<https://leetcode.com/problems/percentage-of-letter-in-string/description/>

Given a string s and a character letter, return*the****percentage****of characters in*s*that equal*letter***rounded down****to the nearest whole percent.*

**Example 1:**

**Input:** s = "foobar", letter = "o"

**Output:** 33

**Explanation:**The percentage of characters in s that equal the letter 'o' is 2 / 6 \* 100% = 33% when rounded down, so we return 33.

**Example 2:**

**Input:** s = "jjjj", letter = "k"

**Output:** 0

**Explanation:**The percentage of characters in s that equal the letter 'k' is 0%, so we return 0.

**Constraints:**

1 <= s.length <= 100

s consists of lowercase English letters.

letter is a lowercase English letter.

**Attempt 1: 2025-03-30**

**Solution 1: (10min)**

class Solution {

    public int percentageLetter(String s, char letter) {

        int count = 0;

        int n = s.length();

        for(int i = 0; i < n; i++) {

            count += (s.charAt(i) == letter ? 1 : 0);

        }

        // Don't write as count / n \* 100, that will always give 0

        return count \* 100 / n;

    }

}

Time Complexity: O(n), where n is the length of the string. We perform a single pass through the string.

Space Complexity: O(1), as we only use a constant amount of additional space regardless of the input size.

**Refer to Deepseek**

This problem requires calculating the percentage of occurrences of a specific letter in a string (case-sensitive) and returning the integer part of that percentage.

**Solution Approach**

Count Occurrences: Iterate through the string and count how many times the target letter appears.

Calculate Percentage: Compute the percentage by dividing the count by the string length and multiplying by 100.

Return Integer Part: Return the integer portion of the calculated percentage.

**Java Solution**

class Solution {

public int percentageLetter(String s, char letter) {

int count = 0;

int length = s.length();

// Count occurrences of the letter

for (int i = 0; i < length; i++) {

if (s.charAt(i) == letter) {

count++;

}

}

// Calculate percentage and return the integer part

return (count \* 100) / length;

}

}

**Explanation**

Initialization: We initialize a counter count to zero and get the string length.

Counting: We iterate through each character in the string. For each character that matches the target letter, we increment our counter.

Percentage Calculation: We calculate the percentage by multiplying the count by 100 and then dividing by the string length. Since we're using integer division, this automatically gives us the floor value (integer part) of the percentage.

Return Result: The calculated integer percentage is returned.

**Complexity Analysis**

Time Complexity: O(n), where n is the length of the string. We perform a single pass through the string.

Space Complexity: O(1), as we only use a constant amount of additional space regardless of the input size.

**Alternative Solution (Using Streams)**

For those who prefer a more concise Java 8+ solution:

class Solution {

public int percentageLetter(String s, char letter) {

long count = s.chars().filter(c -> c == letter).count();

return (int) (count \* 100 / s.length());

}

}

This solution uses Java streams to count the occurrences, but has the same time and space complexity as the first solution. The stream version might be slightly less efficient due to stream overhead, but offers more concise code.

**Refer to**

[L451.P14.5.Sort Characters By Frequency (Ref.L347)](note://512A6121D907491D8CF8FDD9119E5CC3)