# CPW 143 Spring 2017, Programming Assignment 10
# Sorting

## **Purpose**:
Write a program that discovers all anagrams of all words listed in a file.

## **Description:**
The starter code, **`AnagramFinder.java`**, and data file, **`wordlist.txt`**, are provided to you. What's missing is a **`Comparator`**. Your job is to implement that **`Comparator`** class, **`AnagramComparator.java`**. Below is an example of a **`Comparator`** (from Chapter 13, pages 839-843).

```
import java.util.*;
// compares String objects by length
public class LengthComparator implements Comparator<String> {
    public int compare( String s1, String s2 ) {
        return s1.length() - s2.length();
    }
}
```

An anagram of a word is a rearrangement of its letters into another legal word. For example, an anagram of the word "ABACAS" is "CASABA". The provided file, **`wordlist.txt`**, contains many words which are read into a **`List`**. The list is then sorted, but not in alphabetical order. Instead, your **`Comparator`** must order the words by canonical form. The ***canonical*** form of a word contains the same letters as the original, but in sorted order. Thus, the canonical form of "computer" is "cemoprtu", and the canonical form of "program" is "agmoprr". If a word list containing both "program" and "computer" was sorted in order of canonical form, "program" would appear in the list before "computer". If "ABACAS" and "CASABA" both were in the list, they would appear next to each other.

One way to produce canonical form is to convert a **`String`** to an array of **`char`**, sort that array, then convert back to **`String`**. See the JavaDoc page for **`String`** for the details of the method **`toCharArray`** and the page for **`Arrays`** for details of **`sort`**.

When **`wordlist.txt`** is sorted in order of canonical form, all the words are next to their anagrams, so we can discover anagrams from the original list by just comparing adjacent elements of the sorted list.

Here are the first 25 anagrams from **`wordlist.txt`**.

```
PARADISAICALLY and PARADISIACALLY are anagrams.
PARADISAICAL and PARADISIACAL are anagrams.
CARAVANSARIES and CARAVANSERAIS are anagrams.
ABACAS and CASABA are anagrams.
CASABAS and CASSABA are anagrams.
BREAKAWAYS and BREAKSAWAY are anagrams.
PIASABAS and PIASSABA are anagrams.
ALBATAS and ATABALS are anagrams.
ATABALS and BALATAS are anagrams.
ALBATA and ATABAL are anagrams.
ATABAL and BALATA are anagrams.
AUTORADIOGRAPHIC and RADIOAUTOGRAPHIC are anagrams.
```

```
PARADISAIC and PARADISIAC are anagrams.
ACARIDAN and ARCADIAN are anagrams.
ACARIDANS and ARCADIANS are anagrams.
MARACAS and MARASCA are anagrams.
MARASCA and MASCARA are anagrams.
MARASCAS and MASCARAS are anagrams.
CASAVAS and CASSAVA are anagrams.
AUTORADIOGRAPHIES and RADIOAUTOGRAPHIES are anagrams.
ADAMANTINE and AMANTADINE are anagrams.
ALAMEDAS and SALAAMED are anagrams.
AUTORADIOGRAPHS and RADIOAUTOGRAPHS are anagrams.
AUTORADIOGRAPH and RADIOAUTOGRAPH are anagrams.
AUTORADIOGRAPHY and RADIOAUTOGRAPHY are anagrams.
```

## Submit:

The only thing you need to turn in (upload to Canvas) is **`AnagramComparator.java`**.