Runtime of get in MutableList

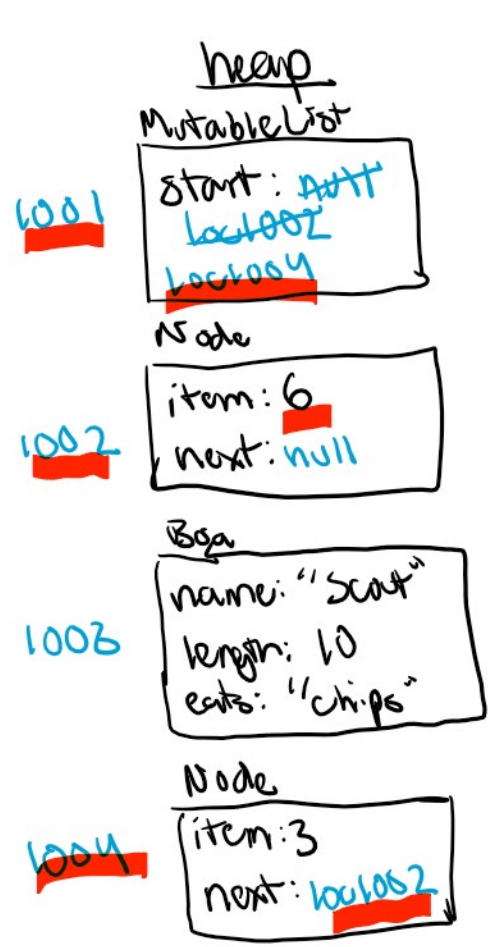
Tuesday, October 4, 2022 10:57 AM

```
public void Example2() {
    MutableList<Integer> L = new MutableList<>();
    L.addFirst(6);
    Boa teenBoa= new Boa("Scout", 10, "chips");
    L.addFirst(3);
}
```

environment

L → loc1001

teenBoa → loc1003



L in memory:
loc 1001
loc 1002
loc 1004

L.get(1): take this route through memory: 1001 → 1004 → 1002

get will be O(N) for MutableList because we have to follow the "nexts" through memory

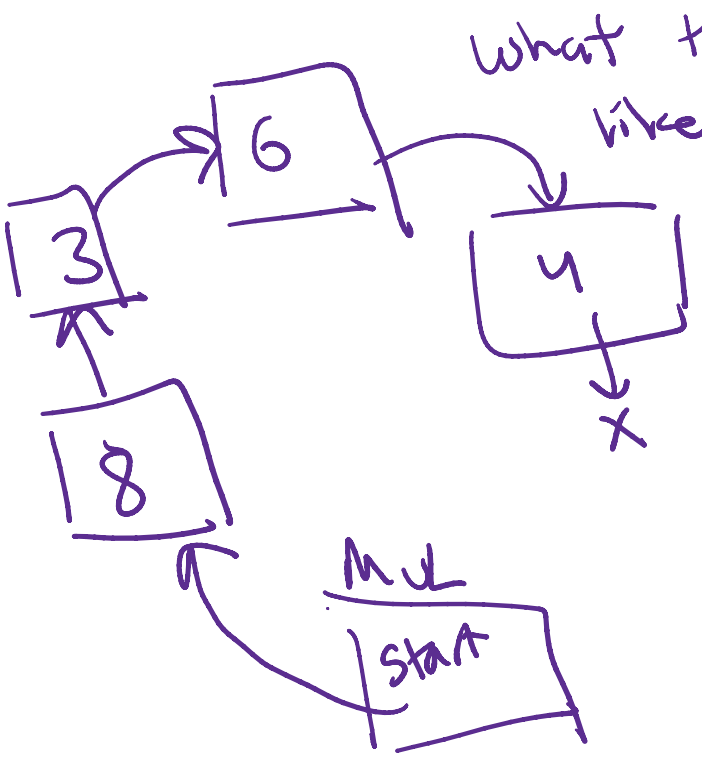
Memory Mystery

Wednesday, September 28, 2022 11:46 AM

loc 1012	**MutableList**(start:loc1017)
loc 1013	Node(item:6, next:loc1016)
loc 1014	Node(item:3, next:loc1013)
loc 1015	Dillo(length:5, isDead:true)
loc 1016	Node(item:4, next:null)
loc 1017	Node(item:8, next:loc1014)
loc 1018	

What code resulted in these values in the heap?

```
MutableList L = new MutableList();
L.addFirst(6);
L.addFirst(3);
Dillo myDillo = new Dillo(5, true);
L.addLast(4);
L.addFirst(8);
```



ArrayLists: addLast

Wednesday, September 28, 2022 11:46 AM

last used index

	Array (length 6)	index:
loc 1012	eltCount: ~~0~~ 1 2 .. 6 end: ~~0~~ 1 .. 5	
loc 1013	"we"	0
loc 1014	"could"	1
loc 1015	"all"	2
loc 1016	"use"	3
loc 1017	"arrays"	4
loc 1018	"!"	5

```
aL = new ArrayList(6) end = 0, eltcount = 0
aL.addLast("we");  end = 0, eltcount = 1
aL.addLast("could"); end = 1, eltcount = 2
aL.addLast("all");
aL.addLast("use");
aL.addLast("arrays");
aL.addLast("!");    end = 5, eltcount = 6
// now what if we wanted to addLast?
// --> next lecture!
```

get becomes constant time, because fetching a value from the heap once we know the exact memory address is constant time (at least for the purposes of 200; if you take later courses you'll learn a little more about the subtleties of memory access).

For an array, the element at index i will be at [location of array] + 1 + i, e.g. location of index 2 for the example array is at loc(1012 + 1 + 2) = loc1015.

Because we have the guarantee that the memory addresses for the array are contiguous, we can do this math and get an element at any index in constant time.