HuffmanTree

- Contains the encoding algorithm
- Builds the tree

Private:

- PQ<HuffNode*> leaves → I had to wrap the HuffNode in a wrapper to compare them.
- HuffNode* root

- uses tree to encode & decode.

Public:

+ HuffTree(FreqCount fc)
+ HuffTree(InputBitstream i)
+ encode(ifstream input, outputBitStream out)
+ decode(inputBitstream in, ofstream output)

outputBitStream

- lets you write bits 1-by-1

Private:

- ofstream file
- int positionInByte
- char currentByte.

Public:

+ outputBitstream(char* file)
+ putABit(bool bit)
+ putAChar(char c)
+ putAnInt(int c)

InputBitStream

Private:
- ifstream file
- int positionInByte
- char currentByte

Public:
+ InputBitStream (char * file)
+ getABit (): bool
+ getAChar (): char
+ getAnInt (): int

- lets you read bits 1-by-1

HuffNode

Private:
- int frequency
- HuffNode * lchild
- HuffNode * rchild
- char containedChars

Public:
+ HuffNode (HuffNode * child1, HuffNode * child2)
+ HuffNode (int freqCount, char character)

- nodes of Huffman Tree

FrequencyCounter

- counts the frequency of character inputs

Private:
- LetterFrequencies

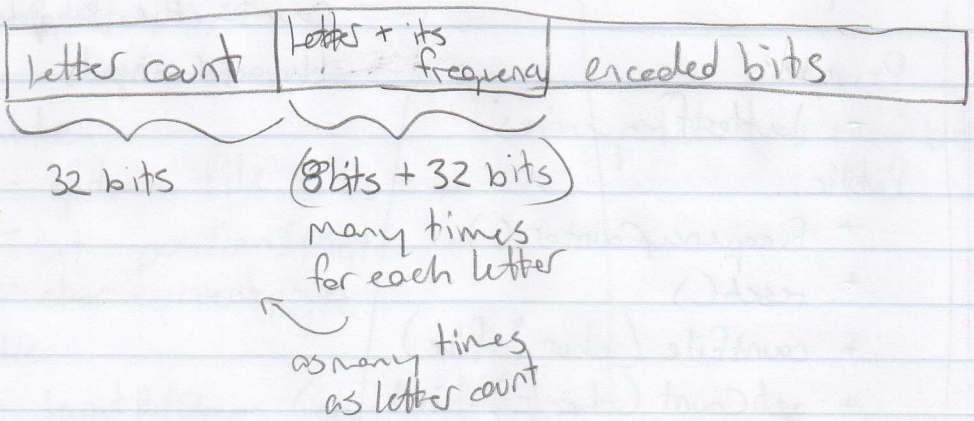
Public:
+ FrequencyCounter()
+ reset()
+ countFile(char * file)
+ getCount(char c): int

PriorityQueue

- A Priority Queue.

Private:
- Node<T> * head
- int elementCount

Public:
+ PriorityQueue
+ ~PriorityQueue
+ getElementCount : int
+ isEmpty : bool
+ enqueue(T& newElement) : bool
+ dequeue : bool
+ peek : T
+ import(PriorityQueue<T> & pq)


letter count
Letter + its frequency
encoded bits
32 bits
(8 bits + 32 bits)
many times for each letter
as many times as letter count