Lecture 10 – Addresses, Equality and ArrayLists

Lesson: Memory Diagrams with Addresses Explicit

```
// 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 addFirst(T newElt) {
   Node<T> newNode = new Node<T>(newElt, this.start);
   this.start = newNode;
}
```

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

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()));
}
```

@1020	MutableList(start: @1022)
@1021	Node(item: 6, next: null)
@1022	Node(item: 8, next: @1021)
@1023	MutableList(start: @1025)
@1024	Node(item: 6, next: null)
@1025	Node(item: 8, next: @1024)

Review: Continuing from last lecture–memory layouts of lists

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	

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	

Arrays in Code

```java
import java.util.Arrays;

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