ASSIGNMENT-2

```
import java.util.*; // Import utilities like Scanner, List, PriorityQueue, etc.

// ✅ HuffmanCoding class (main class — must match filename:
HuffmanCoding.java)
public class HuffmanCoding {

    // ---------- Node CLASS ----------
    // Represents a single node in the Huffman tree
    static class Node {
        char ch;       // Character
        int freq;      // Frequency of that character
        Node left;     // Left child
        Node right;    // Right child

        Node(char ch, int freq) {   // Constructor
            this.ch = ch;
            this.freq = freq;
            this.left = null;
            this.right = null;
        }
    }

    // ---------- BUILD HUFFMAN TREE ----------
    // Builds a Huffman tree based on characters and frequencies
    public static Node buildHuffmanTree(List<Character> chars, List<Integer> freqs) {
        // Create a priority queue (min-heap) ordered by frequency
        PriorityQueue<Node> pq = new PriorityQueue<>(Comparator.comparingInt(n ->
n.freq));

        // Step 1: Insert all characters as leaf nodes into the queue
        for (int i = 0; i < chars.size(); i++) {
            pq.add(new Node(chars.get(i), freqs.get(i)));
        }

        // Step 2: Build the Huffman tree
        while (pq.size() > 1) {
            // Remove two nodes with smallest frequencies
            Node left = pq.poll();
            Node right = pq.poll();

            // Create a new internal node with frequency = left + right
            Node newNode = new Node('\0', left.freq + right.freq);
            newNode.left = left;
            newNode.right = right;
            // Add new node back to the queue
            pq.add(newNode);
        }

        // Step 3: Remaining node is the root of the Huffman tree
        return pq.peek();
    }

    // ---------- PRINT HUFFMAN CODES ----------
    // Recursively prints character and its Huffman code
    public static void printCodes(Node root, String code) {
        if (root == null) return;

        // Leaf node → print character and its code
        if (root.left == null && root.right == null && root.ch != '\0') {
            System.out.println(root.ch + ": " + code);
        }

        // Traverse left as '0' and right as '1'
        printCodes(root.left, code + "0");
        printCodes(root.right, code + "1");
    }

    // ---------- MAIN METHOD ----------
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in); // For input

        // Step 1: Take number of characters
        System.out.print("Enter the number of characters: ");
        int n = sc.nextInt();

        // Step 2: Take character and frequency input
        List<Character> chars = new ArrayList<>();
        List<Integer> freqs = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            System.out.print("Character[" + i + "]: ");
            char c = sc.next().charAt(0);
            chars.add(c);

            System.out.print("Frequency[" + i + "]: ");
            int f = sc.nextInt();
            freqs.add(f);
        }
        // Step 3: Build Huffman tree
        Node root = buildHuffmanTree(chars, freqs);

        // Step 4: Print Huffman codes
        System.out.println("\nHuffman Codes:");
        printCodes(root, "");

        sc.close();
    }
}
```

OUTPUT:

```
Enter the number of characters: 4
Character[0]:a
Frequency[0]:1
Character[1]:b
Frequency[1]:9
Character[2]:c
Frequency[2]:10
Character[3]:d
Frequency[3]:3

Huffman Codes:
c: 0
a: 100
d: 101
b: 11


=== Code Execution Successful ===
```