Lecture 9 handout — Equality and Lists in Memory

Review the Environment and Heap – how can each be changed?

```
// formerly NodeList
public class Link implements IList<T>
{
    T first;
    IList<T> rest;
    ...

    public Link addFirst(T newElt) {
        return new Link(newElt, this);
    }

    public T getFirst() {
        return this.first;
    }
}
```

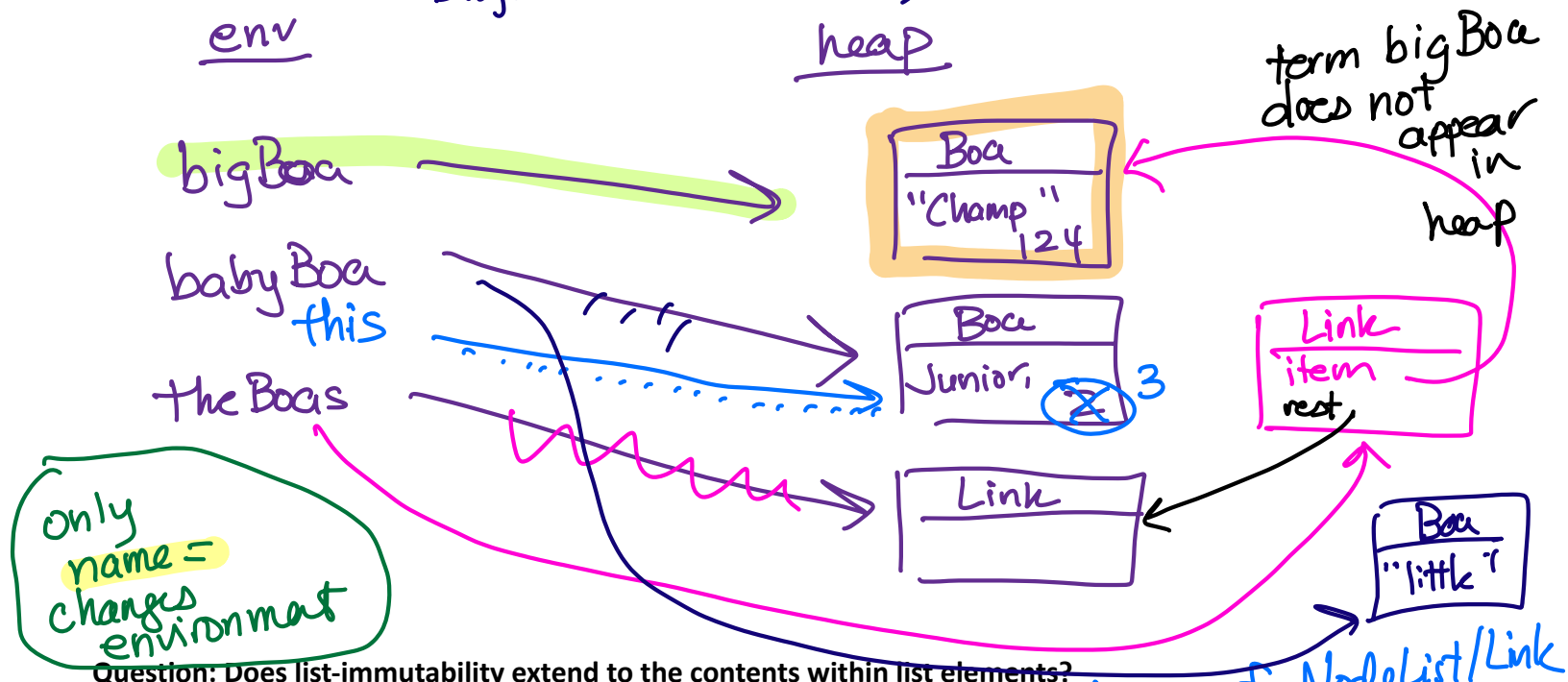
```
public class Boa {
    string name;
    int length;
    string eats;

    public growBy(int amt) {
      this.length = this.length + amt
    }
}
```

```
public void example() {
    Boa bigBoa = new Boa("Champ", 124, "peas");
    Boa babyBoa = new Boa("Junior", 2, "milk");
    Link<Boa> theBoas = new Link<>(); // Note: IMMUTABLE LIST
    theBoas = theBoas.addFirst(bigBoa);
    theBoas = theBoas.addFirst(babyBoa);
    babyBoa().growBy(1);
    System.out.println(theBoas.getFirst().length);
}
```

this.length = this.length +

babyBoa = new Boa("little")

change heap with new (add object) or obj.field =



Question: Does list-immutability extend to the contents within list elements?

No! immutability refers to structure or chain of NodeList/Link
- does NOT limit changes to objects inside the list.

Question: What does it mean for lists to be "the same"

```
public static void equalityExample() {
   MutableList<Integer> L1 = new MutableList<Integer>();
   L1.addFirst(6);
   L1.addFirst(8);
   System.out.println("L1 is " + L1);

   MutableList<Integer> L2 = new MutableList<Integer>();
   L2.addFirst(6);
   L2.addFirst(8);
   System.out.println("L2 is " + L2);


   // what do you expect each of these to produce? (what do == and .equals mean?)
   System.out.println(L1 == L2);
   System.out.println(L1.equals(L2));
   System.out.println(L1.toString() == L2.toString());
   System.out.println(L1.toString().equals(L2.toString()));
}
```

x = 5
y = 5
is x == y?

are these the same objects in heap?

the programmer of MutableList will tell me what equality means (can be flexible)

```
class Boa {
   String name;
   int length;
   String eats;
```

let's say only name matters for equality

```
public boolean equals(Object other) {
   if (other instanceof Boa) {
      return this.name.equals(((Boa) other).name);
   } else return false;
}
```

hey Java, I swear this will be a Boa

legal to write `babyBoa.equals("Bruno")`

Lesson: Writing .equals methods

To ask equals on Lists, need to define what it means at content level

```
class Link<T>
    T first;
    Link<T> rest;

    public boolean equals(Object other){
        return (this.first.equals(other.first)) &&
               this.rest.equals(other.rest))
    }
```

expect this is a list for now

Lesson: Memory Diagrams with Addresses Explicit

```
public class MutableList<T>     {
   Node<T> start; // front of the list

   public void addFirst(T newItem) {
       newNode = new Node<>(newItem, this.start);
       this.start = newNode;
   }
}

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

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

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