CSCI 136: Data Structures and Advanced Programming
Lecture 13
Sorting, part 1
Instructor: Dan Barowy Williams

<table>
<thead>
<tr>
<th>Your to-dos</th>
<th>Announcements</th>
</tr>
</thead>
</table>
| 1. Read **before Wed**: Bailey, Ch 6.4, 6.7-6.9.  
2. Quiz 4, **due Saturday by 6pm**.  
3. Lab 4, **due Tuesday 10/11 by 10pm**.  |  
• TA hours over reading period: **business as usual**  
• Kelly: out of town Monday, Dan will fill in for office hours, but **over Zoom**. See help calendar for link.  
• Colloquium: **What I Did Last Summer (Industry)**, 2:35pm in Wege Auditorium with **cookies**. |

Topics
- More inheritance  
- Comparison sorting
Study Tip #3

Develop a habit of writing little programs.

Recall:

Java provides control over abstractness, which we can use to enforce behavior to varying degrees.

- interface → fully abstract
- abstract class → partially abstract
- class → not abstract

Honkable

Abstract class

An abstract class is a partial implementation, mainly used as a labor-saving device.

E.g., many List implementations will implement methods the same way. Why duplicate all that work?

isEmpty() can always be implemented by checking that size() == 0.
Inheritance is a mechanism for defining a class in terms of another class. It is a labor-saving device employed to reduce code duplication. Inheritance allows programmers to specify a new implementation while:

1. maintaining the same behavior,
2. reusing code, and
3. extending the functionality of existing software.

How to interpret Javadoc declarations

public class Vector<E>
   extends AbstractList<E>
   implements Cloneable

Generic: any type of element
Borrows code from AbstractList
Behaves the same as Cloneable
Sorting algorithms

Sorting algorithm

A sorting algorithm is a procedure for transforming an unordered set of data into an ordered sequence.

A comparison sorting algorithm takes as input a set $S$ and a binary relation $<$ that defines an ordering on $S$.

Strict weak order

A strict weak order is a mathematical formalization of the intuitive notion of a ranking of a set, some of whose members may be tied with each other.

A strict weak order has the following properties:

• Irreflexivity: For all $x$ in $S$, it is not the case that $x < x$.
• Asymmetry: For all $x$, $y$ in $S$, where $x \neq y$, if $x < y$ then it is not the case that $y < x$.
• Transitivity: For all $x$, $y$, $z$ in $S$, where $x \neq y \neq z \neq x$, if $x < y$ and $y < z$ then $x < z$.
• Transitivity of Incomparability: For all $x$, $y$, $z$ in $S$, where $x \neq y \neq z \neq x$, if $x$ is incomparable with $y$ (neither $x < y$ nor $y < x$ hold), and $y$ is incomparable with $z$, then $x$ is incomparable with $z$.

Example order

Example: lexicographical order (aka, “dictionary order”):

Given two different sequences of the same length, $a_1a_2...a_k$ and $b_1b_2...b_k$, the first one is “less than” the second one for the lexicographical order, if $a_i < b_i$, for the first $i$ where $a_i$ and $b_i$ differ.

To compare sequences of different lengths, the shorter sequence is padded at the end with “blanks.”

Lexicographic order is a total order, meaning that there are no ties. A valid comparison sort only needs to be a weak order (i.e., ties are OK).
In-place sort

An **in-place sort** is a sort that takes an unordered set of elements as an array and **modifies** ("mutates") the original array. Most in-place sort functions return **void**.

In principle, in-place sorts can be **faster** than out-of-place algorithms, since they **do not need to copy data**.

**Tradeoff:** make sure that you don’t need the original, unsorted data!

---

Bubble sort

**Bubble sort** is an **in-place sorting algorithm** in which the largest element "bubbles up" during each pass. Bubble sort makes **n-1** passes through the data, performing pairwise comparisons of elements using `<`.

Bubble sort maintains the **invariant** (an always-true logical rule) that the rightmost **n-numSorted** elements are sorted.

I.e., bubble sort builds a sorted order to the right.
**Bubble sort** as a Hungarian dance.

Observe that two things are happening:
1. a comparison, and
2. a swap.

https://bit.ly/3RLgVo3

**Recap & Next Class**

**Today:**

- Inheritance
- Comparison sorting

**Next class:**

- More sorts

```java
public static void bubbleSort(int[] data, int n)
// pre: 0 <= n <= data.length
// post: values in data[0..n-1] in ascending order
{
    int numSorted = 0;    // number of values in order
    int index;            // general index
    while (numSorted < n)
    {
        // bubble a large element to higher array index
        for (index = 1; index < n - numSorted; index++)
        {
            if (data[index - 1] > data[index])
                swap(data, index - 1, index);
        }
        numSorted++;    // at least one more value in place
    }
}
```