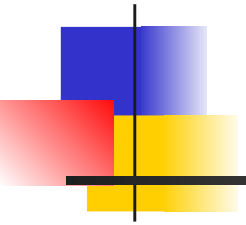

Commonly Used Interfaces

Comparable<E>

```
interface Comparable<T> {
    public int compareTo(T other);
}
```

compareTo defines an ordering of values:

```
                  < 0 if a is "before" b
a.compareTo(b) = 0 if a and b have the
                     same lexical order
                  > 0 if a is "after" b
```

String implements Comparable:

"dog".compareTo("cat") returns +1

"Dog".compareTo("cat") returns -31 (case sensitive!)

Sorting and Comparison

Arrays.sort() and Collections.sort(list) use this to sort any kind of objects that implement Comparable.

```
String[] fruit = {"Orange", "Apple", "grapes",
   "banana", "Durian"};

Arrays.sort( fruit );

> fruit
["Apple", "Durian", "Orange", "banana", "grapes"]
```

Note that compareTo is case sensitive.

The fruit are not in dictionary order!

Custom Comparison & Sorting

What if a class does not have compareTo or it does not do what we want?

Example: we want to sort fruit in dictionary order (ignoring case):

```
String[] fruit = {"Orange", "Apple", "grapes",
   "grapes", "banana", "Durian"};

Arrays.sort( fruit );

["Apple", "Durian", "Orange", "banana", "grapes"]
```

Comparator

Is there another interface we can use?

`java.util.Comparator`

```
interface Comparator<E> {
    /**
     * Compare 2 objects a and b.
     */
    int compare(E a, E b);
}
```

Write a Comparator for string

We can write a comparator for case insensitive ordering of Strings, and use it to sort the array:

```
class DictComp implements Comparable<String> {
    // case insensitive comparison
    public int compare(String a, String b) {
        return a.compareToIgnoreCase(b);
    }
}
Comparator<String> comp = new DictComp();

Arrays.sort( fruit, comp );
> fruit
["Apple", "banana", "Durian", "grapes", "Orange"]
```

Run Something?

Interface for anything with a run() method?

`Runnable`

Iterate over a collection

`Iterator` interface defines two methods:

hasNext( ) - true if the next( ) method can return another element

next( ) - return next element from the underlying source

```
List<String> fruit = Arrays.asList("Orange",
    "Apple", "grapes", "banana");
Iterator<String> iter = fruit.iterator();

while( iter.hasNext() ) {
    System.out.println( iter.next() );
}
```

Scanner is an Iterator

`Scanner implements Iterator<String>`

We can use "while" loop as previous slide to iterate over the words in a String using Scanner:

```
Scanner scan = new Scanner(
                        "Orange Apple grapes banana");

while( scan.hasNext() ) {
    System.out.println( scan.next() );
}
```