# UML diagram

# <<interface>> Coding + getCodingTable(): return Map<Character, String> + startEncode(): return void + startDecode(): return void + getDecodeContent(): return String AbstractCoding - frequencyTable: Map<Character, Integer> - codingTable: Map<Character, String> - rootNode: Node - pq: Queue<Pair> - symbolSets: int - decodeContent: String + buildingPriorityQueue(): return void + encodeHuffmanTree(): return void + decodeContent(String): return void - buildCodingTable(int, String): return void EncodingAndDecoding

### Driver

- message: String
- isTerminal: boolean
- ft: FrequencyTable
- huffmanCoding: Coding
- base: int
- file: File
- + gettingInput(): return void
- + initializeCodingTable(): return void
- + encodingPhase(): return void
- + decodingPhase(): return void

## Frequency Table

- ftMap: Map<Character, Integer>
- + countFrequency(String): return void
- + getFrequencyTable(): return Map<Character, Integer>

#### Node

- children: List<Node>
- symbolID: String
- isWord: boolean
- + setSymbolID(String): return void
- + getSymbolID(): return String
- + getChildren(): return List<Node>
- + getIsWord(): return boolean

## Test Case:

- 1. Initializing Driver:
  - // Choose input type: terminal or file(read or write)
  - // Get input message and initialize frequency table
  - // Choose the symbolSets (2~16)

FrequencyTable ft = new FrequencyTable(input)

2. Encode the message:

Coding huffmanCoding = new EncodingAndDecoding(ft.getFrequencyTable(), base) huffmanCoding.startEncode()

- 3. Decode the password:
  - // Get input password and pass to huffmanCoding huffmanCoding,startDecode(password)

## 4. Test:

FrequencyTable ft = new FrequencyTable("SHE SELLS SEA SHELLS BY THE SEA SHORE")

```
Coding huff = new EncodingAndDecoding(ft.getFrequencyTable(), 2)
Map<Character, String> expectedCT = new HashMap<>();
expectedCT.put('', "110");
expectedCT.put('A', "0001");
expectedCT.put('B', "00100");
expectedCT.put('R', "00110");
expectedCT.put('S', "10");
expectedCT.put('T', "00111");
expectedCT.put('E', "111");
expectedCT.put('H', "010");
expectedCT.put('Y', "0000");
expectedCT.put('L', "011");
expectedCT.put('O', "00101");
huff.startEncode();
assertEquals(expectedCT, huff.getCodingTable());
huff.startDecode("01000101110000"); // HOLY
assertEquals("HOLY", huff.getDecodeContent());
// Test Invalid Password(expected = IndexOutOfBoundsException.class)
huff.startEncode();
huff.startDecode("2");
// Test Invalid Password2(expected = StringIndexOutOfBoundsException.class)
huff.startEncode();
huff.startDecode("1101")
```