Lab 08: Iterators

CS 0445: Data Structures

TAs: Jon Rutkauskas and Brian Nixon

http://db.cs.pitt.edu/courses/cs0445/current.term/

Nov 4, 2019,
University of Pittsburgh, Pittsburgh, PA

A Quick Note on Generics

```
public class MaxList<T extends Comparable<? super T>> extends ListInterface<T>
{
        public T getMax(){
                //...
        }
}
```

- Imagine a list where you want to get the biggest element in it
- Java has an interface to allow comparisons of objects: Comparable
- We need to specify that our data structure can only store objects or subclasses of objects that are comparable to each other

What is an Iterator?

- An object that allows us to sequentially traverse our data structures easily
- Can be written to make this traversal as efficient as possible
  - E.g., traversing a Linked List, but not having to start at the head node each time to get the next element
- Each iterator operates independently of any others
  - Can have multiple iterators at different parts of the data structure

How to use an iterator

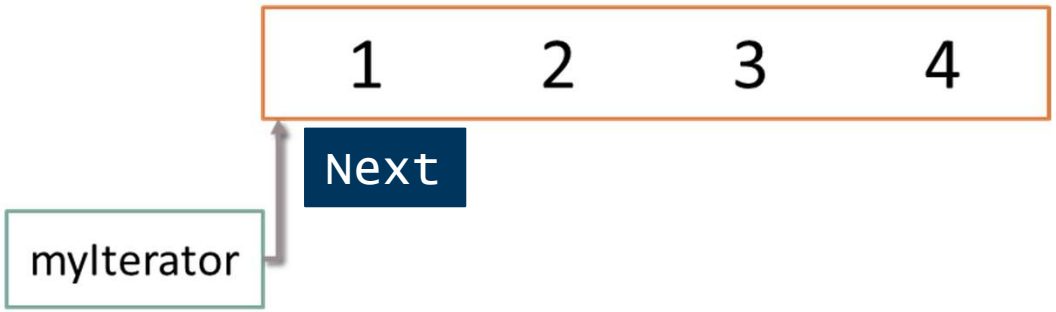
- A data structure that can give you an iterator has a `.iterator()` method. This returns an iterator object that `implements Iterator<T>`
- The iterator has the following methods:
  - `public boolean hasNext()`
    - Returns true if the iterator has more elements
  - `public T next()`
    - Returns the next element
  - `public void remove()`
    - Removes the last element returned by this iterator from the original collection

How an iterator is typically used

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(1);
list.add(2);
list.add(3);
list.add(4);
```

How an iterator is typically used

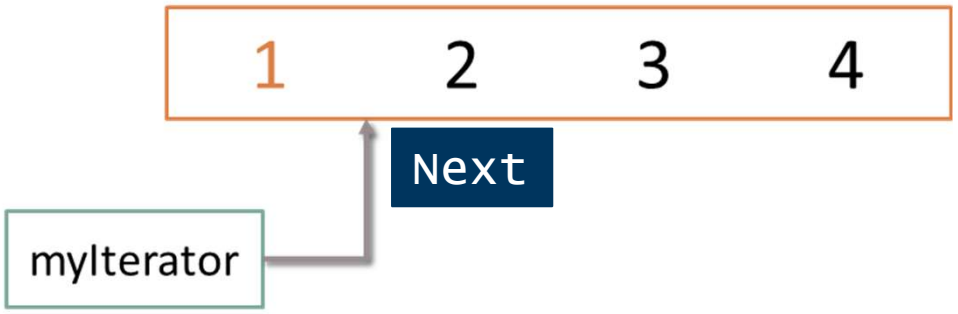
```
Iterator<Integer> myIterator = list.iterator();
```



How an iterator is typically used

```
while(myIterator.hasNext())
{
   Integer i = myIterator.next();
   doSomething(i);
}
```

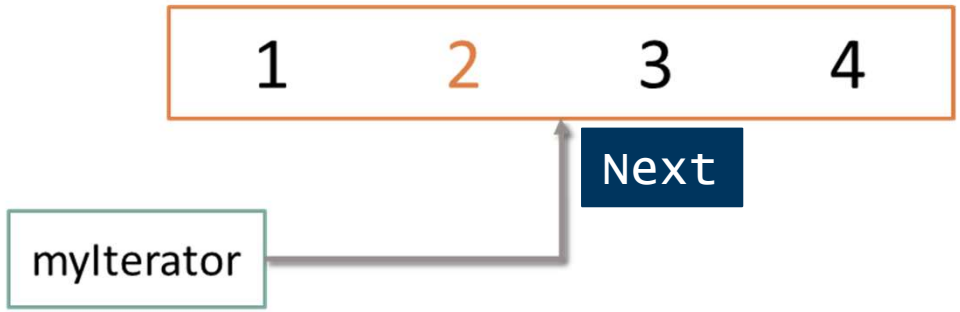
Iteration 1:



How an iterator is typically used

```
while(myIterator.hasNext())
{
   Integer i = myIterator.next();
   doSomething(i);
}
```

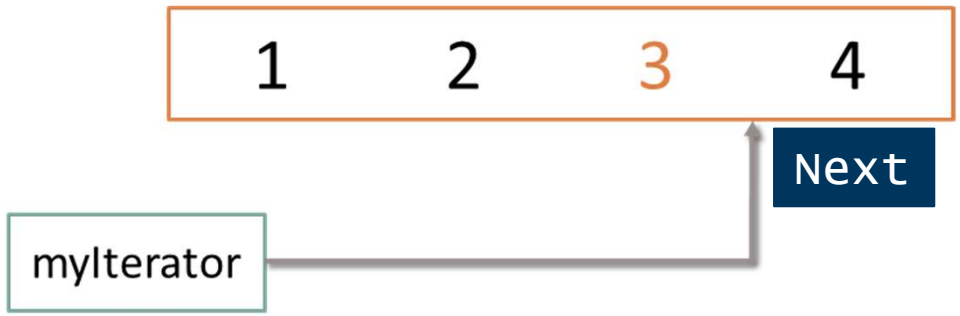
Iteration 2:



How an iterator is typically used

```
while(myIterator.hasNext())
{
   Integer i = myIterator.next();
   doSomething(i);
}
```

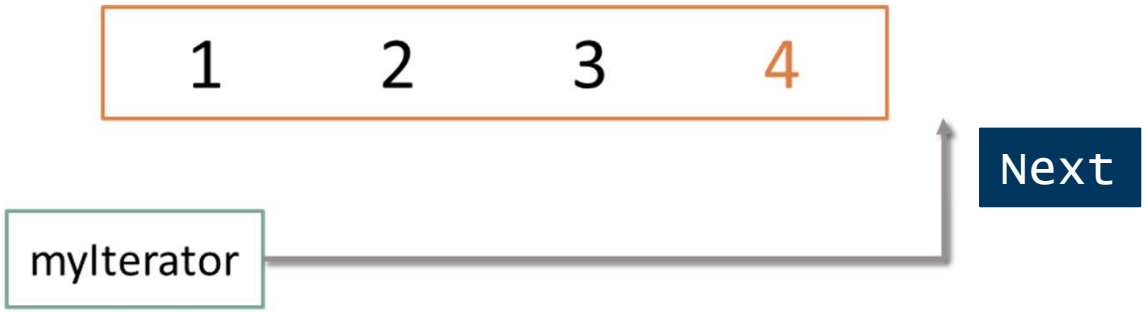
Iteration 3:



How an iterator is typically used

```
while(myIterator.hasNext())
{
   Integer i = myIterator.next();
   doSomething(i);
}
```

Iteration 4:



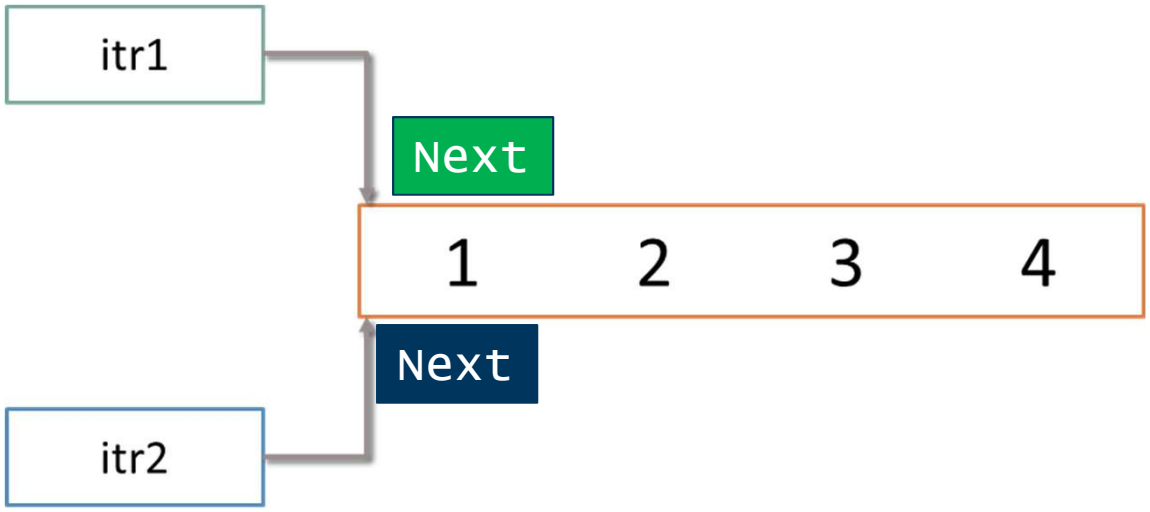
Using Two Iterators Simultaneously

- Since iterators work independently of each other, we can have two or more iterators for one collection.

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```

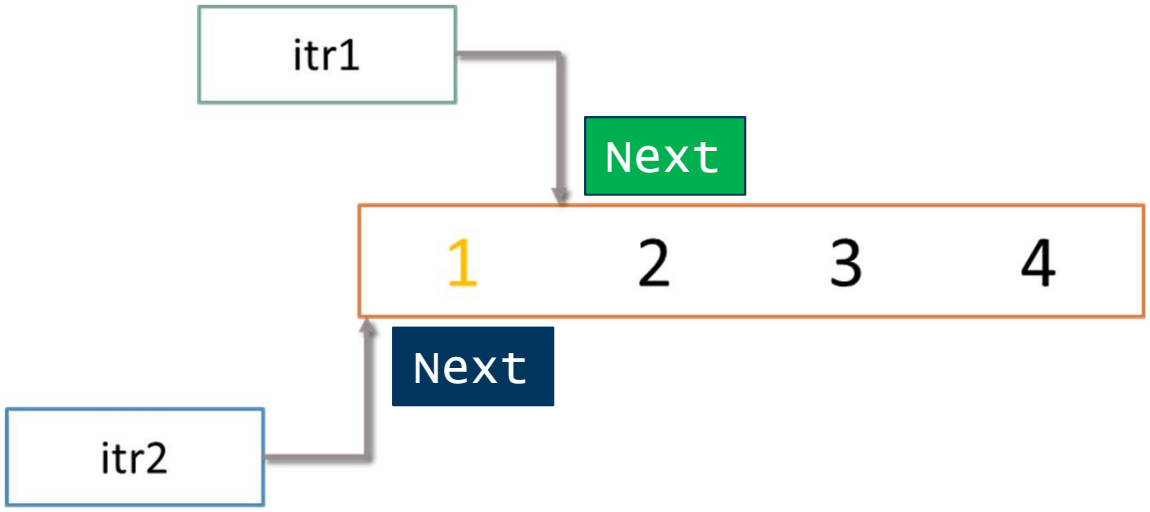
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



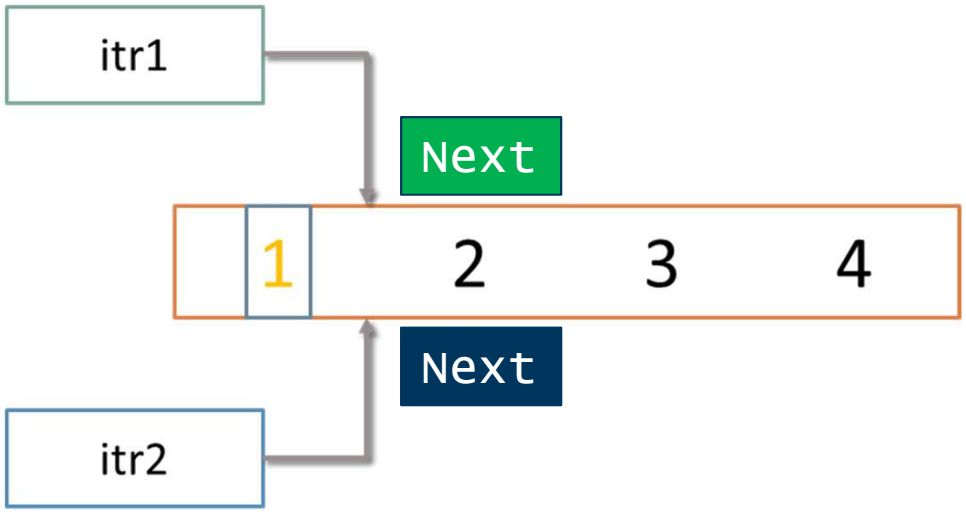
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



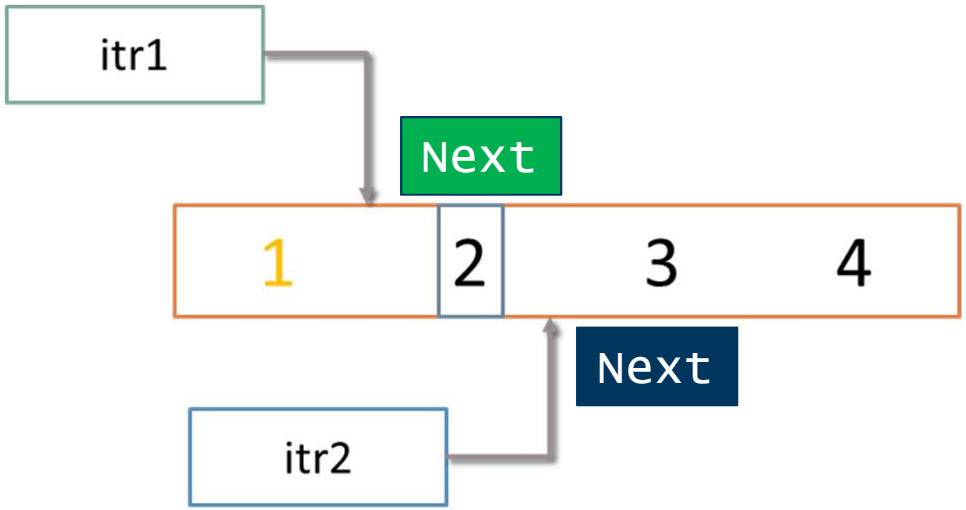
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



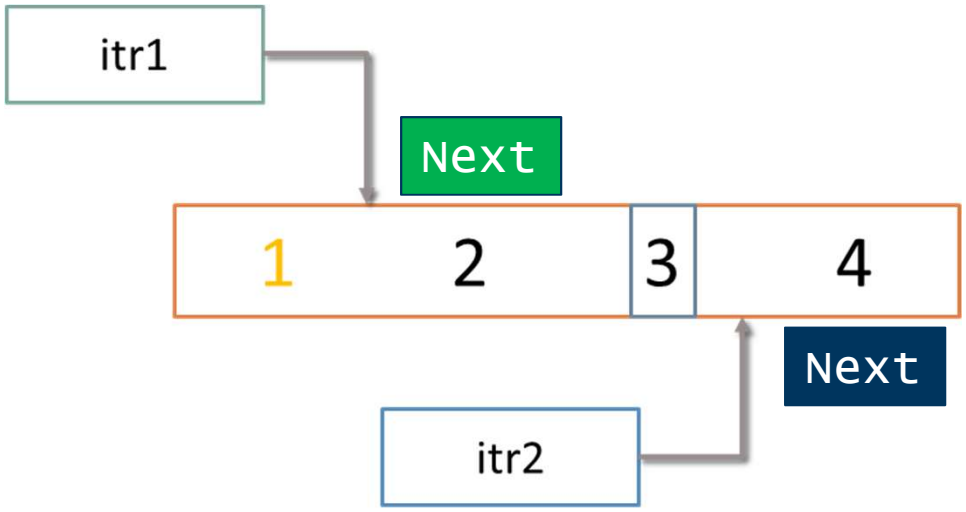
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



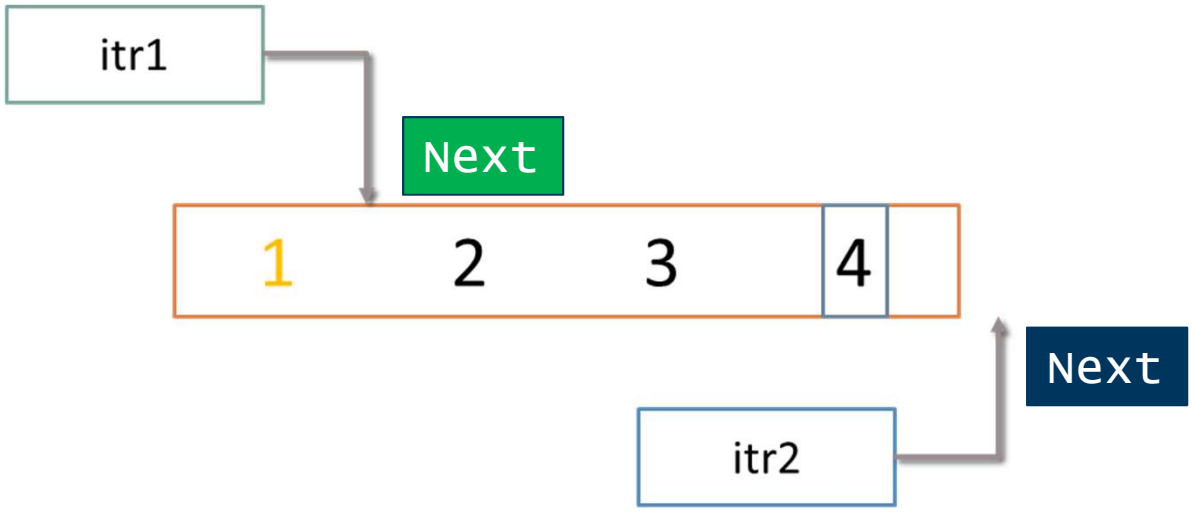
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



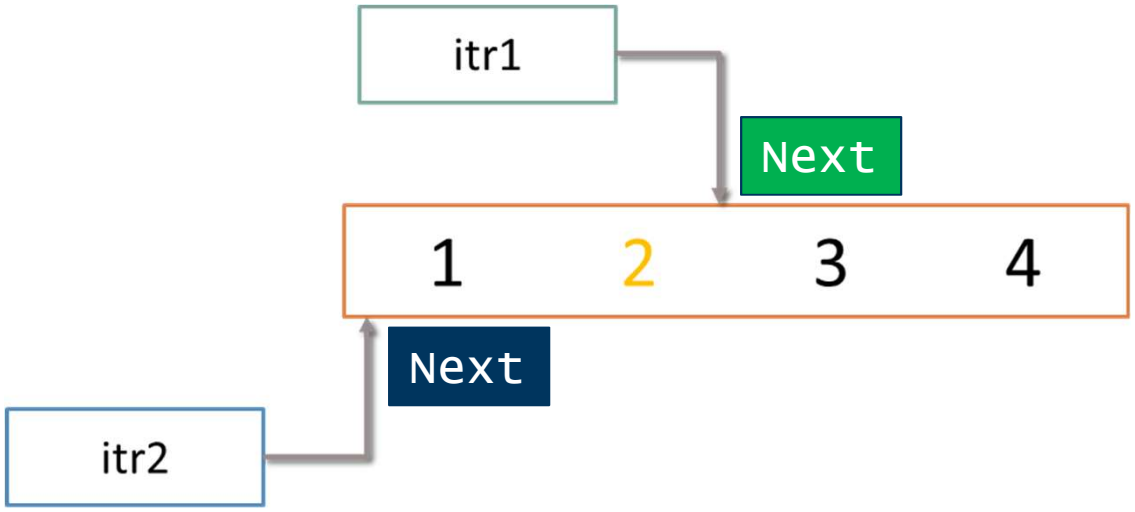
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



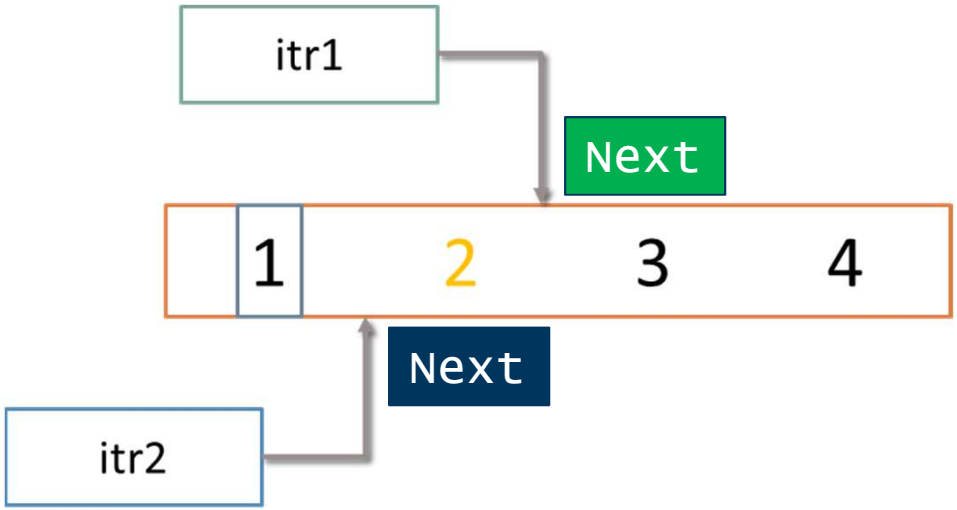
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



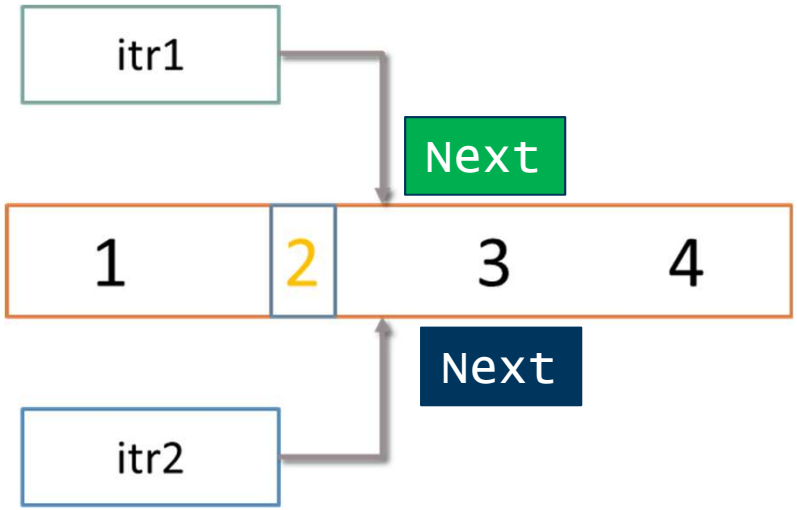
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



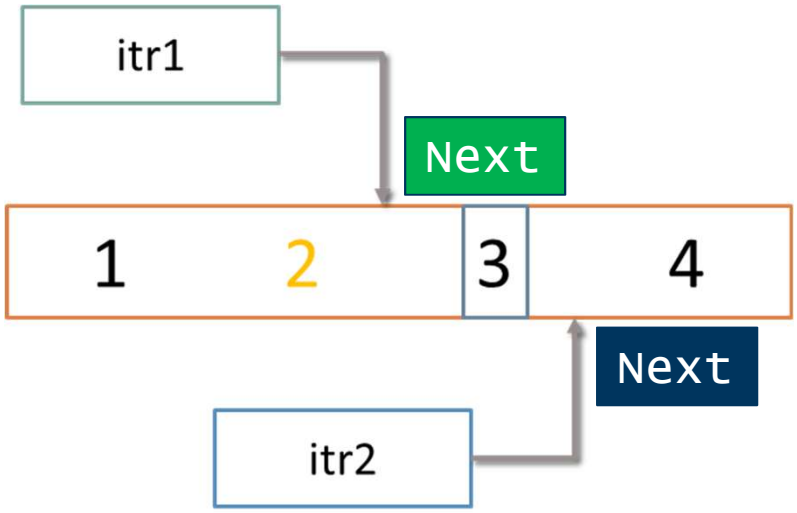
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



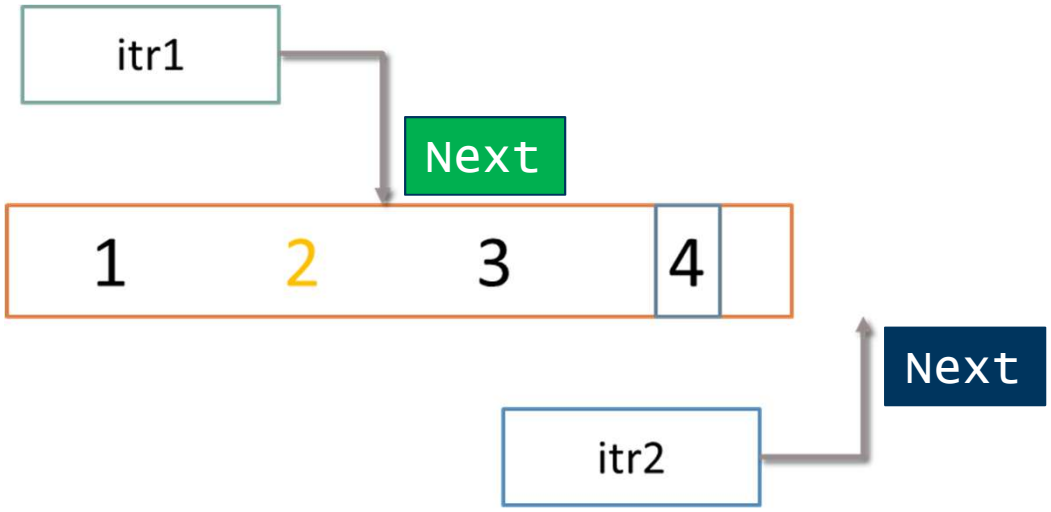
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



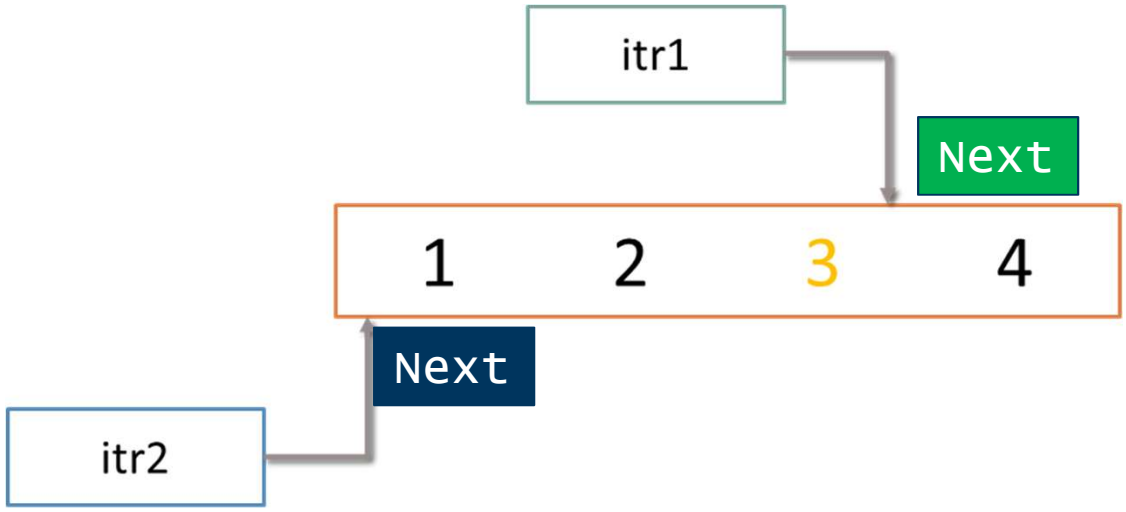
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



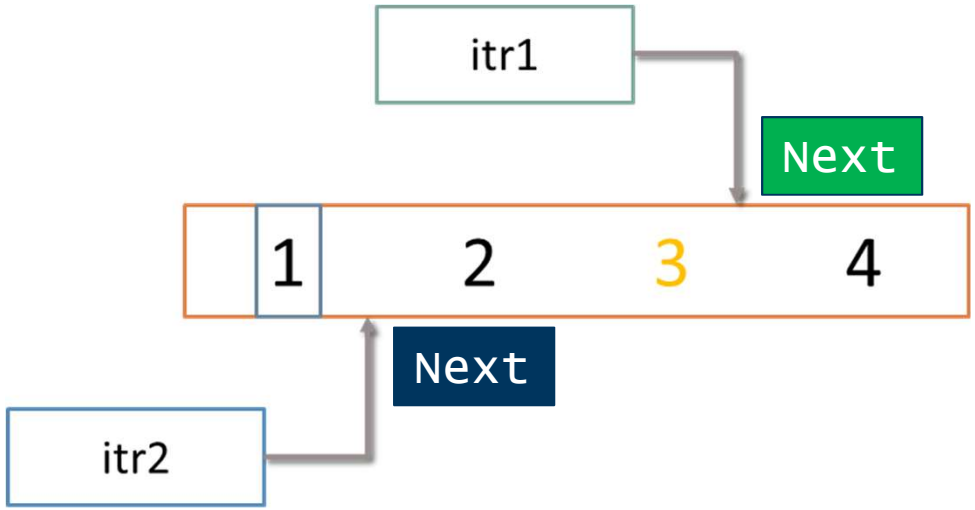
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



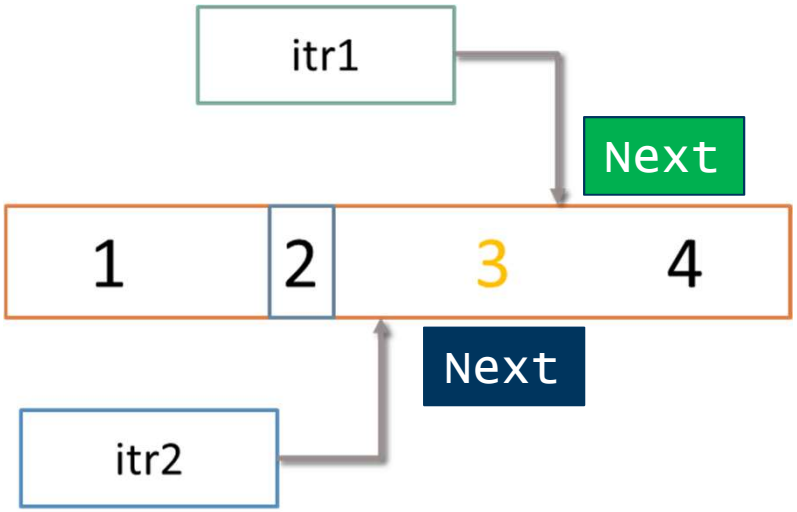
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



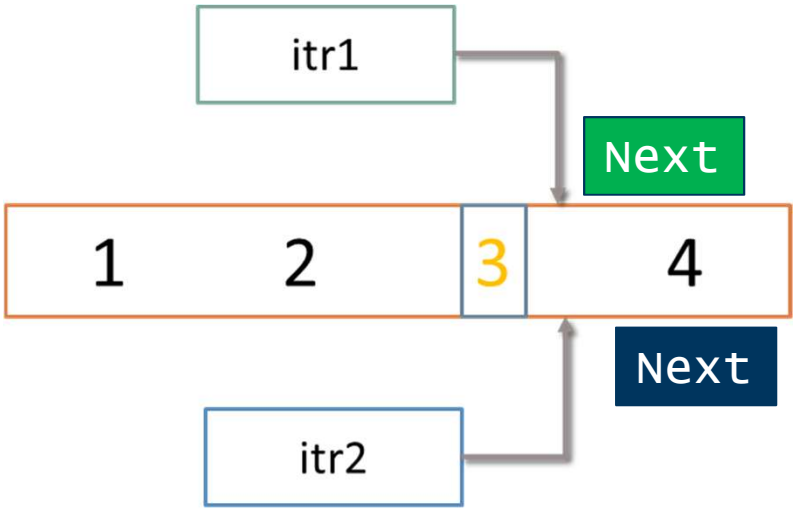
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



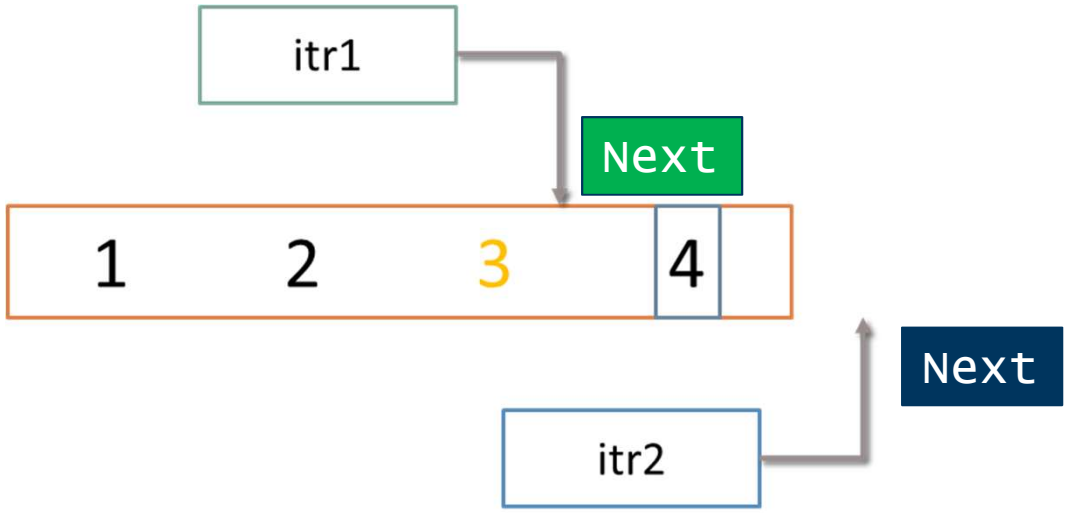
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



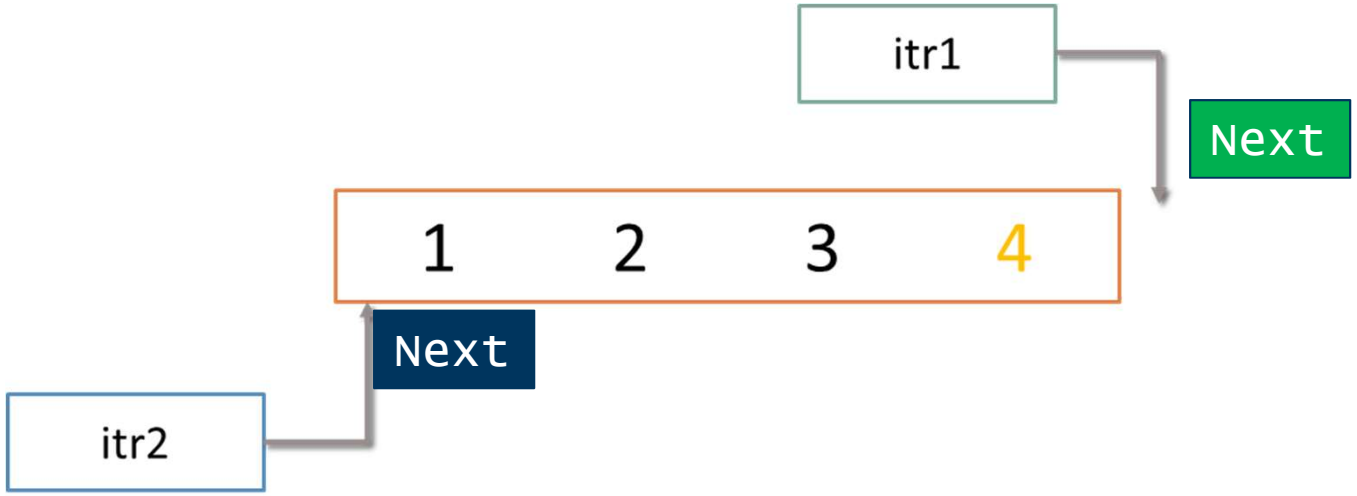
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



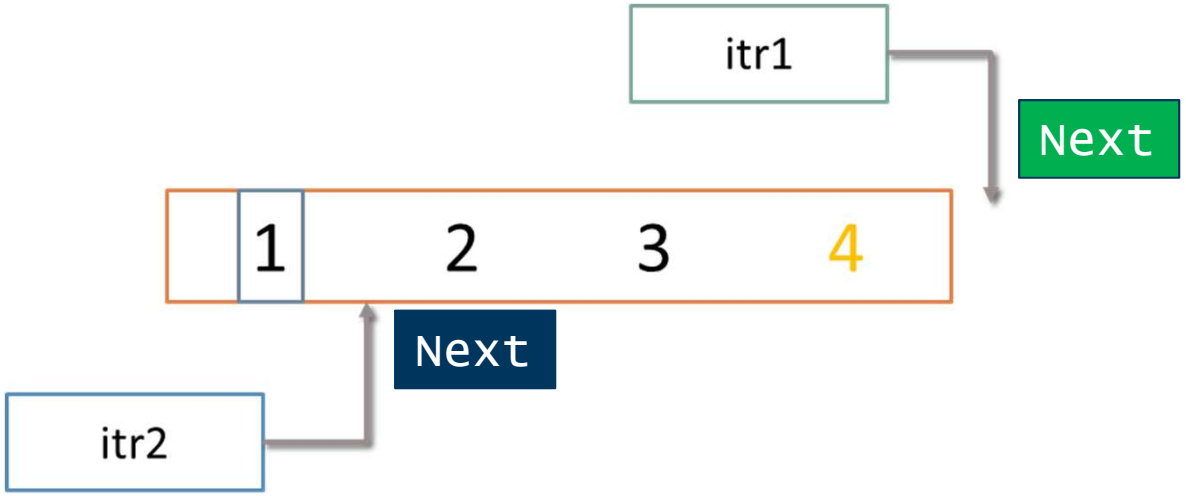
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



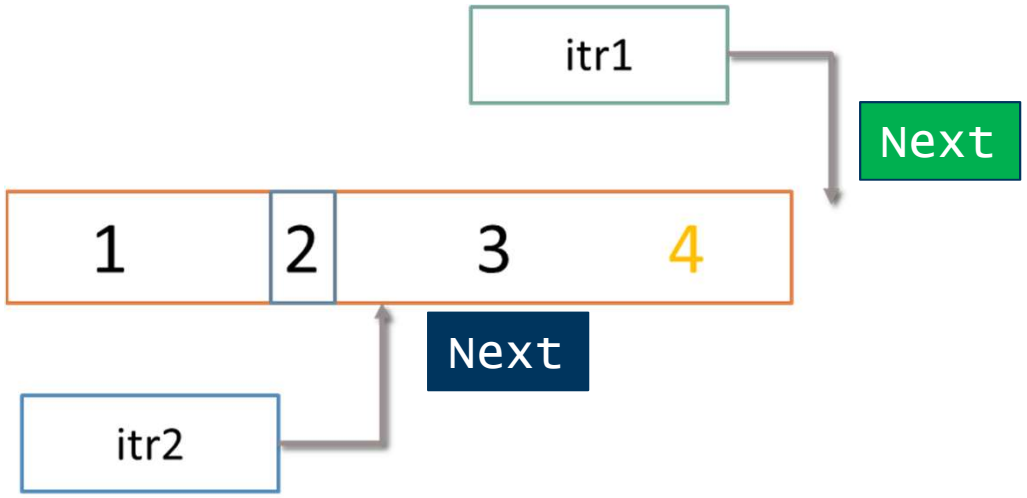
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



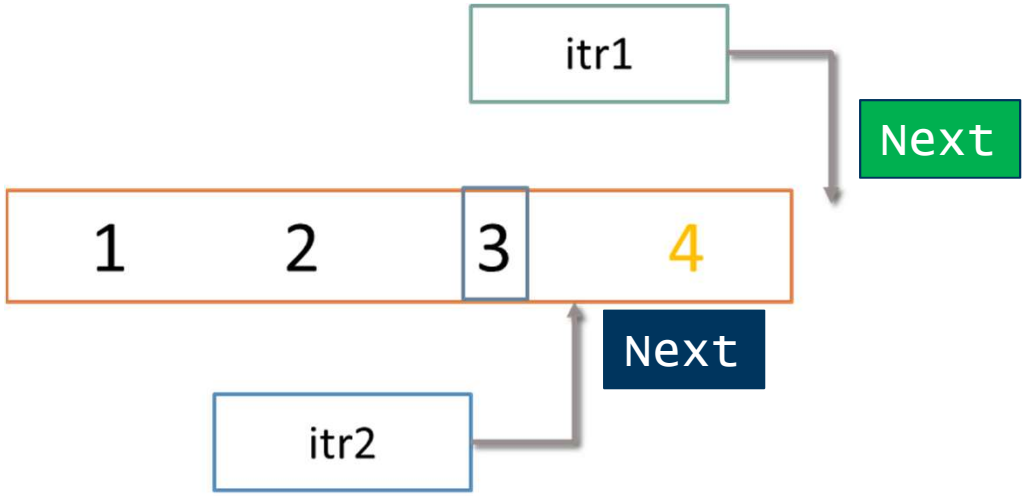
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



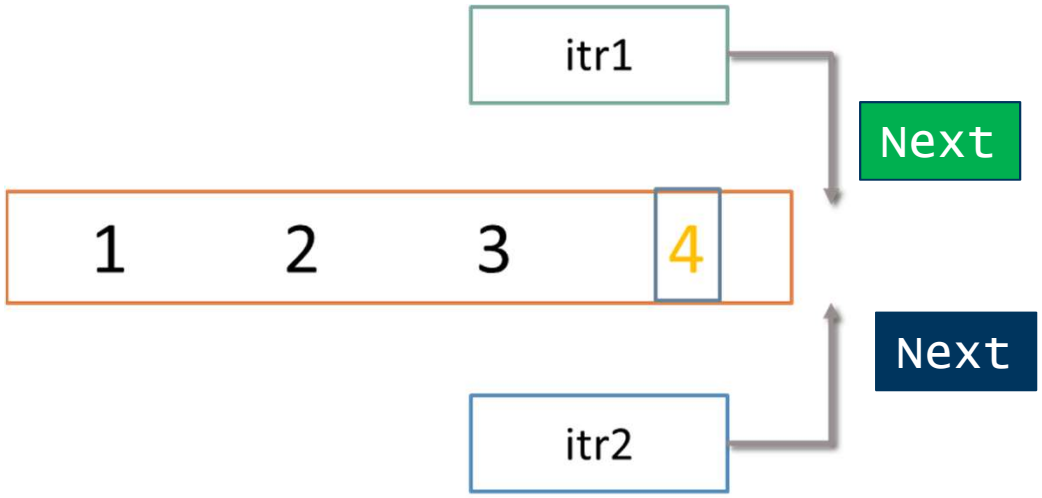
Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



Using Two Iterators Simultaneously

```
Iterator<Integer> itr1 = list.iterator();
while(itr1.hasNext())
{
        Integer i = itr1.next();
        Iterator<Integer> itr2 = list.iterator();
        while(itr2.hasNext())
        {
                Integer j = itr2.next();
                doSomething(i, j);
        }
}
```



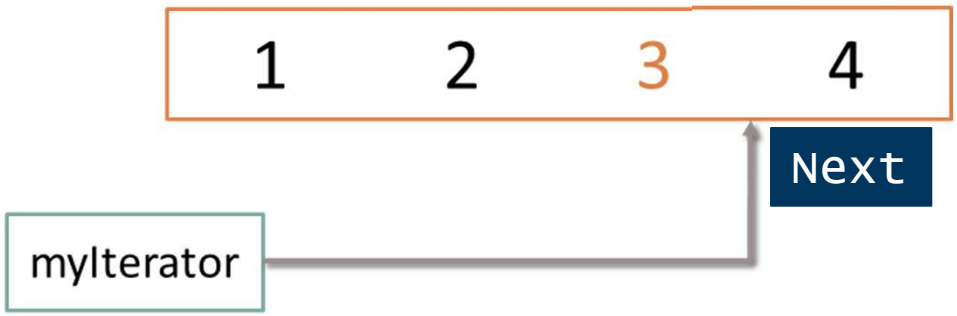
Using .remove()

- Remove is an optional method for iterators
- If implemented, removes the last item that was returned from .next() from the original collection.
- Example, removing all 3s from a list:

```
Iterator<Integer> itr = list.iterator();
while(itr.hasNext())
{
        Integer i = itr.next();
        if(i.equals(3))
                itr.remove();
}
```

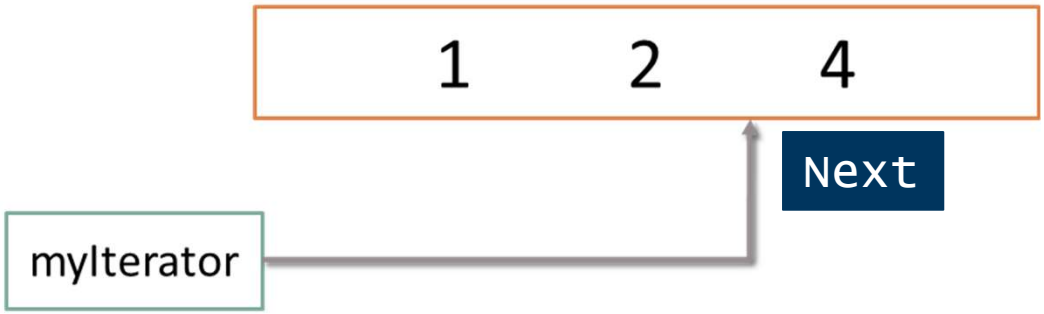
Using .remove()

```
Iterator<Integer> itr = list.iterator();
while(itr.hasNext())
{
        Integer i = itr.next();
        if(i.equals(3))
                itr.remove();
}
```



Using .remove()

```
Iterator<Integer> itr = list.iterator();
while(itr.hasNext())
{
        Integer i = itr.next();
        if(i.equals(3))
                itr.remove();
}
```



Lab

- Today's lab consists of two parts:
  1. Creating a method, using iterators, to print out a list, and creating another method to remove all strings shorter than a specified length from a list, again using iterators
  2. Reimplementing Sieve of Eratosthenes using iterators (no calls to .get())
- The files needed are on the website:
- http://db.cs.pitt.edu/courses/cs0445/current.term/
  - No need to implement the iterators themselves, the ArrayList provided already does
    - Just use the iterators