Assignment 3
Huffman Encoding

Due Wednesday October 22

Quiz 5 Content

- Binary Trees
  - Algorithms with diagrams and writing code
- Expression Trees
  - Algorithms with diagrams and writing code
- Huffman coding
  - Algorithms with diagrams
- Binary Search Trees
  - Algorithms with diagrams

Huffman Coding

- Implement a Huffman coding application
- The application will implement encoding and decoding operations
- The input to encoding will be text files containing characters in the range 0-127
- The input to decoding will be a previously encoded file

Priority Queue

```
public class PriorityQueue {
//A priority queue in which the keys are Integers and the associated data is any object
//the queue must be implemented as a binary heap

        public PriorityQueue(int qsize) {

        }

        public Object deleteMin() {
        //PRE: !empty()
        }

        public Integer getMinKey() {
        //PRE: !empty()
        }


        public boolean empty() {
        }
```

Priority Queue

```
        public Object getMinData() {
        //PRE: !empty()
        }

        public boolean full() {
        }

        public void insert(Integer k, Object d) {
        //PRE !full()
        }

        public int getSize() {
        }

}
```

BitInputStream

```
import java.io.*;

public class BitInputStream {
	DataInputStream in;

	public BitInputStream(String infile)  throws IOException {
		in = new DataInputStream(new FileInputStream(infile));
		//more initialization will be required
	}
	public int readInt()  throws IOException {
	}
	public String readString() throws IOException  {
	}
	public int readBit()  throws IOException {
	}
	public void close() throws IOException {
		in.close();
	}
}
```

BitOutputStream

```
import java.io.*;

public class BitOutputStream {
        DataOutputStream out;

        public BitOutputStream(String filename) throws IOException {
                out = new DataOutputStream(new FileOutputStream(filename));
                //more initialization is needed
        }
        public void writeInt(int i)  throws IOException {
        }
        public void writeString(String s) throws IOException {
        }
        public void writeBits(String s)  throws IOException {
        }
        public void close()  throws IOException {
                //add code to write the last byte when necessary
                out.close();
        }
}
```

HuffmanASCIICompression

```
import java.io.*;
import java.util.*;
public class HuffmanASCIICompression {

    class Node {
        Node left;
        char ch;
        Node right;

        Node(Node L, char c, Node r) {
            left = L;
            ch = c;
            right = r;
        }
    }
    Node root;
    int frequency[];
    int totalChars;
    String encodings[];
    public HuffmanASCIICompression() {
    //depending on your implementation you might not need any initialization
    }
```

HuffmanASCIICompression

```
        //You will need to implement many private methods to do work needed for
        //encode and decode
        public void encode(String infile, String outfile)throws IOException {
        }

        public void decode(String infile, String outfile)throws IOException {
        }

        public static void main(String args[]) throws IOException {
                HuffmanASCIICompression h = new HuffmanASCIICompression();
                h.encode(args[0], args[1]);
                h.decode(args[1], args[0]+"_new");
        }
}
```

Assignment Submission

- You must email me the following files
  - HuffmanASCIICompression.java
  - PriorityQueue.java
  - BitInputStream.java
  - BitOutputStream.java
  - One small text file for which you application worked

Some useful Java classes and methods

- BufferedReader
  - read
- DataOutputStream
  - writeInt
  - writeUTF
  - writeByte
- DataInputStream
  - readInt
  - readUTF
  - readUnsignedByte

Some useful Java classes and methods

- PrintWriter
  - print
- StringBuilder
  - append
  - setCharAt
  - substring
  - setLength

Other comments

- When you encode do not build a single string of 0s and 1s and write bytes at the end of the process. Each time you have a group of 8 bits write the byte to the file.
- Since the data in the priority queue is an Object when you call deleteMin you will have to cast the result to a Node
- char a just small int so you can use them to index arrays
- I will post some example large files you should use to test you program. Do NOT send me these files or the encoded version of these files when you submit the assignment
- See class discussion for how to write the huffman tree to the decoded file