Imperial College London – Department of Computing

MSc in Computing Science

536: Introduction to Java
Tutorial 2 — Iterator

In this exercise you will use two Java API interfaces to make the linked list developed in the lectures *iterable*. This will allow you to use language features like the *enhanced for loop* with such lists. The interfaces, `java.lang.Iterable<T>` and `java.lang.Iterable<T>`, are part of the Java Collections framework (a bit of an extravagant name for part of the Java API, but very useful). There is a whole trail of the Java Tutorial about Collections.

1 Aims

This tutorial will help you understand how to:

- Implement inheritance in Java
- Make use of types from the Java API
- Use anonymous inner classes

2 Anonymous Classes

In this exercise you will write an *anonymous inner class*. Such a class is used when an object has a use that is only relevant to one specific point in the code. It is declared inline at this point, when an object is being created. Read the section on Anonymous classes in the Java Tutorial to see some examples:

https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html

3 Obtain Provided Files

Download the provided code for this tutorial from CATE into a suitable directory and unzip the file.

4 What To Do

4.1 Finish the Lecture Exercises

Finish implementing the inner class `Element<S>` and the `add(T)` and `toString()` methods of `List<T>`.

4.2 Implement `java.lang.Iterable<T>`

A data container type holding a collection of objects of type `T` can implement the `java.lang.Iterable<T>` interface. By doing so, the container guarantees to provide an object that is capable of *iterating* over its collection. The `Iterable` interface has one required method:

- `public Iterator<T> iterator()`

Make your `List` iterable by implementing this interface. The return type of `Iterable` is `Iterator<T>`, or to be accurate `java.util.Iterator<T>`. This type is itself an interface, which must be implemented by the object returned by `iterator()`.

Each time `iterator()` is called a new iterator object should be created. This iterator is one-use object, capable of iterating over the collection exactly *once*. It will need access to the list elements so that it can step through them and extract their content. This is an ideal situation for an anonymous inner class. Therefore, the object returned by `iterator()` should be defined by an anonymous inner class declared inside the `iterator()` method.

The `java.util.Iterator<T>` interface has two required methods:

- `public T next()`
- `public boolean hasNext()`

The `next()` method returns a different object of the collection each time it is called, and each object can only be returned once. Calling this method when no more collection objects can be returned is a runtime error, which should cause it to throw a suitable exception. Have a look at the ones available in `java.util`.

The `hasNext()` method returns true if the iterator has more objects to return, false otherwise.

4.3 Test Your Iterable List

Test your work by adding code to `ListProg.java` that iterates over a `List` using an *enhanced for loop*. This type of for loop is used with collections and has a simplified syntax that omits the counter variable. The general form is:

```
for(T t: collection) {
  // code using object t
}
```

where `collection` is an `Iterable` collection containing objects of type `T`. This loop will run once for each member of `collection`, and the variable `t` will be updated to be a different member of `collection` in each iteration. The loop operates by obtaining an `Iterator` and calling its `next()` and `hasNext()` methods.

5 And Finally

If you have completed the tutorial before the end of the session then use the time to make some notes on Java. You will almost certainly think of questions that have not been answered by the course. In this case, try to work out the answers for yourself by experimenting with the programs from the lectures or devising your own examples. If you get stuck the Java Tutorial is a very good resource.