NOTES Linked Lists

Computing Concepts, Java Essentials (Horstmann)

arrays aren't the best storage method

ex. company storing employees (alphabetizing, people leaving)

~when a new employee joins, moving everyone else in the array or ArrayList over takes a lot of time, energy, and memory space

~we want to minimize this!

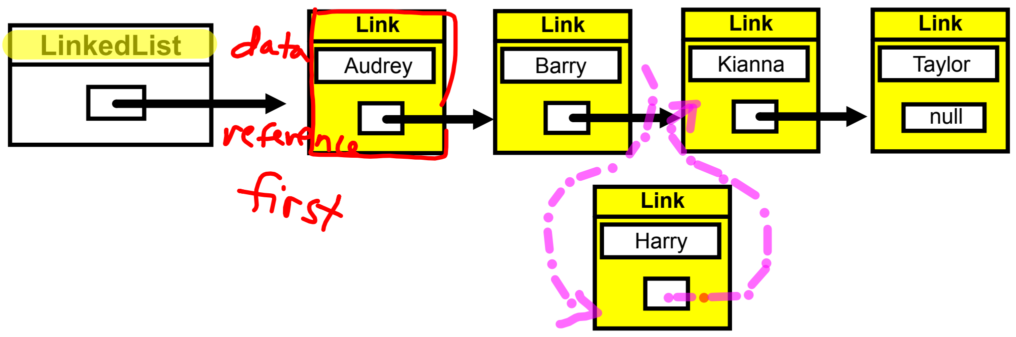
New Storage Structure:

~rather than storing the object references in an array, let us break the array into a sequence of links

link - stores an element and a reference to the next link in the sequence

NEW DATA STRUCTURE:

linked list - when you insert a new element into a linked list, only the neighboring link references need to be updated when you add or remove a link



~good for speedy insertion and removal of links

~bad because element access is slow.. to find the 5th element, you have to traverse first 4 links to get to it

Implementing Linked Lists

--> let's look at the structure of the Linked List Class

```
class LinkedList
{
   private class Link    //private inner class of a Linked List
   {
      public Object data;   //these methods in the class don't
      public Link next;     //return a Link object, so it is
                            //safe to make this instance
                            //variable public
   }
} //the LinkedList class holds a reference to first object or
   //the first link (or null if the list is totally empty)
```

LinkedList --> import the java.util package

```
void addFirst(Object obj)
void addLast(Object obj)
Object getFirst()
Object getLast()
Object removeFirst()
Object removeLast()
```

encapsulation

~you don't have access to the link's references (so you don't mess them up!!)

(don't break the links!)

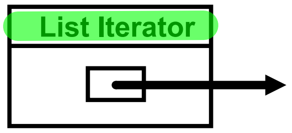
(THIS IS NOT TRIVIAL!)

So how do you access different elements in the linked list?

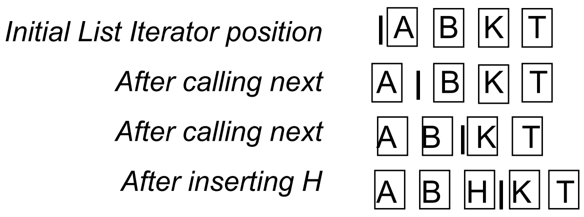
<u>iterator</u> - pointing between 2 links, like a cursor between 2 letters

So how do you access different elements in the linked list?

iterator - pointing between 2 links, like a cursor between 2 letters



Here are the employees from our first list:



```
LinkedList list = ... ; // we will do this extensively soon
ListIterator iterator = list.listIterator();
      //points before the element
iterator.next(); //moves iterator to the next position
   //will throw a NoSuchElementException if you move past the
   //end of the list
//to combat this, do this (always!)
if(iterator.hasNext())
   iterator.next();

//traverse all elements
while(iterator.hasNext())
{
   Object obj = iterator.next();
   //do something with the object
}
```

doubly linked lists - Linked Lists stores next and previous

```
previous();
hasPrevious();
```

$\}$ methods

*Add an object after the iterator, then move the iterator

```
iterator.add("Harry");
```

*Remove an object --> deletes object that was returned after the `next()` / `previous()` call

```
while(iterator.hasNext())
{
   Object obj = iterator.next();
   if(obj //fulfills condition)
      iterator.remove();
}
```

//can only use remove ONCE after previous/next

//can't use immediately after add

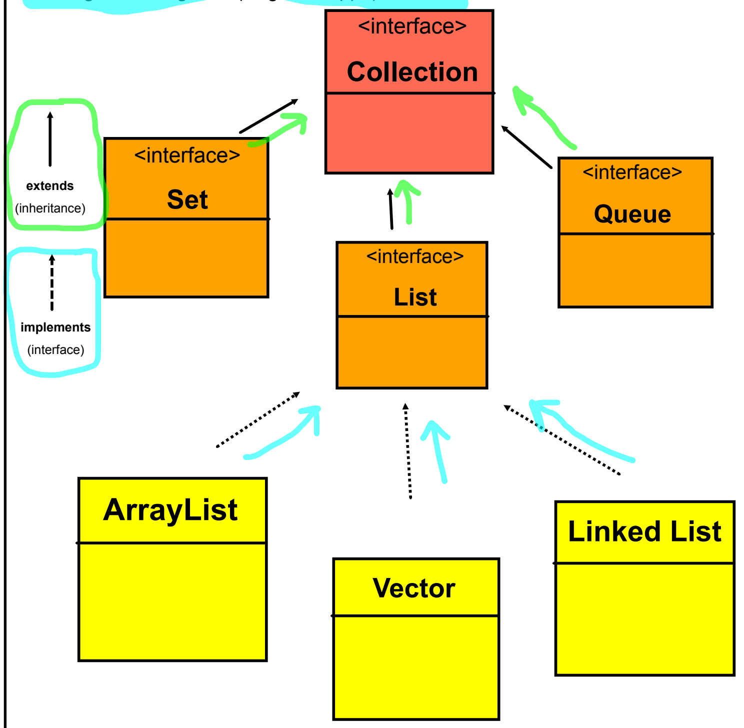
//IllegalStateException

```
FileListTest.java //demonstrates LinkedList class
```

EXAMPLE - page 741

NOTES Lists (in general)

Building Java Programs (Reges, Steppe) - Ch. 11



collection - an object that stores a group of other objects called elements

lists - ordered collection of elements, often accessed by integer indexes or by iteration

Linked List - a collection that stores a list of elements in a small object containers called nodes, which are linked together

iterator - an object that allows the efficient retrieval of the elements of a list in sequential order