
        int counter = 0;

        ArrayList<ArrayList<String>> data = new ArrayList<ArrayList<String>>();

        while ((line = breader.readLine()) != null)
        {
            String[] cols = line.split("\\|");

            ArrayList<String> k= new ArrayList<String>();
            Collections.addAll(k,cols);
            data.add(k);
            if (counter == 0)
            {
                columb_count = cols.length;
                targetValue = cols[cols.length - 1];
            }
            else
            {
                if(!tvalue.containsKey(cols[cols.length-1]))
                {
                    tvalue.put(cols[cols.length-1], 0);

                }
                /*
                if(value1==null)
                {
                    value1=cols[cols.length-1];
                }
                else
                {
                if(!cols[cols.length-1].equalsIgnoreCase(value1))
                {
                    value2=cols[cols.length-1];
                }
                }*/
            }
            counter++;
        }


row_count = counter;
rows= new BitSet(row_count);
columns= new BitSet(columb_count);
columns.set(columb_count-1);

//printvalue
System.out.println("val:"+tvalue);
for(ArrayList<String> k:data)
{
    for(String n:k)
    {
        System.out.print(" "+n);
    }
    System.out.println();
}


// fill attribute value and create the treenode based on that


TNode head=new TNode();
TNode kk=head;

ArrayList<Attribute> att= new ArrayList<Attribute>();

boolean once=true;
int counter2 =0;
while((kk=checktree(head)).getIndex()!=-2)
{
    System.out.println("\n\n\n *** start ***");
    if(counter2==3)
        System.exit(0);

    counter2++;


    System.out.println("tree :"+kk);
    // change data to point only to our required data.
    ArrayList<ArrayList<String>> data1= required_data(data,kk,att);
    System.out.println(data1);
    // calculate value of P and N
    calculate_val1_and_val2(data1);
    System.out.println("value :"+tvalue);
    // Calculate class Entropy
    ArrayList<String>tname =new ArrayList<String>();
    tname.addAll(tvalue.keySet());
    Targetentropy=entropy(tvalue.get(tname.get(0)), tvalue.get(tname.get(1)));
    System.out.println("target: "+Targetentropy);
    //calculate entropy,gain for each attribute
    att=attributevalue(data1,kk);

    //calculate entropy of each class provided in the att list
    att= calculate_attribute_entropy(att);
System.out.println(att);

```

```java
        // max gain index
        int index=0;
        double gain=0.0;
        for(Attribute a1:att)
        {
            if(gain<a1.getGain())
            {
                gain=a1.getGain();
                index=a1.getIndex();
            }
        }
        System.out.println("index:"+index);
        // set the index value in BITSET col
        BitSet temp=kk.getCol();
        temp.set(index);
        temp.set(columb_count-1);
        kk.setCol(temp);
        // create node for Index Attribute
        kk.setName(data.get(0).get(index));
        kk.setIndex(index);
        for(Attribute a1:att)
        {
            if(a1.getIndex()==index)
            {
                for(attvalue a2:a1.getValues())
                {
                    TNode a=new TNode();
                    a.setCol(temp);
                    a.setName(a2.getName());

                    if(a2.getEntropy()==0&&(a2.getVal1().size()==0||a2.getVal2().size()==0))
                    {
                        a.setLeaf(true);
                        a.setIndex(a2.getIndex());
                        if(a2.getVal1().size()==0&&a2.getVal2().size()!=0)
                        {
                        a.setOutput(tname.get(1));
                        }

                        if(a2.getVal1().size()!=0&&a2.getVal2().size()==0)
                        {
                            a.setOutput(tname.get(0));
                        }

                    }
                    kk.getNext().put(a2.getName(), a);

                }
            }
        }
    }


    ArrayList<TNode>que= new ArrayList<TNode>();
    que.add(head);
    int size=1;
    // print tree :
    while(que.size()!=0)
    {
        System.out.println("\n\n");
        int next=0;
        ArrayList<TNode> k= new ArrayList<TNode>();


        for(int i=0;i<size;i++)
        {
        for(String s:que.get(i).getNext().keySet())
        {
            k.add(que.get(i).getNext().get(s));
            next++;
        }
        que.addAll(k);
        }



        for(int i=0;i<size;i++)
        {
            if(que.get(0).isLeaf())
        System.out.print("\t"+que.get(0).getName()+"(output:"+que.get(0).getOutput()+")");
            else
        System.out.print("\t"+que.get(0).getName());


        que.remove(0);
        }
        size=next;


    }







    }



        private static ArrayList<ArrayList<String>>
        required_data(ArrayList<ArrayList<String>> data,TNode t,ArrayList<Attribute> att)
        {
            // TODO Auto-generated method stub
            if(t.getName()==null)
            {
                return data;
            }
            ArrayList<ArrayList<String>> data1 = new ArrayList<ArrayList<String>>();
            data1.add(data.get(0));
            String name=t.getName();
            for(Attribute a1:att)
            {
                for(attvalue a2:a1.getValues())
                {
                if(name==a2.getName())
                {
                    for(Integer k1:a2.getVal1())
                    {
                        data1.add(data.get(k1));
                    }
                    for(Integer k2:a2.getVal2())
                    {
                        data1.add(data.get(k2));
                    }
                }


                }

            }


            return data1;
        }



        private static void calculate_val1_and_val2(ArrayList<ArrayList<String>> data) {
            // TODO Auto-generated method stub
            //clear tvalue
            for(String k1:tvalue.keySet())
            {
            tvalue.put(k1, 0);
            }
            int counter=0;
            for(ArrayList<String> a1:data)
            {
                if(counter==0)
                {
                    counter++;
                    continue;
                }
                int value=tvalue.get(a1.get(columb_count-1));
                tvalue.put(a1.get(columb_count-1), value+1);

            }
        }



        private static ArrayList<Attribute>
        calculate_attribute_entropy(ArrayList<Attribute> att) {
            // TODO Auto-generated method stub
            for(Attribute a1:att)
            {

                ArrayList<String>tname =new ArrayList<String>();
                tname.addAll(tvalue.keySet());
                int val1_count=tvalue.get(tname.get(0));
                int val2_count=tvalue.get(tname.get(1));


                double entropy1=0.0;
                for(attvalue a2:a1.getValues())
                {

                    int val1=a2.getVal1().size();
                    int val2=a2.getVal2().size();

                    a2.setEntropy(entropy(val1, val2));

                    double k1=(double)val1+val2;

                    double k2=(double)val1_count+val2_count;

                    double k3=(k1/k2)*a2.getEntropy();

                    entropy1=entropy1+k3;
                }
                // attribute entropy
                a1.setClassentropy(entropy1);
                // gain
                a1.setGain((Targetentropy-entropy1));

            }


            return att;
        }



        public static TNode checktree(TNode head)
        {

            Queue<TNode> bb= new LinkedList<TNode>();
            bb.add(head);
            TNode kk=null;
            System.out.println("tree1");
            System.out.println("size: "+bb.size());
            while(bb.size()!=0)
            {
            kk=bb.poll();

                if(kk.getIndex()==-1)
                {
                    System.out.println("return kk");
                    return kk;
                }
                for(String s:kk.getNext().keySet())
                {
                    bb.add(kk.getNext().get(s));
                }
            }

            return new TNode(-2);
        }


        public static ArrayList<Attribute> attributevalue(ArrayList<ArrayList<String>>
        data, TNode t)
        {
            //if (!rows.get(counter))
                //continue;
            row_count=data.size();
            int counter =0;
            BitSet col=t.getCol();
            col.set(columb_count-1);
            ArrayList<Attribute> ans= new ArrayList<Attribute>();
            while (counter<columb_count)
            {
                if(col.get(counter))
                {
                counter++;
                continue;
                }
                Attribute a1= new Attribute();
                a1.setName(data.get(0).get(counter));
                a1.setIndex(counter);
                ArrayList<attvalue> values= new ArrayList<attvalue>();
                int counter2 =1;

                while(counter2<row_count)
                {

                    String target=data.get(counter2).get(columb_count-1);
                    String value=data.get(counter2).get(counter);
                    boolean a2=false;
                    // previously seen value in the respective attribute (counter2) (done)
                    ArrayList<String>tname =new ArrayList<String>();
                    tname.addAll(tvalue.keySet());
                    String value1=tname.get(0);
                    String value2=tname.get(1);
                    for(attvalue a:values)
                    {
                        if(a.getName().equalsIgnoreCase(value))
                        {
                            a2=true;
                            if(target.equalsIgnoreCase(value1))
                            {
                            a.getVal1().add(counter2);
                            }
                            if(target.equalsIgnoreCase(value2))
                            {
```

```java
                            a.getVal2().add(counter2);
                            }
                        }


                    }

                    // new value in tha attribute.+ add the new attvaluein values list (done)
                    if(a2==false)
                    {
                        attvalue a4= new attvalue();
                        a4.setName(value);
                        a4.setIndex(0);

                        if(target.equalsIgnoreCase(value1))
                        {

                        a4.getVal1().add(counter2);
                        }
                        if(target.equalsIgnoreCase(value2))
                        {

                        a4.getVal2().add(counter2);
                        }

                        values.add(a4);
                    }
                    counter2++;
                }

                a1.setValues(values);
                ans.add(a1);
                counter++;
                //System.exit(0);
            }

            /*
            for(Attribute at:ans)
            {
                System.out.println("\n\n\nname: "+at.getName());
                for(attvalue ki:at.getValues())
                {
                    System.out.println(ki);
                }
            }

            */

            System.out.println("attribute: "+ans);
            return ans;

        }




        public static double entropy(int a,int b)
        {
            if(a==0||b==0)
                return 0;
            if(a==b)
                return 1;

                double ans=0.0;
                int n=a+b;
                double a1=(double)a/(double)n;
                double a2=logb(a, n);
                ans=ans+(-1*a1*a2);
                a1=(double)b/(double)n;
                a2=logb(b, n);
                ans=ans+(-1*a1*a2);
                return ans;
        }

        public static double logb(double a, double b)
        {
            if (a == 0)
                return 0;
            double k1=Math.log(a)/Math.log(2); // gives log a base 2
            double k2=Math.log(b)/Math.log(2); // gives log n base 2
            return k1-k2; // log(a/b)=log a- log b
        }


    }
```