Contents

Description 1

Structure 2

Class Diagram 2

Executor 3

HuffmanNode 3

MinPriorityQueue 3

HuffmanCodedData 3

Activity Diagram 4

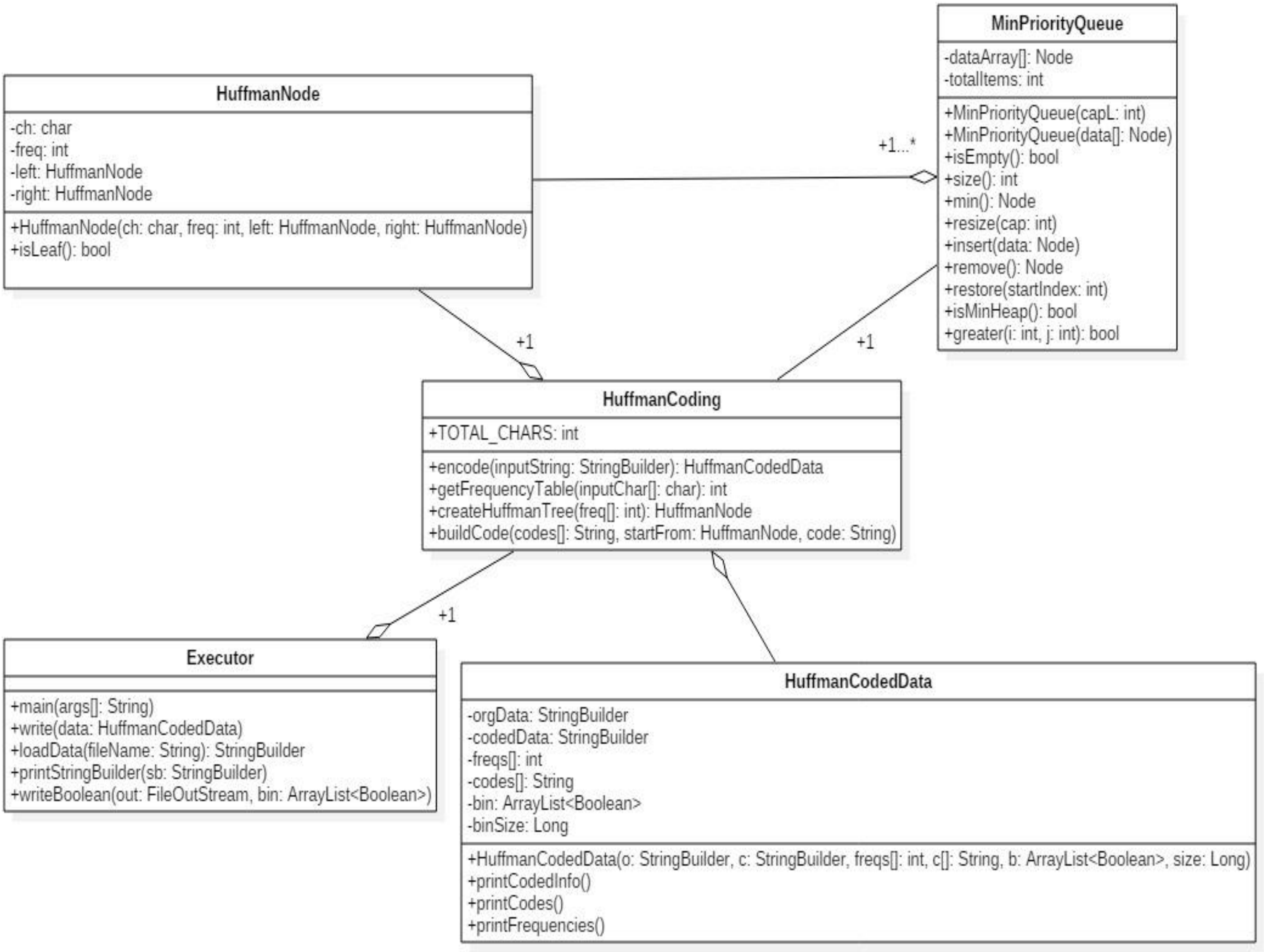
Description

This program is designed to compress a given text file using Huffman Coding algorithm. While starting, it takes file name of a text file, and then using Huffman Coding technique, it convert that text file into binary file named as “output.dat”. This program can recognize & Encodes up to 256 different characters having ASCII from 0 to 255.

Structure

The overall structure and activities in this program have been illustrated in below diagrams.

Class Diagram


MinPriorityQueue
-dataArray[]: Node
-totalItems: int
+MinPriorityQueue(capL: int)
+MinPriorityQueue(data[]: Node)
+isEmpty(): bool
+size(): int
+min(): Node
+resize(cap: int)
+insert(data: Node)
+remove(): Node
+restore(startIndex: int)
+isMinHeap(): bool
+greater(i: int, j: int): bool
HuffmanNode
-ch: char
-freq: int
-left: HuffmanNode
-right: HuffmanNode
+HuffmanNode(ch: char, freq: int, left: HuffmanNode, right: HuffmanNode)
+isLeaf(): bool
+1...*
+1
+1
HuffmanCoding
+TOTAL_CHARS: int
+encode(inputString: StringBuilder): HuffmanCodedData
+getFrequencyTable(inputChar[]: char): int
+createHuffmanTree(freq[]: int): HuffmanNode
+buildCode(codes[]: String, startFrom: HuffmanNode, code: String)
+1
Executor
+main(args[]: String)
+write(data: HuffmanCodedData)
+loadData(fileName: String): StringBuilder
+printStringBuilder(sb: StringBuilder)
+writeBoolean(out: FileOutStream, bin: ArrayList<Boolean>)
HuffmanCodedData
-orgData: StringBuilder
-codedData: StringBuilder
-freqs[]: int
-codes[]: String
-bin: ArrayList<Boolean>
-binSize: Long
+HuffmanCodedData(o: StringBuilder, c: StringBuilder, freqs[]: int, c[]: String, b: ArrayList<Boolean>, size: Long)
+printCodedInfo()
+printCodes()
+printFrequencies()

Executor

This is the main class. It acts like a Driver class for Encoding. It takes the text file name as command parameter, load text from that file, try to encode it using “HuffmanCoding” class, print all data to Standard Output Screen. Along these operations, after encoding, it also create another file named as “Output.dat” to contains binary coded data of that input text.

HuffmanNode

It contains info for a specific character during the creation of Huffman Tree in PriorityQueue. It contains information like character, its frequency, and info of its left & right nodes

MinPriorityQueue

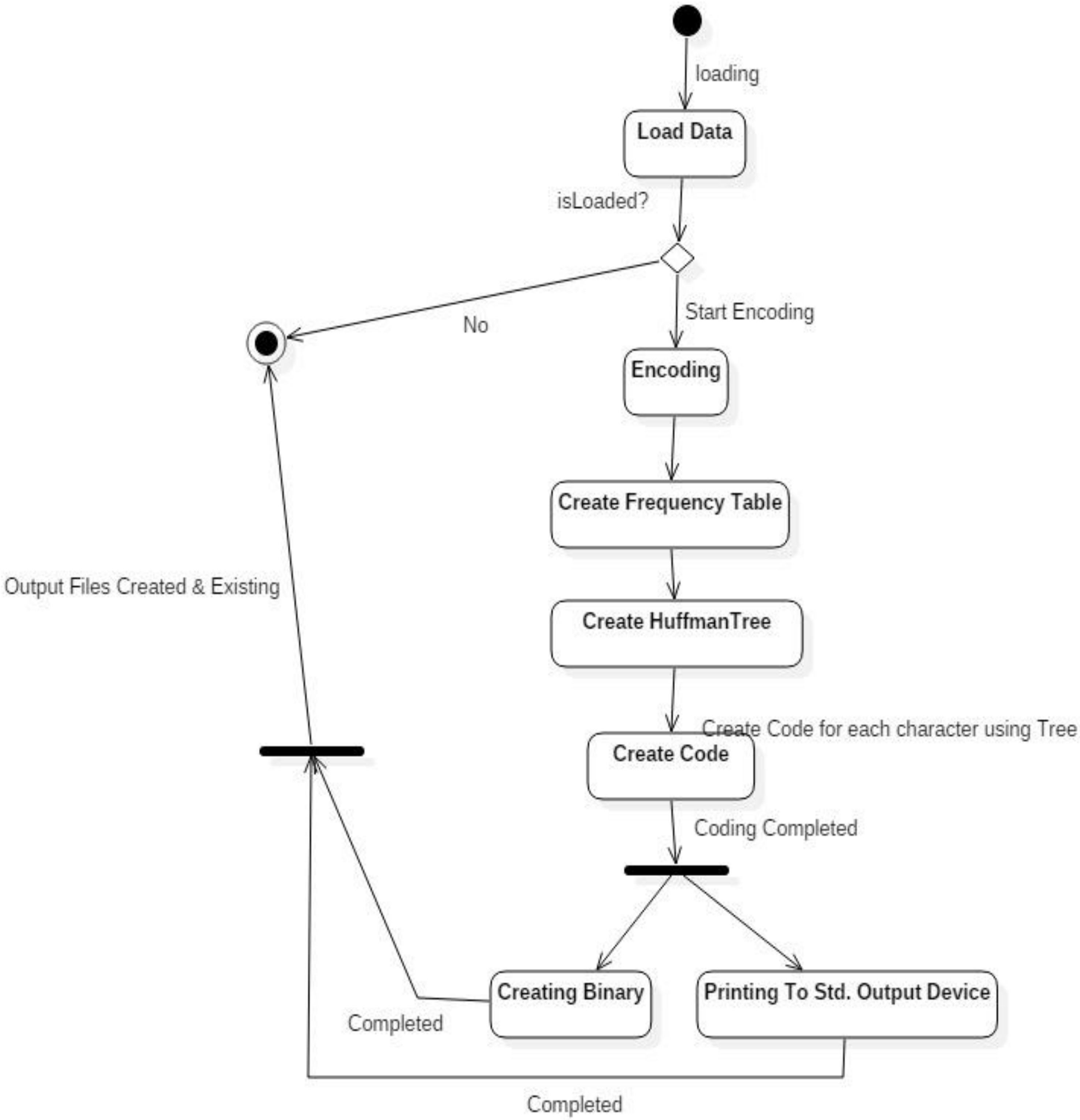
It is a Priority Queue, that is mainly used to create Huffman Tree using frequencies of characters. It takes HuffmanNode object as unit, which contains different information about a specific characters.

HuffmanCodedData

After converting input text into coded text using Huffman Encoding, all related information is stored into another class named as HuffmanCodedData. If contains different information about data like original data, frequencies of its different characters, codes used for each character of that text, encoded text and binary data. It acts just like a Wrapper class or as DTO (Data Transferring Object).

Activity Diagram

Below diagram illustrate complete operation of this program in abstract.


loading
Load Data
isLoaded?
No
Start Encoding
Encoding
Create Frequency Table
Create HuffmanTree
Create Code for each character using Tree
Create Code
Coding Completed
Output Files Created & Existing
Creating Binary
Printing To Std. Output Device
Completed
Completed