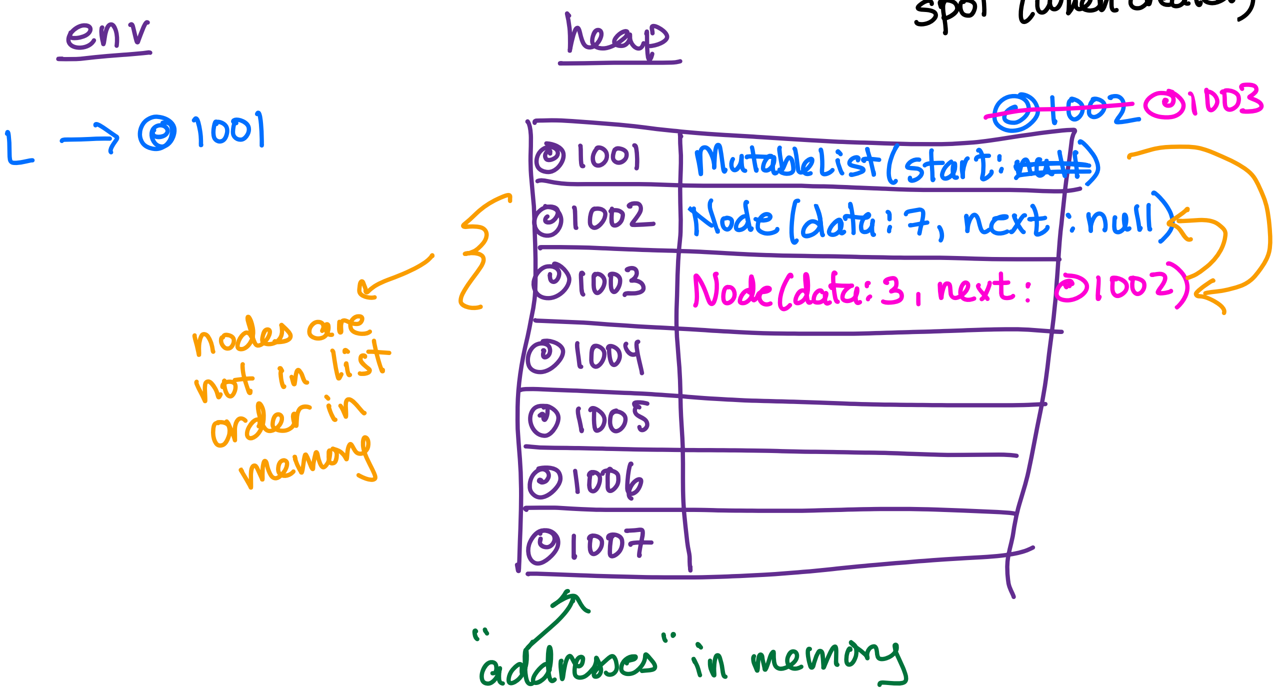
Lecture 10 – Addresses and ArrayLists

Lesson: Memory Diagrams with Addresses Explicit

```
// the list [3, 7]
MutableList<Integer> L = new MutableList<>();
L.addFirst(7);
L.addFirst(3);
```

each object goes into next avail spot (when created)



env

L → @1001

heap

~~@1002~~ @1003

@1001	MutableList(start: ~~null~~)
@1002	Node(data: 7, next: null)
@1003	Node(data: 3, next: @1002)
@1004	
@1005	
@1006	
@1007	

nodes are not in list order in memory

"addresses" in memory

Activity: Draw the memory diagram with addresses for the following program

```
public void Example2() {
    MutableList<Integer> L = new MutableList<>();
    L.addFirst(6);
    Course ai = new Course("CSCI 1410", 200);
    L.addFirst(3);
}
```

x = 2

fill in heap table for this program

env

L → @1007

ai → @1009

x → 2

primitive data (int, bool, str) and object locations

heap

all objects

@1010
~~@1008~~

@1007	MutableList (start: ~~null~~)
@1008	Node (data: 6, next: null)
@1009	Course (name: CSCI 1410, 200)
@1010	Node (data: 3, next: @1008)
@1011	
@1012	
@1013	

Activity: Memory layouts of lists

0 1 2 3

Consider the following layouts for the list [8, 3, 6, 4] – what program might generate this heap layout?

@1012	**MutableList**(start:@1017)
@1013	Node(item:6, next:@1016)
@1014	Node(item:3, next:@1013)
@1015	Course(name: “CSCI1410”, enrollment: 200)
@1016	Node(item:4, next:null)
@1017	Node(item:8, next:@1014)
@1018	

get(2)

get(1)

get(0) 2

M = new MutableList()
M.addFirst(6) or addLast(6)
M.addFirst(3)
new Course(---)
M.addLast(4)
M.addFirst(8)

M.get(2)

Question: How would this memory layout be different if we were making an *immutable* list with the same sequence of addLast/addFirst calls?

Question: Imagine this list were named `L` in the environment. What sequence of memory objects get visited to compute `L.get(2)` [which should return 6]?

Activity: Now imagine the list had the following layout in memory (all the items consecutive and in order). What sequence of memory objects would get visited to compute `L.get(2)`?

*

@1012	**ConsecList**
@1013	8
@1014	3
@1015	6
@1016	4
@1017	
@1018	

get(0)
get(1)
get(2)
get(3)

get(i) must be in
location
address-of-L + 1 + i

if could get all items in
consecutive, ordered
locations, operations like
get become constant time.

Arrays in Code

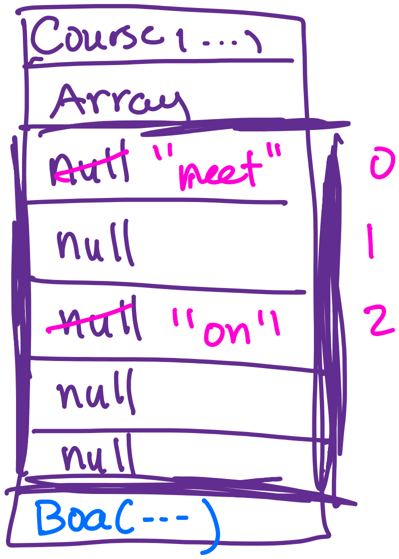
```
import java.util.Arrays;

public class Main {
   public static void main(String[] args) {
```

Array is a sequence of consecutive memory addresses with programmer support to access them.

new String[5];

new Boa(...)



arrays are pieces/chunks of memory. They don't have methods like add First, etc.

search for 10