EEM 480 ALGORITHM AND COMPLEXITY #HW 2

Mehmet Çağrı Aksoy – 12105037560

mcagriaksoy@ya.ru

Introduction

This homework is given to understand the linked list structure, implement the interface and develop object-oriented programming skills.

I have used NetBeans 8.2 IDE for Ubuntu 18.04.

Unlike the first homework, I have used Singly Linked List Structure and Java Interface. An interface is a reference type in Java. It is like a class. It is a collection of abstract methods. Linked List are linear data structures where the and every element is a separate object with a data part and address part. Using "linked list structure" in this assignment will benefit us as time complexity. Insertion and deletion operations only O(1) so, adding and deletion operations of inventory will be very fast.

To return to my program; it has 4 different .java file. The main one, Interface, inventory and class for Nodes. Inventory class implements the interface. Also, Node's elements are used Inventory class. All function purposes are defined in the inventory class. In the inventory class works based on 3 different functions; add, deletion and print. It repeats for each element namely, the resistor, capacitor, inductor, and transistor.

Functions (Explained)

```
public void addResistor(String val, Integer cnt );
```

It takes, 'val' string and integer 'cnt'. Its values come from the main class. It takes 2 parameters, but node requires 4 parameters. So, I filled other parameters with some default values, strings etc.

For homework requirements, I needed to put a resistor into the inventory, so that, I must create Node; due to given *Device class, in my code block, I have changed the name of device class to Node object.

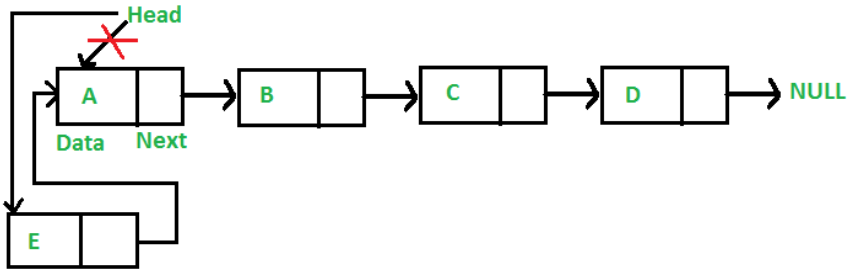

Figure 1 - Add a node at the front

I have added resistor's nodes at the front of my list. Because, It will help later, if I want to print inventory with order "Resistor → Capacitor → Inductor → Transistor". And O(n) complexity.

IF IT FOUND SAME RESISTOR THAT ADDED BEFORE → Addresistor() searches whole list, if it finds element which is the same as the element we want to put, it do not add node after previous one, it takes count data from and sums our "cnt".

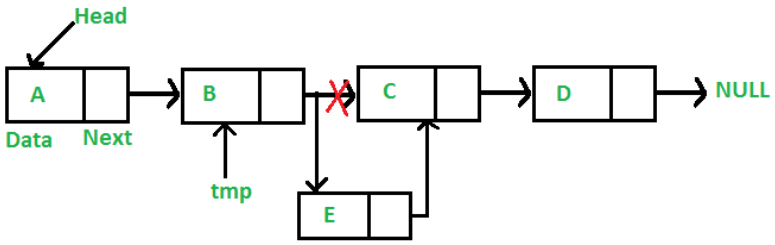

Figure 2 - Add a node after a given node

Capacitors and inductors have also been added to the middle of the list. To do this perfectly;

For inductor; if there is a transistor in the list, add it in front of the transistor. If the transistor does not exist on the list, then add it at the end like Figure-3.

For capacitor; If you have resistance on the list, add it right behind it. If there is no resistance, you can add to the first like Figure-1.

So, I added transistors at the end of my list. Because I don't want to order my list later, and due to my transistor's node location, I can easily add my inductors.

In deletion operations, again program searches all list, if it finds the element, subtract the number of elements we want to remove. If element do not find or, placed element's piece is less than we want to number of pieces returns -1 value.

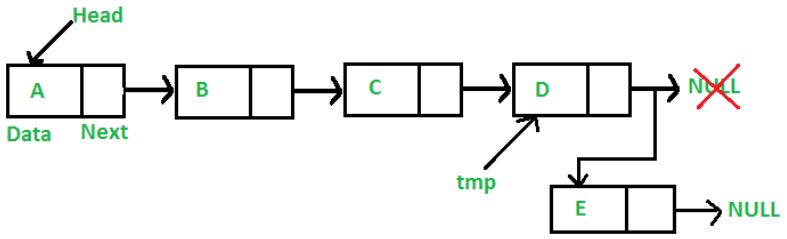

Figure 3 - Add a node at the end

Due to homework's requirement, I need to print inventory with O(n) complexity and desired order. Order is not problem because I have ordered my elements before. So, In this function I only need to search all list from beginning to end after that I can print my list.

public void printInventory(String val, Integer cnt); : The devices in the Inventory has to be printed in an ordered fashion. First Resistors then Capacitors then Inductors and finally Transistors in **O(n) complexity**.

One loop is required to print all Nodes; O(n) complexity represents approximately one loop which is "for" or "while". More information can be found on the picture;

Print inventory;

```
@Override
    public void printInventory() {
        current = head;
        while(current != null){   //It searches all of list and prints datas from llist.
            System.out.println(current.Name+" "+current.Type+" "+current.Value+"
"+current.Count);
            current=current.next;
        }
    }
```

In main class;

Main class is using DB, it uses "Inventory" classes, functions. I have defined;

Inventory myInventory = new Inventory(); after I can use Inventory's methods,classes etc myInventory.addResistor("2K", 20); like In this code block , 2K and 20 values are sent to addResistor method.

OOP Concepts:

Encapsulation , I am using public modifiers to define parameters.

Polymorphism, My object Node, is behaving different if it has parameters or not, so polymorphism feature is using in my program.

Inheritence concept is not exist in my program, because I do not extend any reference to use with multiple object that defined on a class.