# **UML Class Diagrams:**

## FrequencyCounter

elements (linked list) numElements

This class handles the building and manipulation of a frequency table that stores the frequency with which a char appears in a file. The frequency and letters are stored in elements as a HuffNode data type.

## HuffmanTree

root (linked based)

This class handles the tree operations of the Huffman tree, specifically it provides operations to build/rebuild the Huffman tree, to compute the Huffman code of a given character, and to obtain the character given a Huffman code.

#### **HuffNode**

letter frequency

This class stores the necessary data for Huffman's algorithm, specifically each char/letter and its frequency.

### **InBitStream**

buf (char buffer) numBits in (istream)

This class handles the operation of read a sequence of bits from the given istream by using the binary operators.

### **OutBitStream**

buf (char buffer) numBits out (ostream)

This class handles the operation of writing a sequence of bits to the given ostream by using the binary operators

# **PriorityQueue**

head (linked list)

This class aids the Huffman algorithm in which two of the Huffman trees with the least frequency are joined. This class stores each Huffman tree in order of the roots frequency (small to large).

#### Huff

FrequencyCounter of Trees
PriorityQueue of Trees
HuffmanTree

This is the main driver program for the huffman compression application. This code handles the client interaction for both compressing to .huf files and decompressing.

# **Compressed File Format:**

This program will compress data into a compressed .huff file, these files have the format seen below with each section also defined below.

**Compressed Huffman Tree:** This is used to rebuild the Huffman tree when decompressing. This tree is compressed by traversing the tree in order and using the following technique: if current node is a leaf output '1' bit followed by the nodes letter character, else if current node is not a leaf then output '0'. For example, the Huffman tree built of the input string "that's the way it is" would take the form 0001h1a01s1i00001'1e01w1y01t1<space> where <space> is the empty space character.

**Compressed Data:** This is the actual in file data compressed using Huffman's algorithm.

**EOF Marker:** This tells the program when to stop reading the compressed data. Only bytes can be written to a file so using Huffman's algorithm which stores bits will naturally have some garbage at the end of the compressed data. The EOF marker allows us to know when we have reached the actual end of the data and ignore the garbage.

