Interfaces
Commonly used Java interfaces

- The Java class library contains classes and interfaces
- `Comparable` – allows us to order the elements of an arbitrary class
- `Serializable (in java.io)` – for classes whose objects are able to be saved to files.
- `List, Set, Map, Iterator (in java.util)` – describe data structures for storing collections of objects
public interface Comparable<E> {
    public int compareTo(E other);
}

- A class can implement the `Comparable` interface to define a natural ordering for its objects.

- A call of `a.compareTo(b)` should return:
  a value < 0 if `a` comes "before" `b` in the ordering,
  a value > 0 if `a` comes "after" `b` in the ordering,
  or 0 if `a` and `b` are considered "equal" in the ordering.
compareTo tricks

- delegation trick - If your object's fields are comparable (such as strings), you can use their `compareTo`:

```java
// sort by employee name
public int compareTo(StaffMember other) {
    return name.compareTo(other.getName());
}
```
Comparable and sorting

- The `Arrays` class in `java.util` has a static method `sort` that sorts the elements of an array

```java
StaffMember[] staff = new StaffMember[3];
staff[0] = new Executive(...);
staff[1] = new Employee(...);
staff[2] = new Hourly(...);
staff[3] = new Volunteer(...);
Arrays.sort(staff);
```

Note that you will need to provide an implementation of `compareTo`
Another example

class Contact implements Comparable{
    private String firstName, lastName, phone;
    public boolean equals (Object other) {
        if (!(other instanceof Contact)) return false;
        return (lastName.equals(((Contact)other).getLastName()) &&
                firstName.equals(((Contact)other).getFirstName()));
    }
    // Uses both last and first names to determine ordering.
    public int compareTo (Contact other) {
        String otherFirst = other.getFirstName();
        String otherLast = other.getLastName();
        if (lastName.equals(otherLast))
            return firstName.compareTo(otherFirst);
        else
            return lastName.compareTo(otherLast);
    }
}
import java.util.*;

public class PhoneList {
    public static void main (String[] args) {
        Contact[] friends = new Contact[6];

        friends[0] = new Contact ("John", "Smith", "610-555-7384");
        friends[2] = new Contact ("Mark", "Riley", "733-555-2969");

        Arrays.sort(friends);
        for (int i=0; i<friends.length; i++)
            System.out.println (friends[i]);
    }
}