Anagram Checker Dry Run

```
public class AnagramChecker {
    public static boolean isAnagram(String s, String t) {
        // If lengths of the strings are not equal, they cannot be anagrams
        if (s.length() != t.length()) {
            return false;
        }

        // Create an array to count character frequencies
        int[] charCount = new int[26]; // For lowercase English letters only

        // Increment for characters in s and decrement for characters in t
        for (int i = 0; i < s.length(); i++) {
            charCount[s.charAt(i) - 'a']++;
            charCount[t.charAt(i) - 'a']--;
        }

        // Check if all counts are zero
        for (int count : charCount) {
            if (count != 0) {
                return false;
            }
        }

        return true;
    }

    public static void main(String[] args) {
        String s = "listen";
        String t = "silent";

        boolean result = isAnagram(s, t);
        System.out.println("Is anagram: " + result);
    }
}. Dry run the code
```

Code Dry Run

Objective: To check if the two strings `s` and `t` are anagrams of each other. Anagrams have the same characters with the same frequencies.

Inputs:

- `s = "listen"`

- `t = "silent"`

Step-by-Step Execution

Initial Conditions

1. `s = "listen"`, `t = "silent"`.
2. `s.length() = 6`, `t.length() = 6` → Since the lengths are equal, proceed.

Step 1: Initialize `charCount`

- `charCount = new int[26]` → This array is initialized to all zeroes and will track character frequency differences for lowercase English letters (`a` to `z`).

Step 2: Loop through the characters of `s` and `t`

Iterate over each character in `s` and `t`:

1. **Iteration 1 (i = 0)**:
   - `s.charAt(0) = 'l'` → `'l' - 'a' = 11`.
     Increment `charCount[11]`: `charCount[11]++` → `charCount[11] = 1`.
   - `t.charAt(0) = 's'` → `'s' - 'a' = 18`.
     Decrement `charCount[18]`: `charCount[18]--` → `charCount[18] = -1`.
   - `charCount = [0, 0, ..., 1, 0, ..., -1, ..., 0]`.
2. **Iteration 2 (i = 1)**:
   - `s.charAt(1) = 'i'` → `'i' - 'a' = 8`.
     Increment `charCount[8]`: `charCount[8]++` → `charCount[8] = 1`.
   - `t.charAt(1) = 'i'` → `'i' - 'a' = 8`.
     Decrement `charCount[8]`: `charCount[8]--` → `charCount[8] = 0`.
   - `charCount = [0, 0, ..., 1, 0, ..., -1, ..., 0]`.
3. **Iteration 3 (i = 2)**:
   - `s.charAt(2) = 's'` → `'s' - 'a' = 18`.
     Increment `charCount[18]`: `charCount[18]++` → `charCount[18] = 0`.
   - `t.charAt(2) = 'l'` → `'l' - 'a' = 11`.
     Decrement `charCount[11]`: `charCount[11]--` → `charCount[11] = 0`.
   - `charCount = [0, 0, ..., 0, 0, ..., 0, ..., 0]`.
4. **Iteration 4 (i = 3)**:
   - `s.charAt(3) = 't'` → `'t' - 'a' = 19`.
     Increment `charCount[19]`: `charCount[19]++` → `charCount[19] = 1`.
   - `t.charAt(3) = 'e'` → `'e' - 'a' = 4`.
     Decrement `charCount[4]`: `charCount[4]--` → `charCount[4] = -1`.
   - `charCount = [0, 0, ..., -1, ..., 1, ..., 0]`.
5. **Iteration 5 (i = 4)**:
   - `s.charAt(4) = 'e'` → `'e' - 'a' = 4`.
     Increment `charCount[4]`: `charCount[4]++` → `charCount[4] = 0`.

- `t.charAt(4) = 't'` → `'t' - 'a' = 19`.
  Decrement `charCount[19]`: `charCount[19]--` → `charCount[19] = 0`.
- `charCount = [0, 0, ..., 0, ..., 0, ..., 0]`.

6. **Iteration 6 (i = 5)**:
   - `s.charAt(5) = 'n'` → `'n' - 'a' = 13`.
     Increment `charCount[13]`: `charCount[13]++` → `charCount[13] = 1`.
   - `t.charAt(5) = 'n'` → `'n' - 'a' = 13`.
     Decrement `charCount[13]`: `charCount[13]--` → `charCount[13] = 0`.
   - `charCount = [0, 0, ..., 0, ..., 0, ..., 0]`.

Step 3: Verify `charCount`

- Check if all values in `charCount` are 0:
  - `charCount = [0, 0, 0, ..., 0]` → All values are 0.

Final Result

- Since all values in `charCount` are 0, the strings `s` and `t` are anagrams.
- **Output**: `true`

Console Output:

```vbnet

 Is anagram: true
```