This is a simple and efficient Java solution to count how many strings in an array of words start with a given prefix.

**Code Breakdown:**

**1. Class Declaration:**

class Solution {

This declares a class named Solution. The class is typically part of a larger program or used as a utility class in competitive programming or problem-solving platforms.

**2. Method Declaration:**

public int prefixCount(String[] words, String pref) {

* public: This method is accessible from outside the class.
* int: The method returns an integer, which represents the count of words that have the specified prefix.
* prefixCount(String[] words, String pref): The method takes two parameters:
  + words: An array of strings (String[]) containing the words to check.
  + pref: A string representing the prefix to look for.

**3. Variable Initialization:**

int count = 0;

A counter count is initialized to 0. This will store the number of words in the words array that have the prefix pref.

**4. Iterate Through the Array:**

for (String word : words) {

This is an enhanced for loop (also called a "for-each" loop) that iterates over each string in the words array. The current string in the iteration is assigned to the variable word.

**5. Check Prefix:**

if (word.startsWith(pref)) {

The startsWith method of the String class checks if the word begins with the prefix pref. If it does, the condition evaluates to true.

**6. Increment Counter:**

count++;

If the current word starts with the prefix, the counter count is incremented by 1.

**7. Return Result:**

return count;

After the loop completes, the method returns the final value of count, which represents the total number of words in the array that have the specified prefix.

**Example Usage:**

public static void main(String[] args) {

Solution sol = new Solution();

String[] words = {"apple", "app", "apricot", "banana"};

String prefix = "ap";

int result = sol.prefixCount(words, prefix);

System.out.println(result); // Output: 3

}

**Explanation of Example:**

* Words in the array: "apple", "app", "apricot", "banana".
* Prefix to check: "ap".
* Words starting with "ap": "apple", "app", "apricot".
* Result: 3.

**Time Complexity:**

* **O(n \* m)**: Where n is the number of words in the words array, and m is the average length of the strings (for startsWith comparison).
* In practice, this is efficient for reasonably sized arrays and string lengths.

**Space Complexity:**

* **O(1)**: Only a counter variable is used; no extra space is allocated.