Lab 3: Circular Linked-List

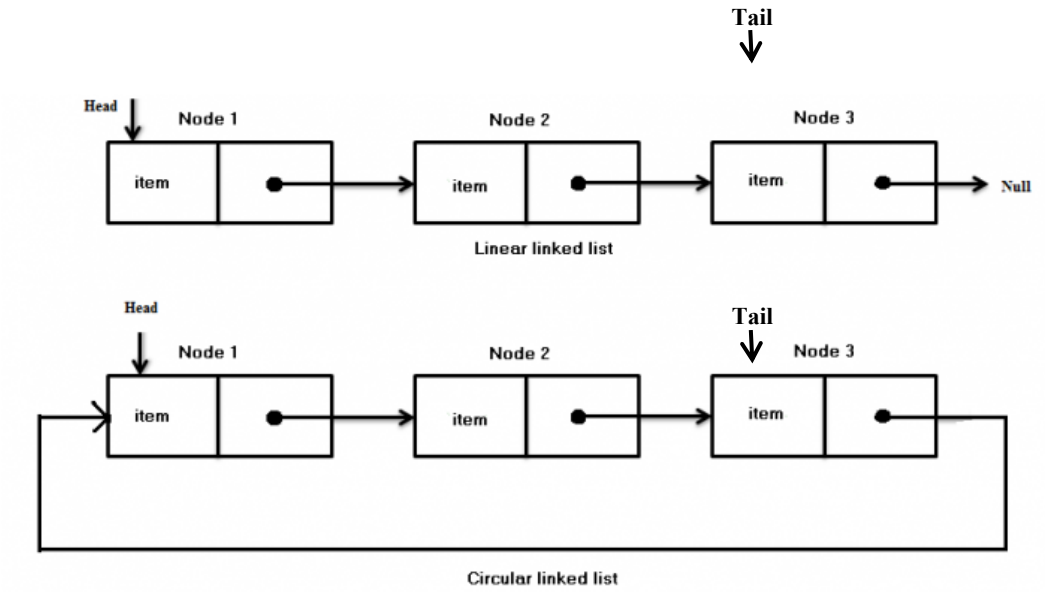
Objective(s)

1- Create Circular Linked-List in Java.
2- Deal with Circular Linked-List in case of: insertion, Deletion, searching .

Tool(s)/Software

Java programming language with NetBeans IDE.

Description:



Singly Circular Linked List: It is just like the singly/linear linked list where tail node holds the address of head node so traversal can be done circular in only one direction.

A. How to Create Circular Linked-List in Java:

There are **3-steps** approach to create Circular Linked-List in Java

Step 1: Declare class for the **Node** – forming the structure of the node

```
private static class Node<E>{
    private E element;
    private Node<E> next;
    public Node(E e, Node<E> n){
        this.element = e;
        this.next = n;
    }
    public E getElement(){
        return this.element;
    }
    public Node<E> getNext(){
        return this.next;
    }
    public void setNext(Node<E> n){
        this.next=n;
    }
}
```

Step 2: Declare the **CircularLinkedList** class that includes the Node class.

```
public class CircularLinkedList<E> {

    private class Node {...26 lines }
    Node head=null;
    Node tail=null;
    int size = 0;
    public CircularLinkedList() {}
    //Access Methods
    public int size() {...3 lines }
    public boolean isEmpty() {...3 lines }
    public int first() {...3 lines }
    public int last() {...3 lines }
    public void display() {...14 lines }

    //Update Methods
    public void addFirst(int value) {...15 lines }
    public void addLast(int value) {...14 lines }
    public void addAtPos(int pos, int value) {...16 l
```

Display method in CircularLinkedList class:

```
public void display() {
    if (isEmpty())
        System.out.println("Empty List..");
        Node current = head;
        int i = 1;
        do {
            System.out.println("Node " + (i++) + ": " + current.getData());
            current = current.getNext();
        } while (current != head);
    }
```

Step 3: Define the object of **CircularLinkedList** class:

```
CircularLinkedList myList=new CircularLinkedList ();
myList.addFirst(10);
myList.addFirst(20);
myList.addFirst(30);
myList.addFirst(40);
```

B. Circular Linked-List Operations:

1. Traversing Circular Linked-List
2. Searching in Circular Linked-List
3. Insertion in Circular Linked-List
4. Deletion from Circular Linked-List

Note: *Please refer to lecture slides for the algorithms.*

Tasks/Assignments(s)

1. Create CircularLinkedList class. Apply all the following operations:
    - **Adition:** addFirst, addLast, addAtPoistion
    - **Deletion:** removeFirst, removeNode.
    - **FindNode**: to find a node with specific given value.

2. Add a method rotate() to CircularLinkedList class that shift the list one node, below is the output before and after calling the method.

```
run:
Display List:
        Node 1 :40      Node 2 :30      Node 3 :20      Node 4 :10
The list is shifted
Display List:
        Node 1 :30      Node 2 :20      Node 3 :10      Node 4 :40
First element: 30
Last element: 40
```

Deliverables(s)

You are required to implement and deliver a Java program as described in the previous section.