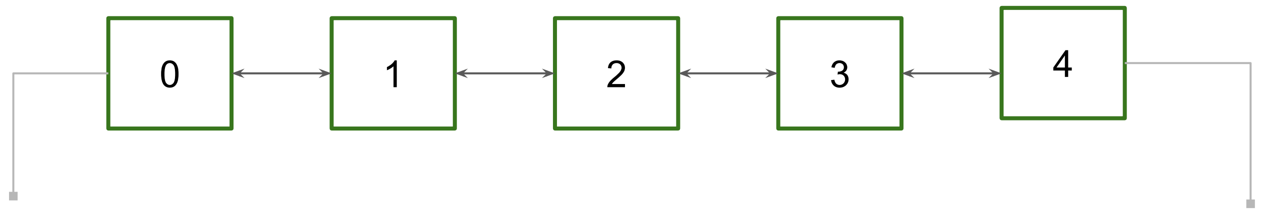
ICS-211 Lab

Assignment 4

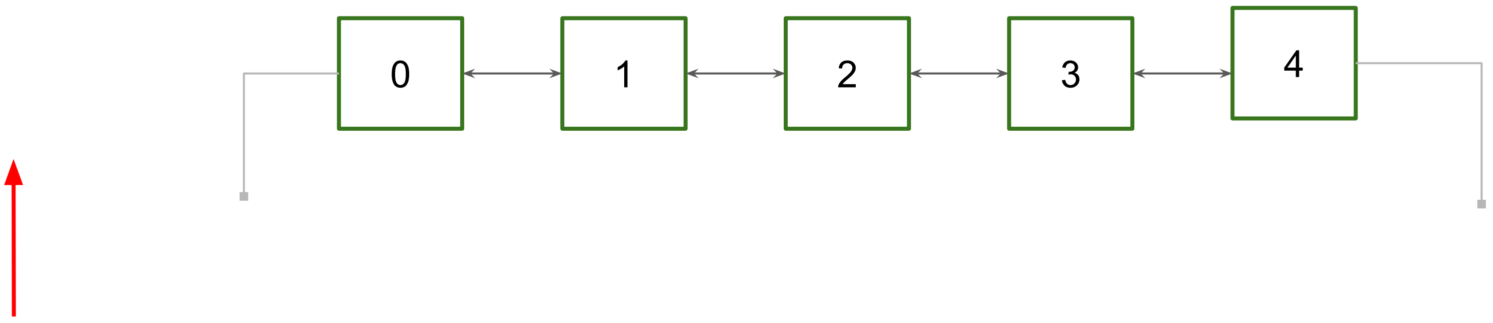
Assignment 4

- Unit tests in “a4” directory
- Same tests can be used for array list (need to change `MyLinkedList` to `MyArrayList`)
- The iterator tests may expose bugs in your list implementations
- `MyLinkedList` and `MyArrayList` must implement the `Iterable` interface
  - Necessary for use with special `for` loop syntax
  - The `iterator()` method returns an instance of your `ListIterator` class
  - `ListIterator` class ideally a nested class (but could be implemented differently)
-

Iterator Concept Example



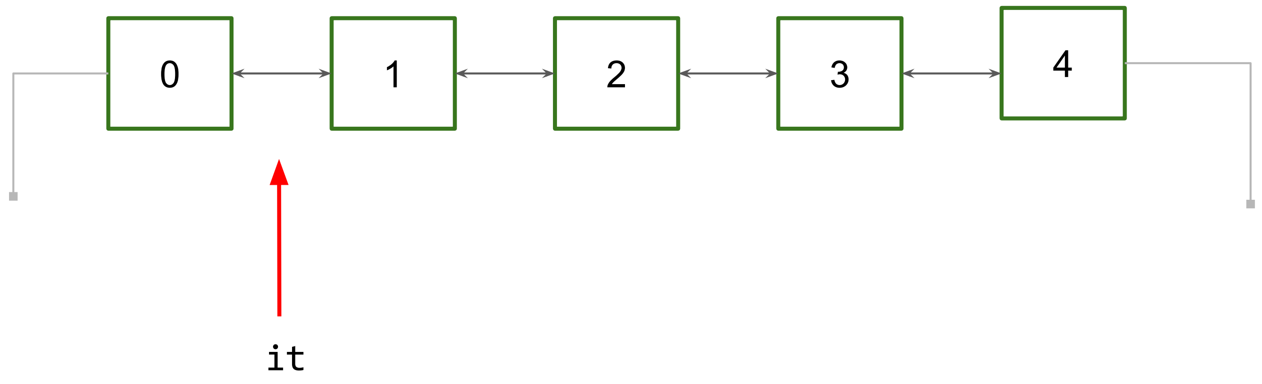
Iterator Concept Example



it

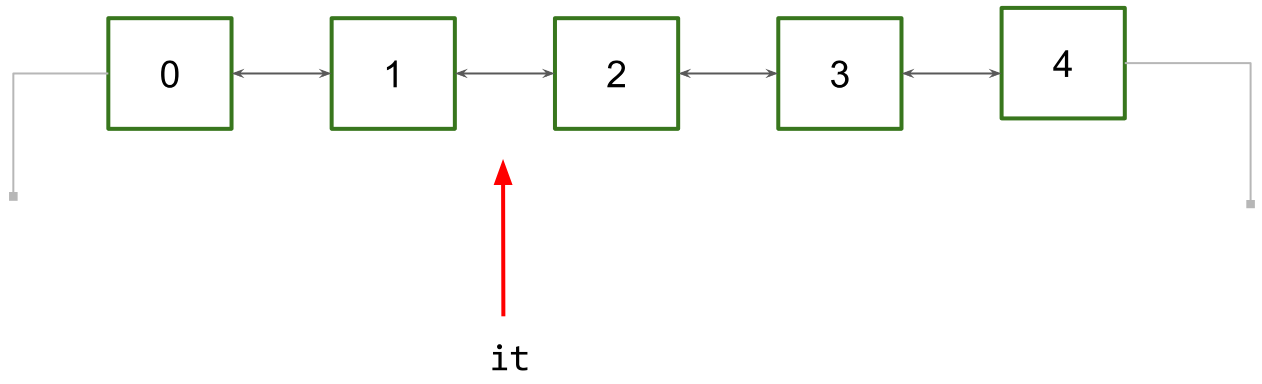
```
1.   it = list.iterator();   // it.hasNext() == true
```

Iterator Concept Example



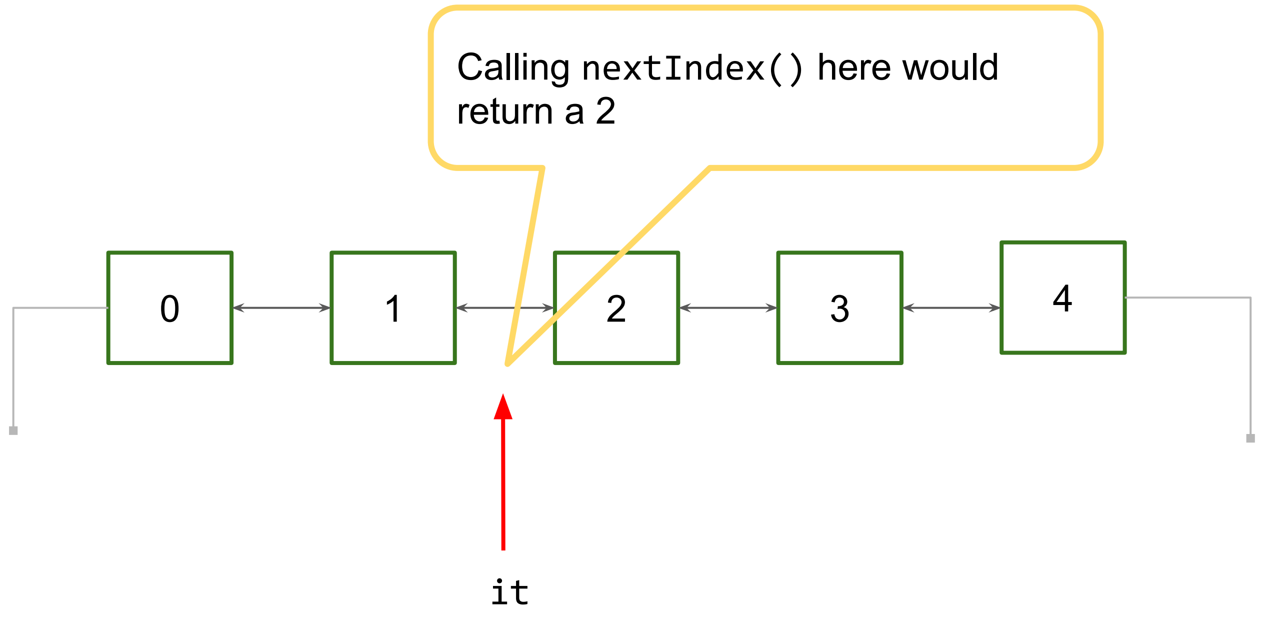
```
1.   it = list.iterator();   // it.hasNext() == true
2.   it.next();              // returns “0”
```

Iterator Concept Example



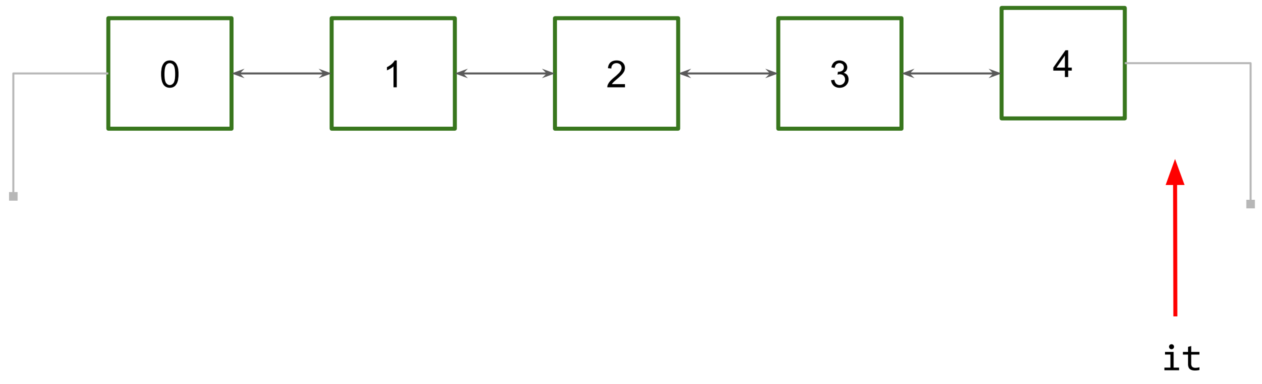
```
1.   it = list.iterator();   // it.hasNext() == true
2.   it.next();              // returns “0”
3.   it.next();              // returns “1”
```

Iterator Concept Example



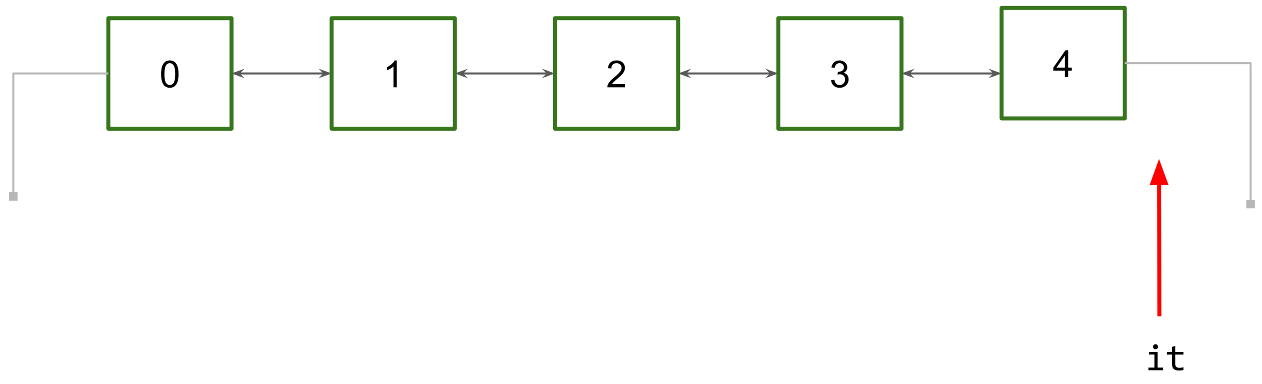
```
1.   it = list.iterator();   // it.hasNext() == true
2.   it.next();              // returns “0”
3.   it.next();              // returns “1”
```

Iterator Concept Example



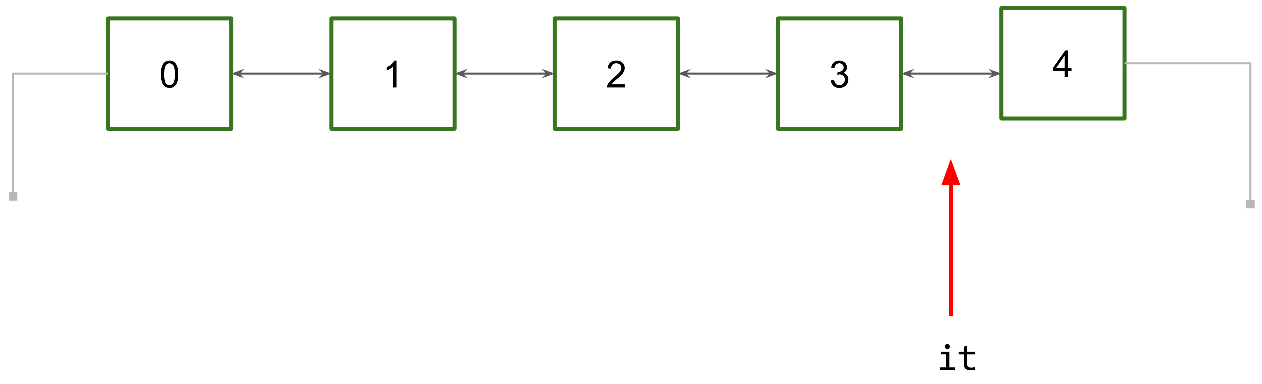
```
1.   it = list.iterator();   // it.hasNext() == true
2.   it.next();              // returns “0”
3.   it.next();              // returns “1”
4.   Keep going until “hasNext()” returns false
```

Iterator Concept Example



```
1.   it = list.iterator();   // it.hasNext() == true
2.   it.next();              // returns “0”
3.   it.next();              // returns “1”
4.   Keep going until “hasNext()” returns false
5.   hasPrevious() should return true
```

Iterator Concept Example



```
1.   it = list.iterator();   // it.hasNext() == true
2.   it.next();              // returns “0”
3.   it.next();              // returns “1”
4.   Keep going until “hasNext()” returns false
5.   hasPrevious() should return true
6.   it.previous()           // returns “4”
```

Announcements

- No lab next week
- I’ll still have office hours
- Understanding linked lists is important

Next Extra Credit (A05)

- Worth 10 points
- There will *not* be extra credit for A06
- Implement a generic stack class

```
class MyStack <E> {
        /**
         * Returns true if the stack is empty, false otherwise.
         */
        boolean empty()

        /**
         * Returns the top element of the stack.
         */
        E peek()

        /**
         * Add ‘item’ to the top of stack.
         */
        push(E item)

        /**
         * Removes top item on stack and returns it.
         */
        E pop()
}
```

- You can use your `MyArrayList` or `MyLinked` list.
- Java containers (i.e., `java.util.Stack`, `java.util.ArrayList`, etc.) ***can’t*** be used