Practical No -23

Aim:- Write a Program to find The No of Words in a Trie

Example Execution

1. Inserting Words:
    - Insert "hello":
        - Create nodes for 'h', 'e', 'l', 'l', 'o' and mark the last node as the end of the word.
        - Increment wordCount to 1.
    - Insert "world":
        - Create nodes for 'w', 'o', 'r', 'l', 'd' and mark the last node as the end of the word.
        - Increment wordCount to 2.
    - Insert "hi":
        - Create nodes for 'h' and 'i' and mark the last node as the end of the word.
        - Increment wordCount to 3.
    - Insert "her":
        - Create nodes for 'e' and 'r' (the 'h' node already exists) and mark the last node as the end of the word.
        - Increment wordCount to 4.
    - Insert "hero":

        - Create a node for 'o' (the 'h', 'e', and 'r' nodes already exist) and mark the last node as the end of the word.
        - Increment wordCount to 5.

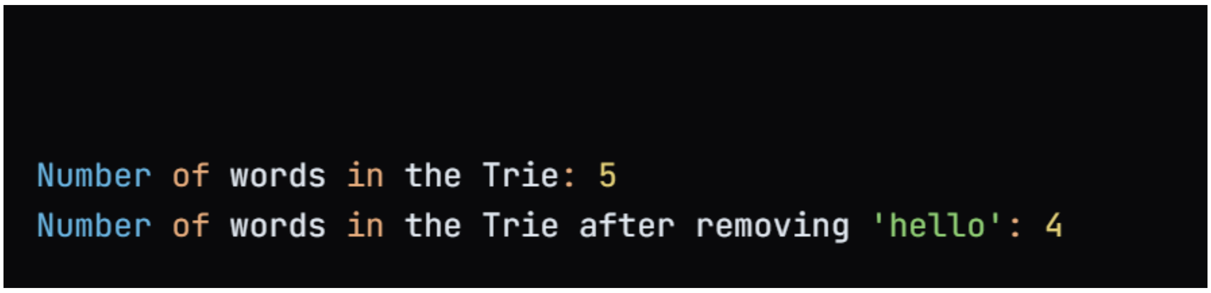
2. Counting Words:
    - After inserting all words, the program outputs the total number of words in the Trie, which is 5.
3. Removing a Word:
    - Remove "hello":
        - Traverse to the end of the word and unset the isEndOfWord flag.
        - Decrement wordCount to 4.
        - Check if the nodes can be deleted (in this case, they cannot be deleted because other words share the prefix).
4. Final Count:
    - After removing "hello

Program :- .

```
import java.util.HashMap;
class TrieNode {
    HashMap<Character, TrieNode> children;
    boolean isEndOfWord;
    public TrieNode() {
        children = new HashMap<>();
        isEndOfWord = false;
    }
}

class Trie {
    private TrieNode root;
    private int wordCount; // To keep track of the number of words

    public Trie() {
        root = new TrieNode();
        wordCount = 0; // Initialize word count to zero
    }

    // Insert a word into the Trie
    public void insert(String word) {
        TrieNode node = root;
        boolean isNewWord = false; // Flag to check if it's a new word

        for (char ch : word.toCharArray()) {
            if (!node.children.containsKey(ch)) {
                node.children.put(ch, new TrieNode());
                isNewWord = true; // A new character means a new word
might be added
```

```
        }
        node = node.children.get(ch);
    }

    if (!node.isEndOfWord) {
        node.isEndOfWord = true; // Mark the end of the word
        wordCount++; // Increment the word count
    }
}

// Search for a word in the Trie
public boolean search(String word) {
    TrieNode node = root;
    for (char ch : word.toCharArray()) {
        if (!node.children.containsKey(ch)) {
            return false; // Word not found
        }
        node = node.children.get(ch);
    }
    return node.isEndOfWord; // Return true if it's a valid word
}

// Remove a word from the Trie
public void remove(String word) {
        remove(root, word, 0);
    }

    private boolean remove(TrieNode node, String word, int depth) {
        if (depth == word.length()) {
            if (!node.isEndOfWord) {
                return false; // Word not found
            }
            node.isEndOfWord = false; // Unmark the end of the word
            wordCount--; // Decrement the word count
            return node.children.isEmpty(); // If true, delete this node
        }

        char ch = word.charAt(depth);
        TrieNode childNode = node.children.get(ch);
        if (childNode == null) {
            return false; // Word not found
        }

        boolean shouldDeleteCurrentNode = remove(childNode, word,
depth + 1);
        if (shouldDeleteCurrentNode) {
            node.children.remove(ch);
            return node.children.isEmpty(); // If true, delete this node
        }

        return false;
    }

    // Get the number of words in the Trie
    public int getWordCount() {
        return wordCount;
    }

    public static void main(String[] args) {
        Trie trie = new Trie();
        trie.insert("hello");
        trie.insert("world");
        trie.insert("hi");
        trie.insert("her");
        trie.insert("hero");

        System.out.println("Number of words in the Trie: " +
trie.getWordCount()); // Output: 5

        trie.remove("hello");
        System.out.println("Number of words in the Trie after removing
'hello': " + trie.getWordCount()); // Output: 4
```

```
    }
}
```



```
Number of words in the Trie: 5
Number of words in the Trie after removing 'hello': 4
```