中南大學
CENTRAL SOUTH UNIVERSITY

Java 语言课程设计
实验报告

学生姓名	张天明
学　　号	8208190222
专业班级	计科 2006 班
指导教师	刘　嫔
学　　院	计算机学院
完成时间	2022.1.10

计算机学院

2021 年 6 月

目录

一、系统描述

(1) 基本要求

1. 从终端读入字符集,即 n 个字符和 n 个权值，建立哈夫曼树并将它存于文件 hfmTree 中。将已在内存中的哈夫曼树以直观的方式（比如树）显示在终端
2. 利用已经建好的哈夫曼树（如不在内存，则从文件 htmTree 中读入），对文件 ToBeTran 中的正文进行编码,然后将结果存入文件 CodeFile 中,并输出结果，将文件 CodeFile 以紧凑格式先是在终端上，每行 50 个代码。同时将此字符形式的编码文件写入文件 CodePrint 中。
3. 利用已建好的哈夫曼树将文件 CodeFile 中的代码进行译码，结果存入文件 TextFile 中，并输出结果。
4. 设计并实现人机交互友好的界面或菜单。

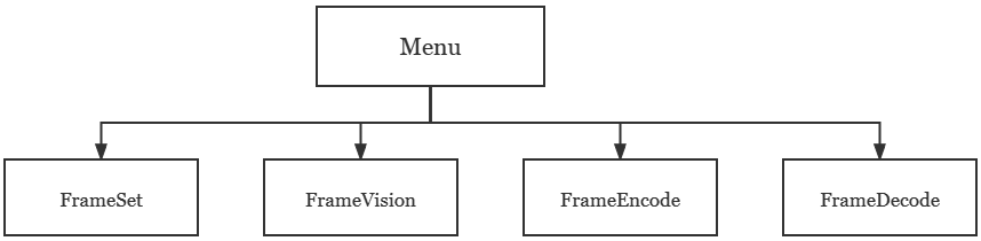
(2) 系统介绍

本系统主要实现哈夫曼树的构造以及可视化,并在哈夫曼树的基础上实现对指定文件的哈夫曼编码和解码。

二、功能模块

(1) 系统界面模块

1) 结构图



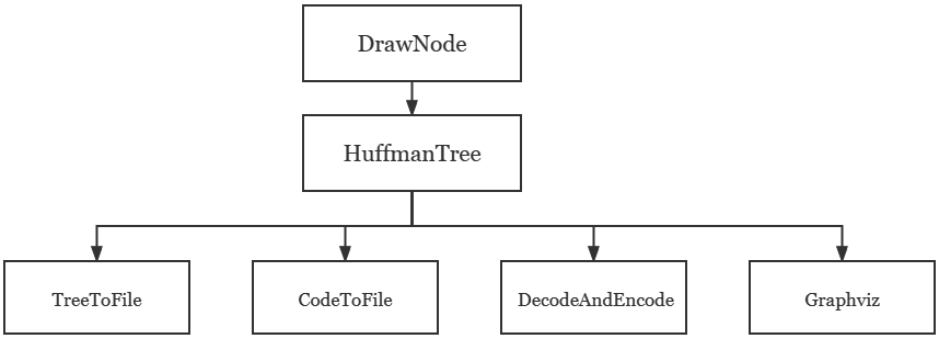
2) 功能描述

- Menu.java
  系统主界面，包含创建哈夫曼树、可视化、解码、译码四个功能的入口，点击对应的按钮可进入对应的页面进行操作
- FrameSet.java
  创建哈夫曼树界面，根据输入的结点值和权值创建哈夫曼树并显示创建信息

- FrameVision.java
  哈夫曼树可视化界面，将已经创建好的哈夫曼树图片展示出来
- FrameEncode.java
  哈夫曼编码界面，读取文件中的文本并显示，将编码后的文本显示出来并存入文件
- FrameDecode.java
  哈夫曼解码界面，读取解码前的文件，将文本显示出来，将解码后的文本显示出来并存入文件

(2) 系统功能模块

1) 结构图



2) 功能描述

- DrawNode.java
  哈夫曼树的结点类，创建哈夫曼树的树节点
- HuffmanTree.java
  哈夫曼树类，根据哈夫曼树的结点创建哈夫曼树
- TreeToFile.java
  哈夫曼树与文件之间的读写操作类，实现将哈夫曼树保存到文件中和从文件中读取哈夫曼树的操作
- CodeToFile.java
  哈夫曼编码与文件之间的读写操作类，实现将哈夫曼编码保存到文件中和从文件中读取哈夫曼编码的操作
- DecodeAndEncode.java
  哈夫曼编码与解码的实现操作类，将哈夫曼树中的结点按照规则进行编码和解码
- Graphviz.java
  哈夫曼树可视化的实现类，根据 Graphviz 的语法将哈夫曼树的结点之间的关系信息保存到文件中并使用脚本在终端运行

三、算法说明

(1) DrawNode 类

数据域：

```
    private String data;//结点数据
    private int weight;//结点权重
    private String code;//哈夫曼编码

    // private DrawNode parent;//父结点
    private DrawNode left;//左子树
    private DrawNode right;//右子树
```

方法：

结点类实现了 Comparable 接口，可以比较两个结点的权重，从而便于构建哈夫曼树时对节点进行选择。

```
    /*重写 Comparable 类的方法 */
    @Override
    public int compareTo(DrawNode o) {
        return o.weight - this.weight;//降序
    }
```

(2) HuffmanTree 类

数据域：

```
private DrawNode root;//根节点
```

方法：

✧ **根据结点链表构建哈夫曼树**

```
public void createHuffmanTree(ArrayList<DrawNode> nodes)
```

算法思路：

将结点链表进行排序，选出权值最小的两个结点，作为叶子结点，创建新节点，新生成的结点的权值为两个叶子结点的权值之和，将两个叶子结点从结点链表中删除，并加入新生成的叶子结点，直到结点链表中没有结点，说明所有结点都加入到哈夫曼树中。

代码实现：

```
    public void createHuffmanTree(ArrayList<DrawNode> nodes) {
        // 只要 nodes 数组中还有 2 个以上的节点
        while (nodes.size() > 1) {
            Collections.sort(nodes);//将结点数组降序排序
            DrawNode left = nodes.get(nodes.size()-1);
            DrawNode right = nodes.get(nodes.size()-2);

            //生成新节点，新节点的权值为两个子节点的权值之和
```

```
            DrawNode parent = new DrawNode(" ", left.getWeight() +
right.getWeight());

            //让新节点作为两个权值最小节点的父节点
            parent.setLeft(left);
            parent.setRight(right);

            //删除权值最小的两个节点
            nodes.remove(nodes.size()-1);
            nodes.remove(nodes.size()-1);

            //将新节点加入到集合中
            nodes.add(parent);
        }
        this.setRoot(nodes.get(0));
    }
```

✧ **使用广度优先遍历来遍历哈夫曼树并对结点进行哈夫曼编码，存入结点信息中**

```
public ArrayList<DrawNode> breadthFirst()
```

算法思路：

从根节点开始，每遍历一个结点，就将其加入到链表中，并将其左子树和右子树加入队列中，如果是左子树就在结点的编码中加“0”，如果是右子树就在结点的编码中加“1”，知道队列中元素为空。

代码实现：

```
public ArrayList<DrawNode> breadthFirst(){
        Queue<DrawNode> queue = new ArrayDeque<>();
        ArrayList<DrawNode> list = new ArrayList<>();

        if(this.getRoot()!=null){
            //将根元素加入队列
            queue.offer(this.getRoot());
            this.getRoot().getLeft().setCode(this.getRoot().getCode() + "0");
            this.getRoot().getRight().setCode(this.getRoot().getCode() + "1");
        }

        while(!queue.isEmpty()){
            //将该队列的队头元素加入到 list 中
            DrawNode p = queue.poll();
            list.add(p);

            //如果左子节点不为 null，将它加入到队列
            if(p.getLeft() != null){
                queue.offer(p.getLeft());
                p.getLeft().setCode(p.getCode()+"0");
```

```
            }

            //如果右子节点不为 null，将它加入到队列
            if(p.getRight() != null){
                queue.offer(p.getRight());
                p.getRight().setCode(p.getCode()+"1");
            }
        }
        return list;
    }
```

(3) TreeToFile 类

✧ **将哈夫曼树结点链表存放在指定的文件中**

```
public void huffmanTreeOutput(ArrayList<DrawNode> list,String filePath)
```

算法思路：

使用 `ObjectOutputStream` 输出流中的 `writeObject()` 方法实现对结点对象到文件的写入

代码实现：

```
public void huffmanTreeOutput(ArrayList<DrawNode> list,String filePath){
        try {
            File file = new File(filePath);
            FileOutputStream fos = new FileOutputStream(file);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(list);//将结点对象链表写入文件中
            System.out.println("Done");
            oos.close();
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

✧ **从文件中读取哈夫曼树**

```
public ArrayList<DrawNode> huffmanTreeInput(String path)
```

算法思路：

使用 `ObjectInputStream` 输出流中的 `readObject()` 方法实现对结点对象到文件的写入

代码实现：

```
    public ArrayList<DrawNode> huffmanTreeInput(String path) throws IOException,
ClassNotFoundException{
        ArrayList<DrawNode> NodeList;
        File file = new File(path);
        FileInputStream fis = new FileInputStream(file);
```

```
        ObjectInputStream ois = new ObjectInputStream(fis);
        NodeList = (ArrayList<DrawNode>)ois.readObject();//从文件中读取结点对象
        for (DrawNode drawNode : NodeList) {
            this.displayTreeNode(drawNode);
        }
        fis.close();
        ois.close();
        return NodeList;
    }
```

(4) CodeToFile 类

✧ 从文件中读取文本

```
public String TextFromFile (String textPath)
```

算法思路：

使用 FileInputStream 输入流中的 read() 方法实现文本的读取

代码实现：

```
    public String TextFromFile (String textPath) throws IOException{
        // "D:/学习/计算机/JAVA/Java 课设/Java 课程设计/Java 课程设计
/src/HuffmanTree/ToBeTran.txt"
        File file = new File(textPath);
        FileInputStream fis = new FileInputStream(file);
        byte[] data = new byte[(int)file.length()];
        fis.read(data);
        fis.close();
        return new String(data);
    }
```

✧ 将文本存入文件

```
public void HuffmanCodeToFile (String code ,String codePath)
```

算法思路：

使用 PrintStream 输出流对编码字符串进行输出

代码实现：

```
    public void HuffmanCodeToFile (String code ,String codePath) throws
FileNotFoundException{
        //"D:/学习/计算机/JAVA/Java 课设/Java 课程设计/Java 课程设计
/src/HuffmanTree/CodeFile.txt"
        File file = new File(codePath);
        FileOutputStream fos = new FileOutputStream(file);
        PrintStream ps = new PrintStream(fos);
        ps.println(code);
        ps.close();
    }
```

(5) DeCodeAndEnCode 类

✧ 对文本进行哈夫曼编码

```
public String HuffmanEncode (ArrayList<DrawNode> list,String text)
```

算法思路：

遍历文本字符串的每一个字符，在结点链表中寻找对应的数据值，匹配后将结点对应的哈弗曼编码取出

代码实现：

```
    public String HuffmanEncode (ArrayList<DrawNode> list,String text){
        StringBuilder code = new StringBuilder();
        for(int i = 0; i < text.length(); i++){
            for (DrawNode node : list) {
                if (text.charAt(i) == node.getData().charAt(0)) {
                    code.append(node.getCode());
                }
            }
        }
        return code.toString();
    }
```

✧ 将哈夫曼编码进行解码

```
public String HuffmanDecode (ArrayList<DrawNode> list,String codeAll)
```

算法思路：

将哈夫曼树结点链表中的有效结点（即数据不为空的结点）中的数据和对应的哈夫曼编码取出组成 Map 键值对，便于进行快速查找，查看要解码的文本是否以 Map 中的哈夫曼编码为开头，如果命中，则取出对应的数据，直到遍历整个编码。

代码实现：

```
    public String HuffmanDecode (ArrayList<DrawNode> list,String codeAll){
        StringBuilder text = new StringBuilder();
        Map <String,String> map = new HashMap<>();//建立编码与字符之间的键值对，方便使
用
        for (DrawNode node : list) {
            if (!node.getData().equals(" "))
                map.put(node.getCode(), node.getData());
        }

        while(codeAll.length() > 2){
            for(Map.Entry<String,String> e: map.entrySet()){//判断编码字符串是以哪一个
编码开头的
                String code = e.getKey();
                if(codeAll.startsWith(code)){
                    text.append(e.getValue());
```

```
                codeAll = codeAll.substring(code.length());
            }
        }
    }
    return text.toString();
}
```

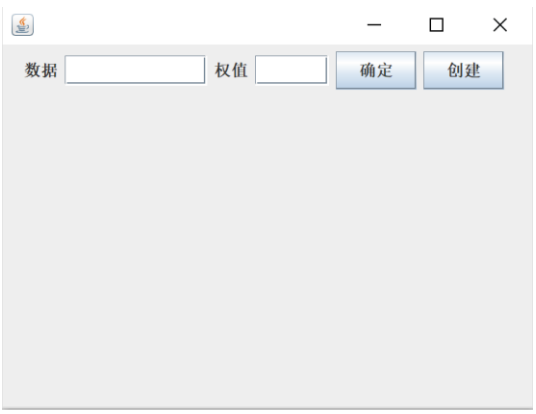
四、运行结果

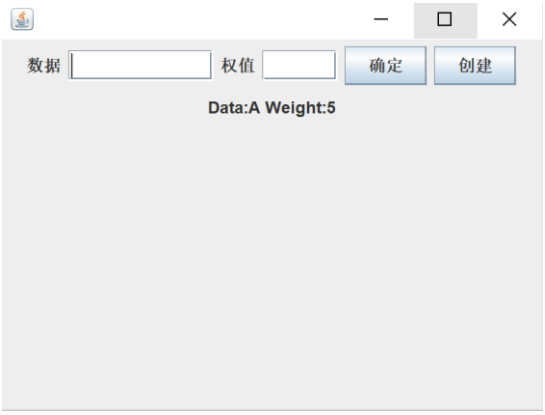
（1）系统界面

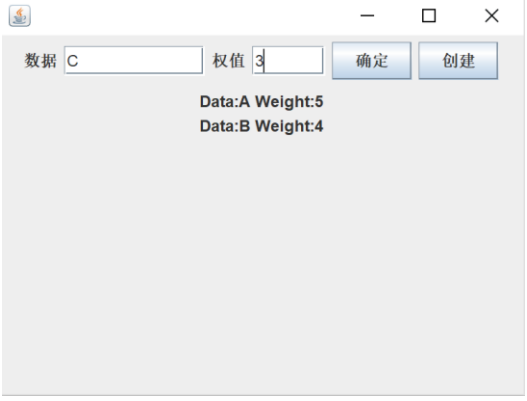
按对应的按钮选择对应的操作

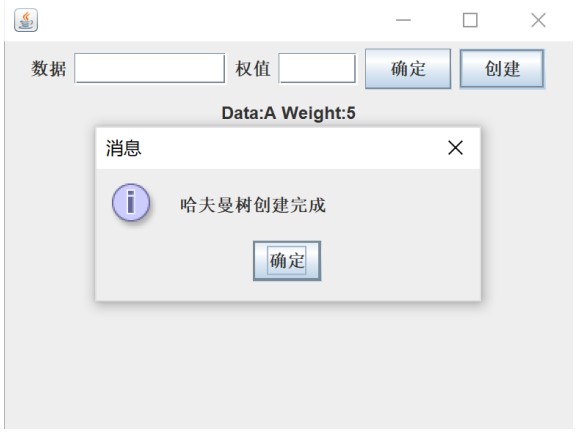
（2）创建哈夫曼树界面

按确定键创建一个结点，按创建表示结点输入完成，创建哈夫曼树。

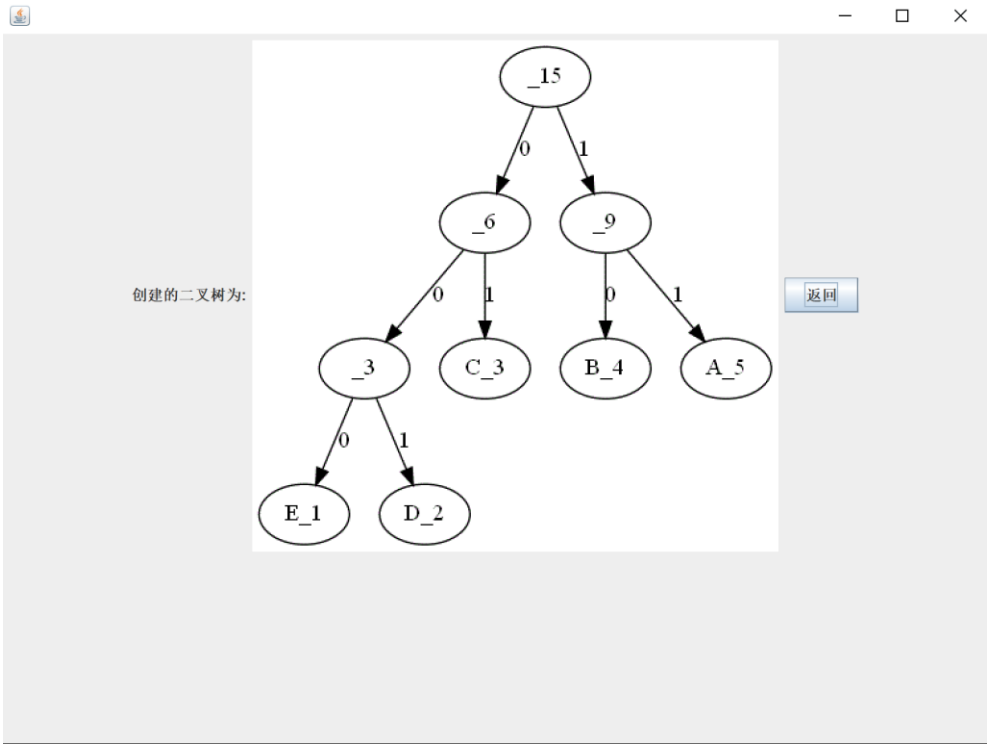

数据
权值
确定
创建


数据
权值
确定
创建
Data:A Weight:5


数据
C
权值
3
确定
创建
Data:A Weight:5
Data:B Weight:4

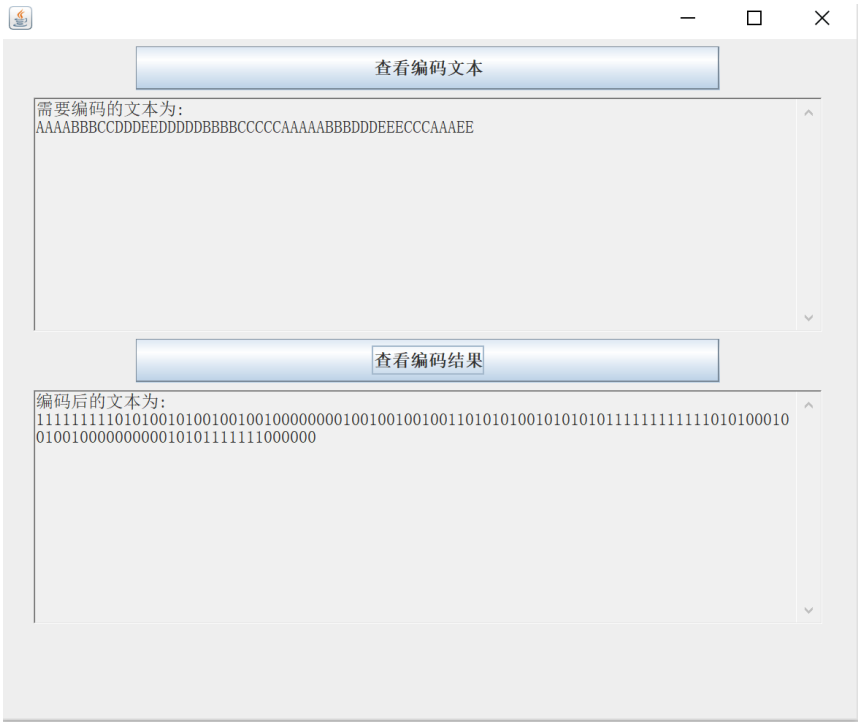

数据
权值
确定
创建
Data:A Weight:5
消息
哈夫曼树创建完成
确定

(3) 可视化哈夫曼树界面


创建的二叉树为:
_15
0
1
_6
_9
0
1
0
1
_3
C_3
B_4
A_5
0
1
E_1
D_2
返回

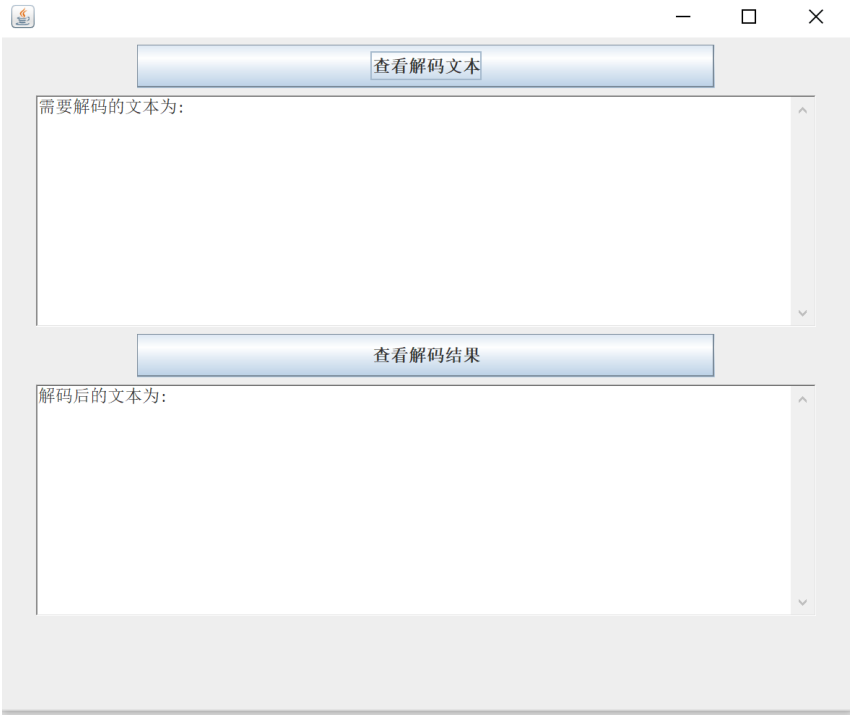
（4）编码界面

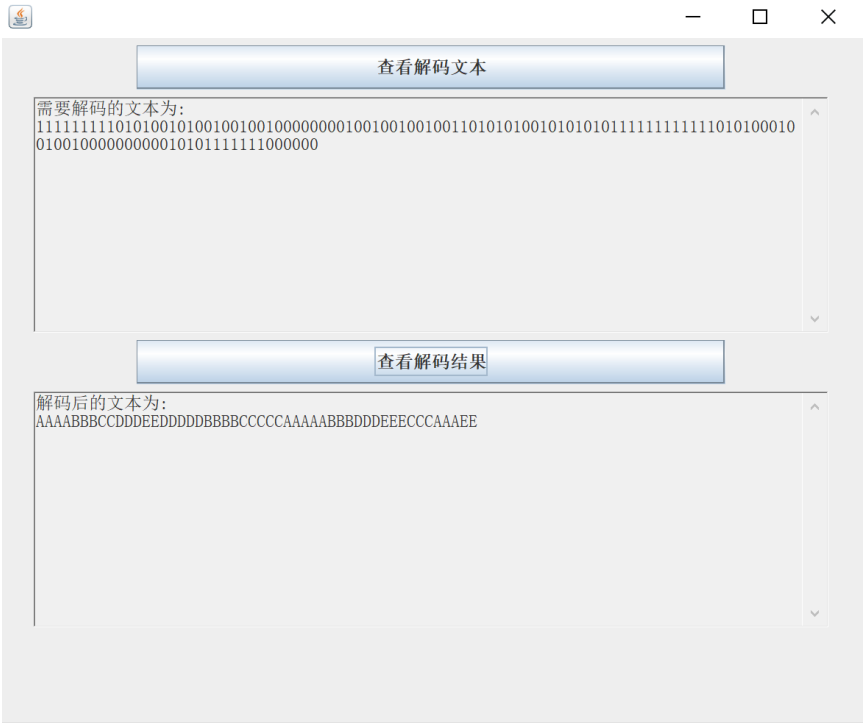
点击“查看编码文本”将 ToBeTran.txt 文件中的文本显示出来
点击“查看编码结果”将存入 CodeFile.txt 文件中的编码显示出来



(5) 解码界面

点击“查看解码文本”将 CodeFile.txt 文件中的文本显示出来
点击“查看解码结果”将存入 TextFile.txt 文件中的编码显示出来





五、课程设计总结

(1) 问题与解决方法

1. 如何将哈夫曼树可视化？
   解决过程：起初想到了利用 2D 画笔在面板上画图的方式，通过画出圆圈和连线的方式画出二叉树，但是实现过程中要考虑圆圈的位置和连线的两个点的位置等一系列的问题，实现过程较为困难，难度较大，于是转换方向，上网查找资料后发现一个可以用来可视化二叉树的开源软件 Graphviz,它使用 dot 语言在终端运行的方式生成二叉树，有特定的语法，只要给出结点之间的信息并使用规定的语法，就可以使用命令行在中断运行脚本，从而得到可视化的二叉树。考虑该方法可行性较高，于是将结点信息和对应的边的编码添加到文本中使用 run.exec()在终端运行，并将生成的二叉树作为图片保存起来。
2. 如何将二叉树保存到文件中？
   解决过程:起初想到了使用对象的输出流 ObjectOutputStream,但是第一次编译后报错，上网查找解决方式后得知，需要把 writeObject()的对象进行序列化，于是将 DrawNode 结点实现 Serializable 接口后成功。
3. 如何实现对象之间的比较？
   解决过程：由于构建哈夫曼树的过程中需要找出结点中权值最小的两个作为叶子结点，于是需要对 DrawNode 对象根据权值大小进行比较。上网查找后发现实现 Comparable 接口是比较好的方法，于是将 DrawNode 类实现 Comparable 接口并重写 CompareTo 方法得以实现。
4. 如何实现界面和交互？
   解决过程：使用 Java 的 Swing 库来实现界面，在实现页面时要考虑很多问题，如组件的大小、放置顺序、位置，页面的布局方式，事件的监听以及处理方式等等。这些都通过上网查找相似的例子学习相关的知识来实现。

(2) 课程设计收获

通过本次高级语言程序设计，我加强了 Java 语言在实际项目工程中的运用能力，加深了对类与对象的特性的理解，并能在实际运用中使用类与对象的思想来解决问题。在开发哈夫曼编码的系统过程中遇到了很多之前没有遇到过的问题，接触到了一些未曾涉及过的知识，例如界面的开发，收获知识颇丰。在学习一门语言时，除了要掌握基本的语法知识，还要思考这门语言存在的意义，有什么特性，和其他语言有哪些不同之处，在实际解决问题过程中加深理解，才更利于对知识的掌握和编程能力的提高。对于没有学过的知识，要善于利用网络，借助其他工具协助完成任务。

(3) 程序提高

界面设计比较简单，既要考虑到人机交互时便于用户的操作，也要考虑到界面的美观，应该更多考虑对于错误输入的处理和引导，使程序更加健壮。

六、附录

程序清单

HuffmanTree 目录
- DrawNode.java
- HuffmanTree.java
- TreeToFile.java
- CodeToTile.java
- Graphviz.java
- DeCodeAndEnCode.java
- frame 目录:
    - Menu.java
    - FrameSet.java
    - FrameVision.java
    - FrameDecode.java
    - FrameEncode.java