THE STRATEGY PATTERN

CREATE A BUNCH OF ALGORITHM OBJECTS

EACH OF WHICH IMPLEMENTS
THE INTERFACE

EACH "ALGORITHM OBJECT"
CAN SPECIFY ITS OWN LOGIC
FOR DETERMINING THIS ORDER

Comparator<String>

ANY OBJECT THAT IMPLEMENTS
THIS INTERFACE HAS A METHOD
THAT TAKES IN 2 STRINGS, AND
SPECIFIES WHICH
STRING COMES FIRST

HOW? BY RETURNING -1,0,1 TO SAY IF
THE FIRST STRING IS "LESS THAN", "EQUAL
TO", OR "GREATER THAN" THE SECOND STRING

int compare(String s1, String s2)

ONCE YOU HAVE ALL THESE ALGORITHM OBJECTS READY,
PASS ANY ONE OF THEM INTO THE

Collections.sort(List<String> list, Comparator<String> comparator)

METHOD OF THE COLLECTIONS CLASS

THIS IS PRECISELY THE

STRATEGY PATTERN AT WORK

THIS IS PRECISELY THE

STRATEGY PATTERN AT WORK

THE STRATEGY PATTERN IS USED TO SPECIFY A BEHAVIOR ("HOW TO SORT")

ON THE FLY (WHEN THE USER SELECTS A SORT METHOD FROM A USER INTERFACE)

BY PASSING ALGORITHM OBJECTS IN AS MEMBER VARIABLES

IMPLEMENTING THE STRATEGY PATTERN

THEN PASS AN OBJECT OF THIS CLASS RIGHT INTO THE COLLECTIONS.SORT METHOD

```
// Step 1: Create a list of strings
List<String> listOfStrings = new ArrayList<String>();

// Step 2: populate listOfStrings with names from a data file
// Won't bother with the code for this here

// Step 3: Sort this list of strings in lexicographical order,
// but with the added twist that if the name "Donald Trump" appears
// in this list, it must appear first.
Collections.sort(listOfStrings, new Comparator<String>() {
    public int compare(String s1, String s2) {
        if (s1.equals("Donald Trump") && !s2.equals("Donald Trump")) {
            return 1;
        } else if (s2.equals("Donald Trump") && !s1.equals("Donald Trump")) {
            return -1;
        }
        return s1.compareTo(s2);
    }
});
```

CREATE AND INSTANTIATE A CLASS THAT IMPLEMENTS THE COMPARATOR<STRING> INTERFACE IN A SINGLE LINE

COMPARATOR<STRING> HAS ONLY ONE MEMBER FUNCTION THAT IT MUST IMPLEMENT - DO THAT RIGHT HERE ON THE FLY

WE CAN CREATE AS MANY DIFFERENT COMPARATOR OBJECTS AS WE LIKE - EACH DEFINING A DIFFERENT SORT ALGORITHM - AND USE THEM ALL ON THE FLY

DEPENDENCY INJECTION

WHICH WE BRIEFLY MENTIONED -

IS A WIDELY USED AND VERY POWERFUL TECHNIQUE OF SETTING UP MEMBER VARIABLES OF COMPLICATED CLASSES ON THE FLY

IT IS FUNDAMENTALLY SIMILAR TO THE STRATEGY PATTERN

FOR THESE 2 REASONS IT IS WORTH EXPLORING IN MORE DETAIL