Day 10: Reverse Words in a String

Abhinav Yadav

"The best error message is the one that never shows up."
— Thomas Fuchs

---

1 Introduction

Reversing words in a string while maintaining their original order is a common string manipulation problem. Unlike reversing the entire string, this problem focuses on reversing individual words while keeping their positions intact. This document outlines the efficient approach and code solution for this problem.

2 Problem Statement

Problem: Reverse words in a string while maintaining their order. **Hint:** Use a two-pointer approach to reverse individual words. **Edge Case:** Handle multiple spaces or special characters between words gracefully.

3 Algorithm

3.1 Steps to Solve the Problem

1. Identify the start and end of each word in the string.
2. Reverse each word in place:
    - Swap characters from the start and end of the word using two pointers.
    - Continue swapping until the pointers meet.
3. Maintain the original order of the words by processing the string sequentially.

4 Code

```java
import java.util.Scanner;

public class ReverseWords {

    // Method to reverse a word in place
    public static void reverseWord(char[] str, int start, int end) {
        while (start < end) {
            char temp = str[start];
            str[start] = str[end];
            str[end] = temp;
            start++;
            end--;
        }
    }

    // Method to reverse words in a string while maintaining their order
    public static String reverseWords(String str) {
        char[] chars = str.toCharArray();
        int start = 0;

        // Reverse each word
        for (int i = 0; i <= chars.length; i++) {
            if (i == chars.length || chars[i] == ' ') {
                reverseWord(chars, start, i - 1);
                start = i + 1;
            }
        }

        return new String(chars);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Input the string
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        // Reverse the words in the string
        String result = reverseWords(input);

        // Output the modified string
        System.out.println("Reversed words: " + result);

        scanner.close();
    }
}
```

5 Step-by-Step Explanation

1. **Reverse individual words:**
   - Use a helper function (`reverseWord()`) to reverse characters of a word in place.
   - Process each word by identifying its start and end indices.
2. **Preserve word order:**
   - Iterate through the string, applying the reversal logic to each word sequentially.
   - Handle the end of the string using a special case where `'\0'` is encountered.

6 Complexity Analysis

- **Time Complexity:** $O(n)$ Each character is processed once for reversing words.
- **Space Complexity:** $O(1)$ The algorithm operates in place without using additional memory.

7 Examples and Edge Cases

Input String	Output String	Description
"Hello World"	"olleH dlroW"	Two words reversed
" Code Everyday "	" edoC yadrevE "	Handles leading/trailing spaces
"DSA is fun!"	"ASD si !nuf"	Works with special characters
""	""	Empty string remains unchanged

8 Conclusion

The program efficiently reverses words in a string while maintaining their original order. This solution operates in-place, making it both time and space efficient. It demonstrates the power of two-pointer techniques for solving string manipulation problems.

9 Output

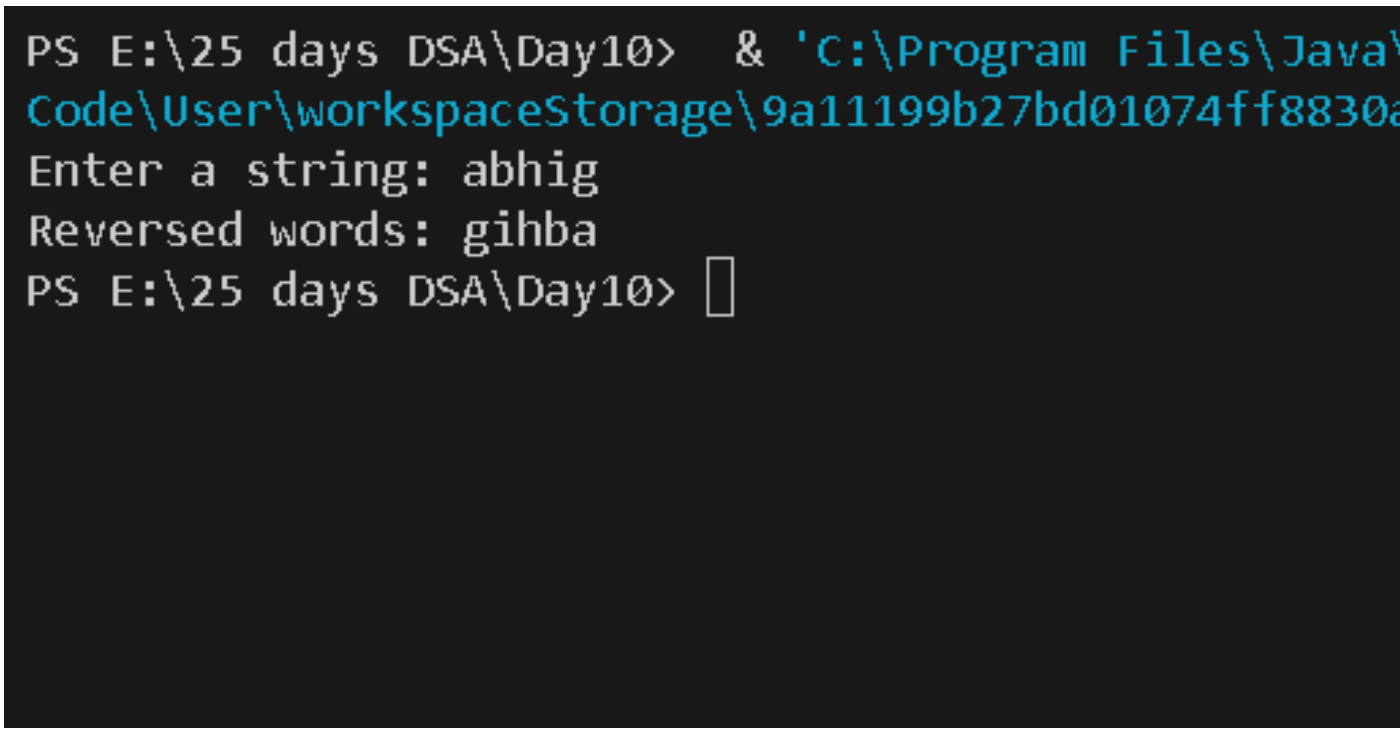

Figure 1: Output