Reminder about Comparators
We looked at Comparators in mid-February when we started talking about sorting.

A class T implements the *Comparable* interface if the class has a method

```java
    int compareTo(T x)
```

that compares the current element with x. Usually there is an obvious `compareTo()` method. For example, with strings `s.compareTo(x)` returns -1 if s comes before x in the usual dictionary order.

Java calls the order created by `compareTo` the *natural ordering* for the class.
However, we might want to compare elements of a class in many different ways. We might want to compare strings based on their length rather than their dictionary order. In computational biology two DNA sequences might be compared by how close they are to a standard gene. And there are many other ways we might want to compare strings. This is the role of Comparators. A class implements the Comparator\langle T\rangle interface if it has a function

\begin{verbatim}
   int compare(T x, T y)
\end{verbatim}

that compares x and y.

Usually we define a Comparator\langle T\rangle separately from the T class. A given datatype T might have many different comparator classes.
If A is an `ArrayList<T>` and Comp is a class that implements the `Comparator<T>` interface, then

```java
A.sort( new Comp() )
```

sorts A according to the Comp ordering.

For example, if you want to be able to sort from smallest to largest, and also be able to sort from largest to smallest you just provide two comparators and you can use the one you want at any point.