groupMap.computeIfAbsent(item.substring(0, 1), key -> new ArrayList<>()).add(item);

**Explanation:**

1. **item.substring(0, 1)**:
   * This extracts the first character (as a string) from item. It acts as the **grouping key**.
   * Example: If item = "apple", then item.substring(0, 1) evaluates to "a".
2. **groupMap.computeIfAbsent(...)**:
   * Checks if the groupMap already contains an entry for the extracted key ("a", in this case).
   * If the key is **absent**, the lambda function (key -> new ArrayList<>()) is executed to create a new, empty ArrayList, which is then stored in the map under that key.
3. **Lambda Function (key -> new ArrayList<>())**:
   * This provides the default value (a new ArrayList) if the key is not already present in the map.
   * If the key **exists**, the function is not called, and the existing value (the list) is used.
4. **.add(item)**:
   * Whether the list is newly created or already exists in the map, item is added to the list.
   * This allows grouping multiple values under the same key.

map.computeIfPresent("A", (key, value) -> value + 5);

System.out.println(map.getOrDefault("C", "Default")); // Output: Default

The TreeMap class in Java is a part of the java.util package and implements the NavigableMap interface, which in turn extends SortedMap. It is a **Red-Black Tree-based implementation** that provides a naturally ordered collection of key-value pairs. Here's a detailed breakdown of its internal working:

[**Stack data structure**](https://www.geeksforgeeks.org/stack-data-structure/). The class is based on the basic principle of **LIFO**(last-in-first-

97-122 a-z

48-57 0-9