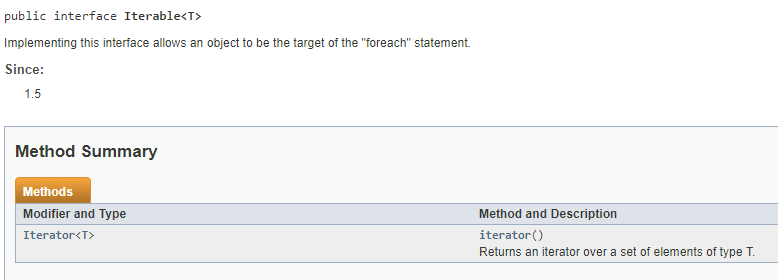
# COSC 222 Bonus Lab – Iterables

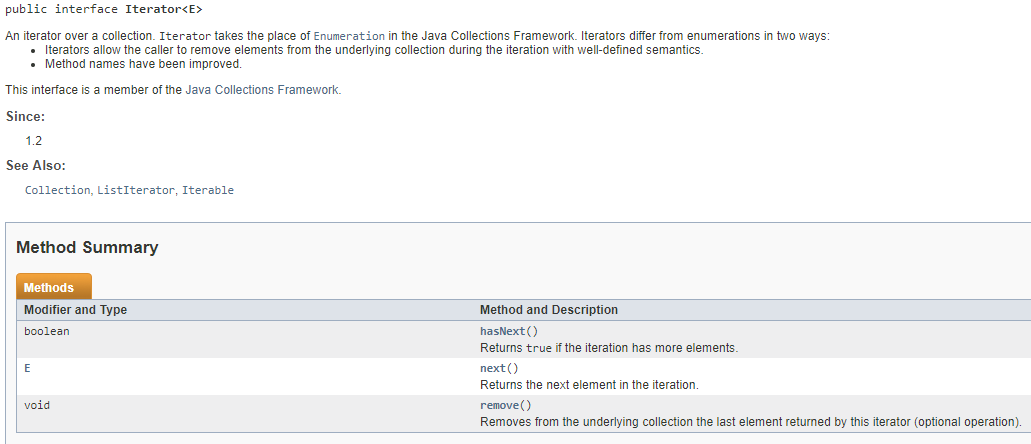
In this lab you will create a class that implements the Iterable<> interface.

Consider the following scenario:

|  |
| --- |
| A company maintains 4 levels of customer loyalty: bronze, silver, gold, and platinum. When customers ask for service, they are inserted into a special queueing system that handles customers 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 ArrayLists 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.

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 iterator for Requests, we can create an inner class Iterator<Customer> inside Requests which implements the defining functions for an Iterator, which are .hasNext() and .next() (see image below).

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 required in the Requests.java class (and the accompanying test class).

Your iterator should be efficient: that is, it should not have to scan through every item in every list every time it needs to extract the .next() item. It should run in O(1) time. (-2 points if your solution does not run in O(1) time).

You must also test your code with Unit tests. Several tests are given to you in TestRequest.java, and specific other tests are requested 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.

Submit any files you have edited. As before, all your files should have your name and student number in a comment at the top of the file (-1 if missing).