COSC 222 Lab 6 – Iterables

In this lab you will create a class that implements the `Iterable<>` interface.

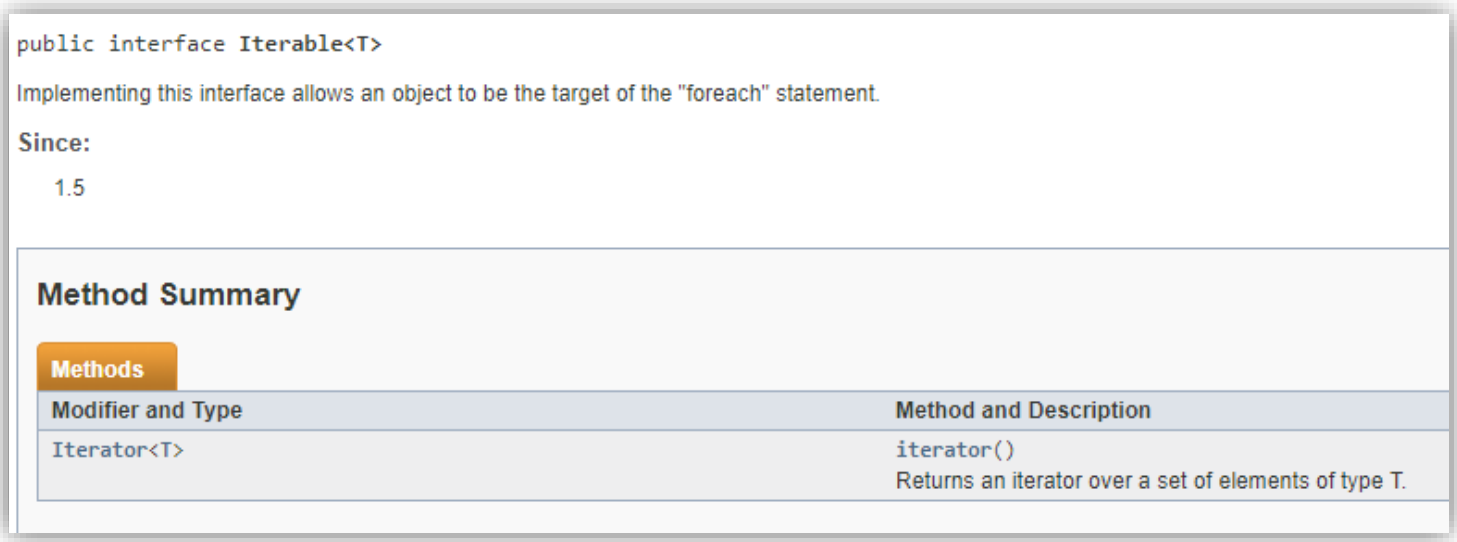
Consider the following scenario:

A company maintains 4 levels of customer loyalty: bronze, silver, gold, and platinum. When customers ask for service, they are handled first by their loyalty level (with Platinum being the highest priority) and within that loyalty level, on a first-come first-serve basis. To simulate this service the company provides, we will use an ArrayList of LinkedLists as the data structure. Your goal is to

1. implement an iterator for the data structure
2. write unit tests for the iterator

You are given a `Customer` class, which is complete and requires no editing. A customer has a name (String) and a loyalty index (0=Platinum, 1=Gold, 2=Silver, 3=Bronze).

The class `Requests` is where this ArrayList of LinkedLists is implemented, and Requests implements the `Iterable` Interface.



`public interface Iterable<T>`

Implementing this interface allows an object to be the target of the "foreach" statement.

Since:

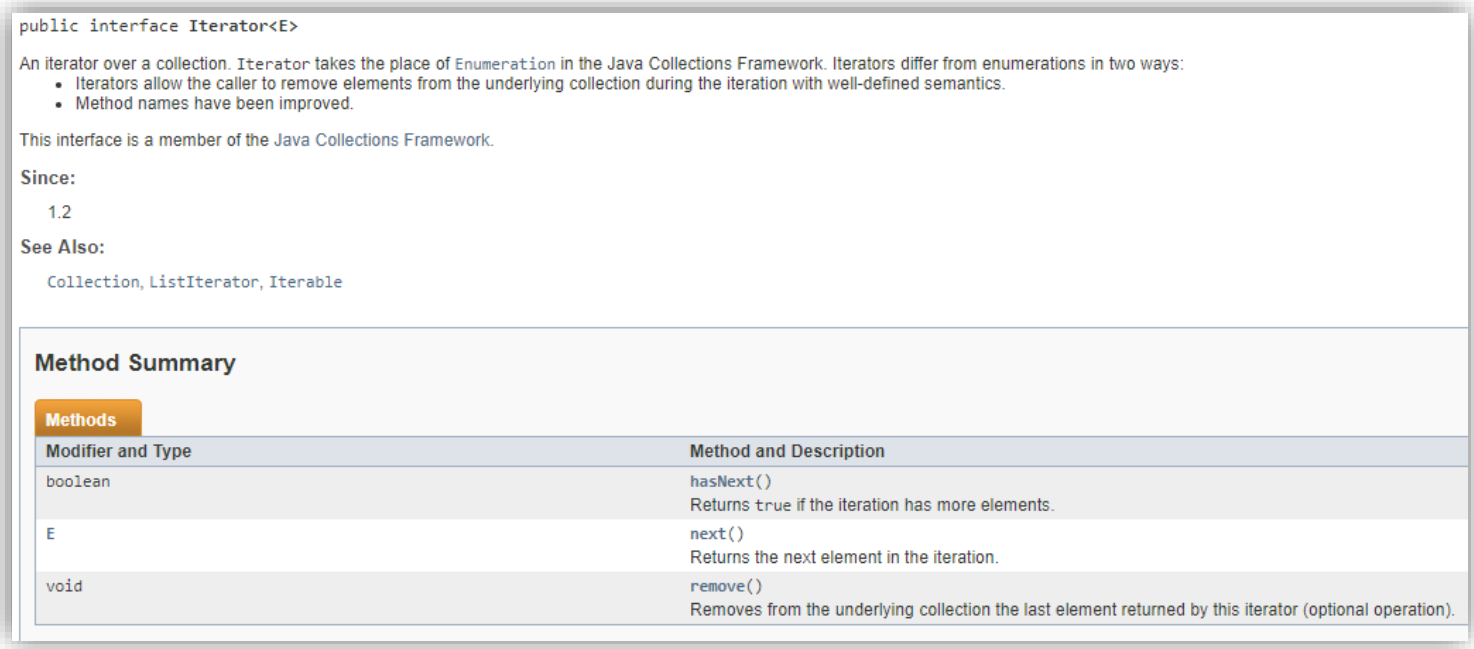
1.5

Method Summary

Methods

Modifier and Type	Method and Description
`Iterator<T>`	`iterator()` Returns an iterator over a set of elements of type T.

The only obligation a class has in order to implement the `Iterable` interface is that it must contain a method `iterator()` that returns an iterator. Now, to implement an interator for Requests, we can create an `Iterator<Customer>` inside `Requests` which implements the defining functions for an `Iterator`, which are `.hasNext()` and `.next()` (see image below).



public interface **Iterator<E>**

An iterator over a collection. `Iterator` takes the place of `Enumeration` in the Java Collections Framework. Iterators differ from enumerations in two ways:

- Iterators allow the caller to remove elements from the underlying collection during the iteration with well-defined semantics.
- Method names have been improved.

This interface is a member of the Java Collections Framework.

Since:

1.2

See Also:

`Collection, ListIterator, Iterable`

Method Summary

Methods

Modifier and Type	Method and Description
boolean	`hasNext()` Returns `true` if the iteration has more elements.
E	`next()` Returns the next element in the iteration.
void	`remove()` Removes from the underlying collection the last element returned by this iterator (optional operation).

There is also a `remove()` method, which is optional in Java, but our Lab Assignment will require it if you want to earn full marks. Solutions without `.remove()` completed will earn a maximum of 8/10.

The starter code has the structure of implementing these interfaces and points you to specific tasks with `//TODO` items. All your code edits are done in the `Requests.java` class.

Your iterator should be relatively efficient: that is, it should not have to scan through every item in every list every time it needs to extract the `.next()` item. (-1 point if your solution is inefficient in this way).

You must also test your code with Unit tests. A test is given to you in `Test.java`, and specific other tests are requests there with `//TODO` items.

A class `MainCustomers.java` is also given to you with a `main()` method to illustrate the usage of this class. Feel free to edit this file as you wish while testing your development.

Please submit any files you edit (`Requests.java` and `Test.java`, minimally).