Anagram.java

```
1   package StringAlgorithms;
2
3   import java.util.Arrays;
4   import java.util.HashMap;
5
6   /**
7    * An anagram is a word or phrase formed by rearranging the letters of a different word or phrase,
8    * typically using all the original letters exactly once.[1]
9    * For example, the word anagram itself can be rearranged into nag a ram,
10   * also the word binary into brainy and the word adobe into abode.
11   * Reference from https://en.wikipedia.org/wiki/Anagram
12   */
13  public class Anagram {
14          boolean approach1(String s, String t) {
15 1                if (s.length() != t.length()) {
16 1                        return false;
17                 } else {
18                         char c[] = s.toCharArray();
19                         char d[] = t.toCharArray();
20 1                        Arrays.sort(c);
21 1                        Arrays.sort(d);
22 1                        if (Arrays.equals(c, d)) {
23 1                                return true;
24                         } else {
25 1                                return false;
26                         }
27                 }
28          }
29
30          boolean approach2(String a, String b) {
31 1                if (a.length() != b.length()) {
32 1                        return false;
33                 } else {
34                         int m[] = new int[26];
35                         int n[] = new int[26];
36                         for (char c : a.toCharArray()) {
37 2                                m[c - 'a']++;
38                         }
39                         for (char c : b.toCharArray()) {
40 2                                n[c - 'a']++;
41                         }
42 2                        for (int i = 0; i < 26; i++) {
43 1                                if (m[i] != n[i]) {
44 1                                        return false;
45                                 }
46                         }
47 1                        return true;
48                 }
49          }
50
51          boolean approach3(String s, String t) {
52 1                if (s.length() != t.length()) {
53 1                        return false;
54                 }
55                 else {
56                         int a[] = new int[26];
57                         int b[] = new int[26];
58                         int k = s.length();
59 2                        for (int i = 0; i < k; i++) {
60 2                                a[s.charAt(i) - 'a']++;
```

```
61 2                            b[t.charAt(i) - 'a']++;
62                          }
63 2                        for (int i = 0; i < 26; i++) {
64 2                            if (a[i] != b[i]) return false;
65                          }
66 1                        return true;
67                  }
68          }
69
70          boolean approach4(String s, String t) {
71 1                 if (s.length() != t.length()) {
72 1                         return false;
73                  }
74                  else {
75                          HashMap<Character, Integer> nm = new HashMap<>();
76                          HashMap<Character, Integer> kk = new HashMap<>();
77                          for (char c : s.toCharArray()) {
78 1                              nm.put(c, nm.getOrDefault(c, 0) + 1);
79                          }
80                          for (char c : t.toCharArray()) {
81 1                              kk.put(c, kk.getOrDefault(c, 0) + 1);
82                          }
83 2                        return nm.equals(kk);
84                  }
85          }
86  }
```

Mutations

15 1. negated conditional → KILLED
16 1. replaced boolean return with true for StringAlgorithms/Anagram::approach1 → KILLED
20 1. removed call to java/util/Arrays::sort → KILLED
21 1. removed call to java/util/Arrays::sort → KILLED
22 1. negated conditional → KILLED
23 1. replaced boolean return with false for StringAlgorithms/Anagram::approach1 → KILLED
25 1. replaced boolean return with true for StringAlgorithms/Anagram::approach1 → KILLED
31 1. negated conditional → KILLED
32 1. replaced boolean return with true for StringAlgorithms/Anagram::approach2 → KILLED
37 1. Replaced integer subtraction with addition → KILLED
2. Replaced integer addition with subtraction → KILLED
40 1. Replaced integer subtraction with addition → KILLED
2. Replaced integer addition with subtraction → KILLED
42 1. changed conditional boundary → KILLED
2. negated conditional → KILLED
43 1. negated conditional → KILLED
44 1. replaced boolean return with true for StringAlgorithms/Anagram::approach2 → KILLED
47 1. replaced boolean return with false for StringAlgorithms/Anagram::approach2 → KILLED
52 1. negated conditional → KILLED
53 1. replaced boolean return with true for StringAlgorithms/Anagram::approach3 → KILLED
59 1. changed conditional boundary → KILLED
2. negated conditional → KILLED
60 1. Replaced integer subtraction with addition → KILLED
2. Replaced integer addition with subtraction → KILLED
61 1. Replaced integer subtraction with addition → KILLED
2. Replaced integer addition with subtraction → KILLED
63 1. changed conditional boundary → KILLED
2. negated conditional → KILLED
64 1. replaced boolean return with true for StringAlgorithms/Anagram::approach3 → KILLED
2. negated conditional → KILLED
66 1. replaced boolean return with false for StringAlgorithms/Anagram::approach3 → KILLED
71 1. negated conditional → KILLED
72 1. replaced boolean return with true for StringAlgorithms/Anagram::approach4 → KILLED
78 1. Replaced integer addition with subtraction → KILLED
81 1. Replaced integer addition with subtraction → KILLED
83 1. replaced boolean return with false for StringAlgorithms/Anagram::approach4 → KILLED

```
2. replaced boolean return with true for StringAlgorithms/Anagram::approach4 → KILLED
```

Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR

Tests examined

- StringAlgorithms.AllStringTesting.[engine:junit-jupiter]/[class:StringAlgorithms.AllStringTesting]/[method:testAnagram3()] (17 ms)
- StringAlgorithms.AllStringTesting.[engine:junit-jupiter]/[class:StringAlgorithms.AllStringTesting]/[method:testAnagram2()] (10 ms)
- StringAlgorithms.AllStringTesting.[engine:junit-jupiter]/[class:StringAlgorithms.AllStringTesting]/[method:testAnagram4()] (11 ms)
- StringAlgorithms.AllStringTesting.[engine:junit-jupiter]/[class:StringAlgorithms.AllStringTesting]/[method:letterCombinationsOfPhoneNumber()] (31 ms)
- StringAlgorithms.AllStringTesting.[engine:junit-jupiter]/[class:StringAlgorithms.AllStringTesting]/[method:testAnagram1()] (14 ms)

Report generated by PIT 1.6.8