INTRODUCTORY:

LISTS, INTERFACES, JAVA 5

What's a list?

- A bunch of things. . .
- In order. . .
- Possibly with repetitions.
- More?

What can you do with a list?

You can think of a lot of things (and the implementers of Java have!), but here's what's crucial:

- add an item
  - in Java: `lis.add("broccoli")`
- remove an item
  - in Java: `lis.remove(5)`
- list the list
  Listing is such an important activity that the verb is the same as the noun.

Listing a list in Java

```
for (int i = 0; i < lis.size(); i++) {
  Object item = lis.get(i);
  System.out.println(item.toString());
}
```

But do we care about the order?

```
for (Object item : lis) {
  System.out.println(item);
}
```

The kinds of things in a list

- no kind in particular

```
List lis = new ArrayList();
```

- only Strings allowed

```
List<String> lis = new ArrayList<String>();
```

- only ints allowed
  Oops! We can't. Why not?

- only Integers allowed

```
List<Integer> lis = new ArrayList<Integer>();
```

Lists of integers

```
> import java.util.*
> List<Integer> lis = new ArrayList<Integer>();
> lis.add(new Integer(5));  // You'd expect to have to do this ...
true
> lis.add(6);               // ... but this works!
true
> lis
[5, 6]
> int sum = 0;
> for (int i : lis) {
    sum += i;
  }
> System.out.println(sum);
11
```

New stuff in Java 1.5

We just saw:

- generics
- autoboxing
- the new "iterator-based" `for` loop

Listing lists and iterators

"Tell me what's on the list."
. . . not "in the order they're written." All we want is all the things, in *some* order.

A "machine" for giving you each item in a list, as part of a programming language, is called an *iterator*. Here's the old way of using an iterator in Java:

```
Iterator it = lis.iterator();
while (it.hasNext()) {
  Object o = it.next();
  Integer i = (Integer) o;
  sum += i.intValue();
}
```

Try rewriting this with the new-style for statement.

`Iterator` is an interface

What's an iterator?

It's a thing that provides two methods:

- `hasNext()`
- `next()`

We don't care how it does this. We don't even know what actual class an Iterator belongs to when a List's `iterator()` method gives it to us. In fact, with the new-style *for* loop, we don't see the Iterator itself at all.

We'll come back to this — but remember that an interface is a promise about available behaviour.

More about lists in Java

See:

- the `List` interface in the Java API
  - *not* the `List` *class*
- the text, especially chapter 6