An interface is like a contract…

It's not a full class

In a class we have:

1) instance variables – items that comprise the object

2) constructors – instructions on how to build the object

3) instance methods – methods that pertain to the object and modify them, etc.

An interface only does the following:

It only lists **method signatures** that are required for anything that implements the interface.

The method signature includes:

1. The return type

2. Name of the method

3. Formal Parameter List

example

```
public int countTokens();

public int countStringsStartLetter(String[] all,
char startLetter)


public interface [PUTINTERFACENAME) {
    // Place the method signatures
    // Put semicolon at end of each.
}
```

Once an interface exists, we can create classes that implement the interface. Also, it's possible to implement more than 1 interface.

The syntax is as follows:

```
public class {PUTCLASSNAME] implements
[INTERFACE]
```

Comparable Interface

This is built into Java and it only has one method:

T is a variable representing any non-primitive type.

```
Comparable<T> interface...

// Returns a negative integer if this comes
before other.

// REturns a positive integer if this comes
after other.

// Returns 0 if they are equal.

public int compareTo(T other);
```

ArrayList class in Java

```
ArrayList<Cat> list = new ArrayList<Cat>();

add, get, size() for right now
```

Kattis Website

Problem: Sideways Sorting

In Kattis style problems, you don't prompt the user to enter any information. Instead you pretend that it'll get typed in as specified.

To test on your own, after you write your program, copy the input you want to test on in an input file. Let's say this file is input.txt. To test, you would do this on the command line:

```
java program < input.txt
```

what should happen is that the output your program produces prints out to the screen. Basically, the less than arrow on the command prompt is redirecting the input from the file input.txt to standard input.

We can store the output the program produces into a file as follows:

```
java program < input.txt > output.txt
```

What the greater than arrow does is redirect output that was supposed to go to standard output (screen) to the file name listed after the greater than sign. If that file had existed before, it's wiped out before your program's output is placed in the file.

Finally, if you have the correct output for the program saved in a file called correctoutput.txt, you can see if the two files are the same in Windows if you do

```
fc output.txt correctoutput.txt
```

If the two files are the same, you'll get the following output:

```
Comparing files myoutput.txt and correctoutput.txt
FC: no differences encountered
```