CS 2112 Lab: Iterators
What is an Iterator?

- In OOP, an iterator is an object that will allow you to traverse and access each element in a collection.
- Data structures can get very large: an iterator only needs to store the current element, and any information needed to find the next element.
- Want to avoid duplicating work – that defeats the purpose of an iterator!
What is an Iterator?

Most data structures have their values ready to access at any time repeatedly. An iterator can only bring up its values in the order they are assigned. That, and only once.

Basically, an iterator has a series of things that it would like to return in some order.

Figure: http://www-inst.eecs.berkeley.edu/~cs61a/su14/
The Iterator Interface

In Java, an Iterator is simply an object that implements the Iterator interface. This interface contains two methods:

- `next()` – Returns the next element in the iteration
- `hasNext()` – Return true if the iteration has more elements

For more complex data structures, iterators usually need to store information about the state of the iteration in order to ensure that the correct object is always returned by `next()`, independent of any calls to `hasNext()`.
A Simple Example

How would you implement an iterator for a linked list?
A Harder Example

How would you implement an iterator for a tree? Think about:

- How can I impose an order on the tree?
- How can I avoid duplicating work? (i.e. getting the elements from the original data structure more than once)
- How can I keep track of the state between calls?
Group Exercise

Given the skeleton code for `IterableStream` on the website, implement an iterator for `InputStream`.