CheckAnagrams.java

```
   package com.example;

   import java.util.HashMap;
   import java.util.Map;

   /**
    * Two strings are anagrams if they are made of the same letters arranged
    * differently (ignoring the case).
    */
   public class CheckAnagrams {
       /**
        * Check if two strings are anagrams or not
        *
        * @param s1 the first string
        * @param s2 the second string
        * @return {@code true} if two string are anagrams, otherwise {@code false}
        */
       public static boolean isAnagrams(String s1, String s2) {
           int l1 = s1.length();
           int l2 = s2.length();
           s1 = s1.toLowerCase();
           s2 = s2.toLowerCase();
           Map<Character, Integer> charAppearances = new HashMap<>();

3          for (int i = 0; i < l1; i++) {
               char c = s1.charAt(i);
               int numOfAppearances = charAppearances.getOrDefault(c, 0);
1              charAppearances.put(c, numOfAppearances + 1);
           }

3          for (int i = 0; i < l2; i++) {
               char c = s2.charAt(i);
1              if (!charAppearances.containsKey(c)) {
1                  return false;
               }
1              charAppearances.put(c, charAppearances.get(c) - 1);
           }

           for (int cnt : charAppearances.values()) {
1              if (cnt != 0) {
1                  return false;
               }
           }
1          return true;
       }

       /**
        * If given strings contain Unicode symbols.
        * The first 128 ASCII codes are identical to Unicode.
        * This algorithm is case-sensitive.
        *
        * @param s1 the first string
        * @param s2 the second string
        * @return true if two string are anagrams, otherwise false
        */
       public static boolean isAnagramsUnicode(String s1, String s2) {
           int[] dict = new int[128];
           for (char ch : s1.toCharArray()) {
1              dict[ch]++;
           }
           for (char ch : s2.toCharArray()) {
1              dict[ch]--;
           }
           for (int e : dict) {
1              if (e != 0) {
1                  return false;
               }
           }
1          return true;
       }

       /**
        * If given strings contain only lowercase English letters.
        * <p>
        * The main "trick":
        * To map each character from the first string 's1' we need to subtract an integer value of 'a' character
        * as 'dict' array starts with 'a' character.
        *
        * @param s1 the first string
        * @param s2 the second string
        * @return true if two string are anagrams, otherwise false
        */
       public static boolean isAnagramsOptimised(String s1, String s2) {
           // 26 - English alphabet length
           int[] dict = new int[26];
           for (char ch : s1.toCharArray()) {
1              checkLetter(ch);
2              dict[ch - 'a']++;
           }
           for (char ch : s2.toCharArray()) {
1              checkLetter(ch);
2              dict[ch - 'a']--;
           }
           for (int e : dict) {
```

```
95  1              if (e != 0) {
96  1                  return false;
97                 }
98             }
99  1          return true;
100        }
101
102        private static void checkLetter(char ch) {
103 1          int index = ch - 'a';
104 4          if (index < 0 || index >= 26) {
105                throw new IllegalArgumentException("Strings must contain only lowercase English letters!");
106            }
107        }
108 }
```

Mutations

```
25   1. changed conditional boundary → KILLED
     2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
28   1. Replaced integer addition with subtraction → KILLED
31   1. changed conditional boundary → KILLED
     2. Changed increment from 1 to -1 → KILLED
     3. negated conditional → KILLED
33   1. negated conditional → KILLED
34   1. replaced boolean return with true for com/example/CheckAnagrams::isAnagrams → KILLED
36   1. Replaced integer subtraction with addition → KILLED
40   1. negated conditional → KILLED
41   1. replaced boolean return with true for com/example/CheckAnagrams::isAnagrams → NO_COVERAGE
44   1. replaced boolean return with false for com/example/CheckAnagrams::isAnagrams → KILLED
59   1. Replaced integer addition with subtraction → KILLED
62   1. Replaced integer subtraction with addition → KILLED
65   1. negated conditional → KILLED
66   1. replaced boolean return with true for com/example/CheckAnagrams::isAnagramsUnicode → KILLED
69   1. replaced boolean return with false for com/example/CheckAnagrams::isAnagramsUnicode → KILLED
87   1. removed call to com/example/CheckAnagrams::checkLetter → KILLED
88   1. Replaced integer subtraction with addition → KILLED
     2. Replaced integer addition with subtraction → KILLED
91   1. removed call to com/example/CheckAnagrams::checkLetter → SURVIVED
92   1. Replaced integer subtraction with addition → KILLED
     2. Replaced integer subtraction with addition → KILLED
95   1. negated conditional → KILLED
96   1. replaced boolean return with true for com/example/CheckAnagrams::isAnagramsOptimised → NO_COVERAGE
99   1. replaced boolean return with false for com/example/CheckAnagrams::isAnagramsOptimised → KILLED
103  1. Replaced integer subtraction with addition → KILLED
104  1. changed conditional boundary → KILLED
     2. changed conditional boundary → SURVIVED
     3. negated conditional → KILLED
     4. negated conditional → 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

- com.example.CheckAnagramsTest.testOptimisedAlgorithmStringsAreValidAnagrams(com.example.CheckAnagramsTest) (0 ms)
- com.example.CheckAnagramsTest.testStringAreValidAnagramsCaseSensitive(com.example.CheckAnagramsTest) (1 ms)
- com.example.CheckAnagramsTest.testCheckAnagrams(com.example.CheckAnagramsTest) (1 ms)
- com.example.CheckAnagramsTest.testCheckDifferentCasesAnagram(com.example.CheckAnagramsTest) (0 ms)
- com.example.CheckAnagramsTest.testOptimisedAlgorithmShouldThrowExceptionWhenStringsContainUppercaseLetters(com.example.CheckAnagramsTest) (0 ms)
- com.example.CheckAnagramsTest.testCheckSameWordAnagrams(com.example.CheckAnagramsTest) (1 ms)
- com.example.CheckAnagramsTest.testStringAreNotAnagramsCaseSensitive(com.example.CheckAnagramsTest) (0 ms)
- com.example.CheckAnagramsTest.testCheckFalseAnagrams(com.example.CheckAnagramsTest) (0 ms)

Report generated by PIT 1.5.0