Workshop 9 - Linked Lists

Red – instructions, Green – text from the presentation, Blue – pay attention for the staff

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. Specifically - 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 simplified example in the presentation (People in line):

  In the presentation: "Simplified Example":
  - 4 people are standing in line.
  - Each person has an ID.
  - Each person **ONLY** knows 2 things:
    - His own ID
    - Who is the next in line (stands behind him).
  - We want to know the ID of all the people standing in line.

In the presentation: “How can we discover all the ID’s?”:

- We will start from the 1st person in line
- Each Person will:
  - Tell me his own ID
  - Direct me to the next person in line
    - Notice: If person b is next in line behind person a,
      Person a can only point at person b,
      and **we will ask b** what’s his ID!
- The last person in line is the one that has no one to point at. And after getting his ID, we have discovered all ID’s.

- Make sure to clarify the difference between the ID (equivalent to Value) and Person (Equivalent to Node) – a person has an ID.
- 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.

In the presentation: “Relating to Nodes”:

- 4 people are standing in line **- 4 Nodes (Person = Node)**
- Each person has his own ID **- Value**
- Each person knows 2 things:
  - His own ID – **Value**
  - Who is next in line (standing behind him) – **Next ("הבא בתור")**

In the presentation: “Node Class”:

- Defines a new **type** called “Node”
- Each object of type Node has 2 properties/field:
  - **Value -** the value it holds (In our example - the ID)
  - **Next -** the next node in the list (In our example – the person behind in line)
- We refer to field “Next” as a **pointer.** It points to the next Node in the list

Now, Go through "Node.java" file. Explain:

- The fields.
- The 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 (the line).

In the presentation: "Back to our example":

- The 1st person in line is our "Starting point":
    - Each person will lead us to the next one in line
- Each Person had:
    - His own ID
    - Pointer to the next person in line
- The last person in line is the one that has no one to point at.

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 next person in line) is equivalent to a linked list.

In the presentation: “Relating to Linked Lists”:

- If I ask you to define the line, how would you do that?
  - A bunch of people standing one after the other
  - Each person “leads” to the person after him - points at him
- The line in our example is exactly the **Linked List!**
  - “A bunch” of nodes connected to one another by pointers
  - A person in line = a node in the list

Now, Go through “List.java” file. Explain:

- The fields.
- The first constructor implementation
  - **Very important!** - Rememmber to explain the dummy concept.
    If we want to make a parallel example to our line example, one can think of the dummy as a person that his entire job is to point to the first person in the line (this dummy person **is not in the line**).

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:

- Remind the students that in the List object they have access to the “first” Node, and that this node is the dummy, and it has a next pointer.
- 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**.

Techniques / tricks

Without yet mentioning iterators, 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; // saving temporarily the next nodeof current

Feel free to add anything useful that can help students!

Ex #3

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.

Iterating over a list

Now, raise the question: "how can we insert a value at end of list?"

(leading to simplicity of iterator).

Finally - mention and discuss list iterator:

1. Review "ListIterator.java"
    - A confusing point worth talking about:

      "hasNext()" is returning whether the **current** node is not null

      ==

      "Is the node that the iterator is currently at is not null"

      ==

      "If we use the "next()" function, would we get a value which is not null?"
2. Review the "listIterator(int index)" function in "List.java".

Extra time

Ask the students to implement the rest of the function in “List.java”:

```
/** Returns true if this list contains the specified value. */
public boolean contains(int val)

/** Returns the index of the last occurrence of the specified value. */
public int lastIndexOf(int val)

/** Returns the value at the specified position in this list. */
public int get(int index)

/** Sets the value at the given index to the given value */
public void set(int index, int val)

/**
 * Disposes this list. Goes through the list and
 * sets the next field of each node to null.
 */
public void dispose()

/** Returns an array containing all of the elements in this list. */
public int[] toArray()
```