CS 321 Homework 0 – Attempt by Monday, Oct. 4, but do NOT hand in!

Checking for Anagrams

Two words are anagrams if they contain the same letters, possibly in a different order. For example, `brad`, `bard`, and `drab` are all anagrams of each other. If we also allow "words" that are not in any English dictionary, then there are many other possible anagrams of these three, such as `arbd`, `dbar`, etc. Also, every word is an anagram of itself.

Your task is to write a Java application class, `AnaTest`, that tests two or more character strings to see if they are all anagrams of each other. The strings to be tested are provided as command line arguments. You can assume that the strings all consist of lowercase alphabetic characters (`a`, `b`, `c`, . . ., `z`). Your program should examine all the strings provided; if they are all anagrams of each other (or if fewer than two strings are given) it should print the string `true` to standard output; otherwise it should print the word `false`. For example:

```
% java AnaTest arbd brad radb
true
% java AnaTest arb arc rab ra
false
% java AnaTest x
true
%
```

The purpose of this exercise is to get you going using Java and the specific compiler and JVM on your system. This exercise will *not* be graded and should *not* be submitted.

There are many possible algorithms for detecting anagrams: think about sorting and counting. Once you've chosen an algorithm, there will be many ways to code it in Java. You may well find useful routines in the Java libraries; in particular, take a look at `java.lang.String` and `java.util.Arrays`.