Exercise 2

Instructions

Given two strings s1 and s2, we want to visualize how different the two strings are. We will only take into account the *lowercase* letters (a to z). First let us count the frequency of each *lowercase* letters in s1 and s2.

```
s1 = "A aaaa bb c"

s2 = "& aaa bbb c d"

s1 has 4 'a', 2 'b', 1 'c'

s2 has 3 'a', 3 'b', 1 'c', 1 'd'
```

So the maximum for 'a' in s1 and s2 is 4 from s1; the maximum for 'b' is 3 from s2. In the following we will not consider letters when the maximum of their occurrences is less than or equal to 1.

We can resume the differences between s1 and s2 in the following string: "1:aaaa/2:bbb" where 1 in 1:aaaa stands for string s1 and aaaa because the maximum for a is 4. In the same manner 2:bbb stands for string s2 and bbb because the maximum for b is 3.

The task is to produce a string in which each *lowercase* letters of s1 or s2 appears as many times as its maximum if this maximum is *strictly greater than 1*; these letters will be prefixed by the number of the string where they appear with their maximum value and :. If the maximum is in s1 as well as in s2 the prefix is =:.

In the result, substrings (a substring is for example 2:nnnnn or 1:hhh; it contains the prefix) will be in decreasing order of their length and when they have the same length sorted in ascending lexicographic order (letters and digits - more precisely sorted by codepoint); the different groups will be separated by '/'. See examples and "Example Tests".

Hopefully other examples can make this clearer.

```
s1 = "my&friend&Paul has heavy hats! &"

s2 = "my friend John has many many friends &"

mix(s1, s2) -->
"2:nnnnn/1:aaaa/1:hhh/2:mmm/2:yyy/2:dd/2:ff/2:ii/2:rr/=:ee/=:ss"



s1 = "mmmmm m nnnnn y&friend&Paul has heavy hats! &"

s2 = "my frie n d Joh n has ma n y ma n y frie n ds n&"

mix(s1, s2) -->
"1:mmmmmm/=:nnnnnn/1:aaaa/1:hhh/2:yyy/2:dd/2:ff/2:ii/2:rr/=:ee/=:ss"



s1="Are the kids at home? aaaaa fffff"

s2="Yes they are here! aaaaa fffff"

mix(s1, s2) --> "=:aaaaaa/2:eeeee/=:fffff/1:tt/2:rr/=:hh"
```

Solution

```
public class Mixing {

     public static String mix(String s1, String s2) {
     // your code
     }
}
```

Tests

```
import static org.junit.Assert.*;
import java.util.Comparator;
import java.util.Arrays;
import org.junit.Test;

public class MixingTest {
```

```
    @Test
    public void test() {
    System.out.println("Mix Fixed Tests");
    assertEquals("2:eeeee/2:yy/=:hh/=:rr", Mixing.mix("Are they
here", "yes, they are here"));
    assertEquals("1:ooo/1:uuu/2:sss/=:nnn/1:ii/2:aa/2:dd/2:ee/=:gg",
              Mixing.mix("looping is fun but dangerous", "less
dangerous than coding"));

assertEquals("1:aaa/1:nnn/1:gg/2:ee/2:ff/2:ii/2:oo/2:rr/2:ss/2:tt",
              Mixing.mix(" In many languages", " there's a pair of
functions"));
    assertEquals("1:ee/1:ll/1:oo", Mixing.mix("Lords of the Fallen",
"gamekult"));
    assertEquals("", Mixing.mix("codewars", "codewars"));
    assertEquals("1:nnnnn/1:ooooo/1:tttt/1:eee/1:gg/1:ii/1:mm/=:rr",
              Mixing.mix("A generation must confront the looming ",
"codewarrs"));
    }

}
```