**Linked Lists Workshop**

The aim of this workshop is to make students understand the fundamentals of Linked List.

We will do so by starting with a simplifying example, and continue with practicing basic functions with Node class and List class.

**Important notes:**

* The relevant knowledge of the students is up to, and including, lecture 9-2.
* Linked List class is now called “List” - but there’s no difference from what we’ve used and learned so far.
* The purpose of the workshop is to make students feel comfortable with writing code that involves nodes and lists.
* We assume that one of the harder points for students to grasp is the structure of each node, understanding why “next” is of type node.
* Another tricky thing for students is how to build and initialize a new linked list - we will help them implement a new list by themselves in this workshop.
* Please make sure to send the students the skeletons in advance so they can work on them during the workshop.
* If you want to include more exercises, you’re more than welcome to delete any function from the skeleton file and add it as an exercise in your workshop.

**Introduction to Node class - simplifying the structure of Nodes**

* Start with introducing the simplifying example in the presentation (People in line) -
  + Make sure to clarify the difference between the ID (equivalent to Value) and Person (Equivalent to Node)
  + This example is presented in order to give the students a better understanding of every Node’s structure, and to grasp the idea of a pointer.
  + It is crucial to make sure that every student understands the fields of an object of type node after this part of the workshop.
* Go through “Node.java” first constructor implementation.

**Ex #1**

Instructions:

Ask the students to write a constructor function for the last node in a list.

The function receives a value and sets the field “next” to be null.

Important Notes:

* It is most likely that a lot of students do not know the “this(value,null)” implementation and they will write code that “repeats itself”.
  + It is a great opportunity to remind them of the DRY principle and introduce them to an implementation that uses the basic constructor that is already implemented in the class.
* Try to show as many solutions as possible and review codes together.

**Introduction to List class - Back to our example**

Return to our example. Points for discussion:

* Ask the students to try to define the line, and explain why the line in our example (each person holds his ID and points at the person behind him) is equivalent to a linked list.

**Ex #2**

Instructions:

Ask the students to Implement a function that Adds the given value to the beginning of this list - implementing “addFirst” in List.java

Important Notes:

* Notice - the students have seen this implementation before.

This part should take 5-10 minutes and it’s necessary only if your students still feel very uncomfortable with lists.

* That’s a great time to make sure all the students understood those 3 important things:
  + Node.next
  + List.first
  + Dummy node
* Raise the question – how can we insert at end of list? (leading to simplicity of iterator).

**Code review - Iterating over a list**

Take 2-3 minutes to explain a few techniques/known variables used to iterate over a list.

That’s our time to help the students understand the “tricks” used in linked lists!

For example:

* Node current = first.next; //skipping the dummy node
* While (current != null) //checks if we’ve reached the end of the list
* Node next = current.next;
* Node prev = first;
* Feel free to add anything useful that can help students!

Discuss list iterator.

**Ex #3 - Recommended in small breakout rooms**

Instructions:

Ask the students to Implement the function “indexOf” in List.java.

The function Returns the index of the first occurrence of the given value, or -1 if not in list.

Notes:

This implementation was supposed to be included in HW8/9 but I couldn’t find it.  
We might change the function to a different one from List class

Extra material and information:

* Lists (what was given to the students)
* Lists solutions