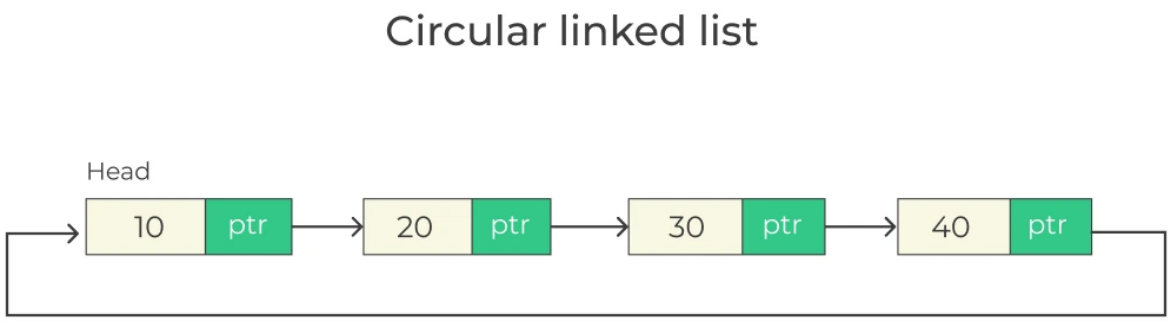
Circular Linked List

A circular linked list is similar to the single linked list except that the last node points to the first node in the list. This means that circular linked list is a sequence of elements in which every element has link to its next element in the sequence and the last element has a link to the first element in the sequence.

Applications of Circular Linked List

A Circular Linked List can be used for the following –

- Circular lists are used in applications where the entire list is accessed one-by-one in a loop.
- It is also used by the Operating system to share time for different users, generally uses a Round-Robin time-sharing mechanism.
- Multiplayer games use a circular list to swap between players in a loop.
- Implementation of Advanced data structures like Fibonacci Heap
- The browser cache which allows you to hit the BACK button
- Undo functionality in Photoshop or Word
- Circular linked lists are used in Round Robin Scheduling
- Circular linked list used Most recent list (MRU LIST)



- Circular linked lists on the other hand have eliminated the concept of null.

- The head points to the very first element and the first node points to the second and the chain continues.
- The last node instead of pointing towards null, points to the first node making the linked list complete a imaginary circle.

Insertion in Circular Linked List

We can perform three different types of insertion in circular linked list, these different types of insertion are:-

- Insertion at beginning .
- Insertion at specific position .
- Insertion at end .

Insertion in the Beginning of a Circular Linked List

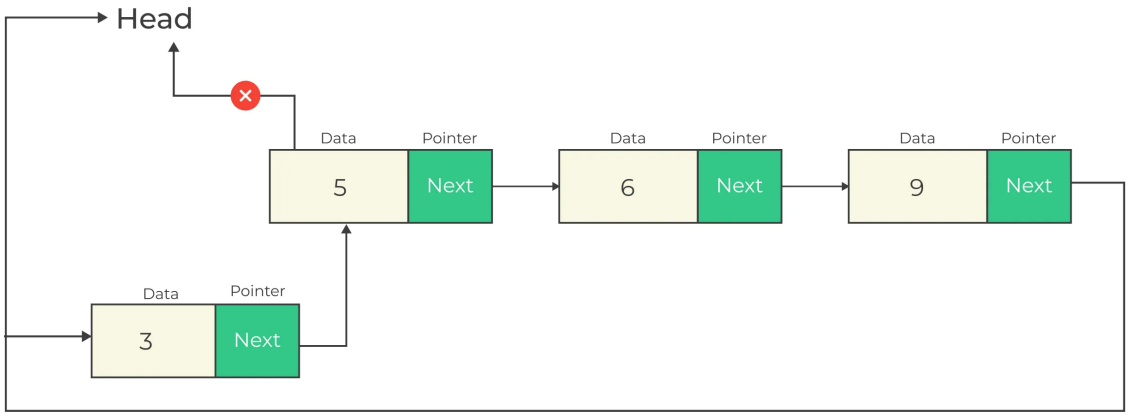
- The circular linked is empty –
  - To insert an element in an empty circular list is quite easy.
  - Both the head and the tail points to the very first element in a circular linked list.
- The circular linked list is not empty –
  - In a circular linked list the first element serves the purpose of a head.
    It stores the information of the next node of the circular linked.
  - To insert an element in the beginning of the linked list. The new node should point towards the previous Head making it the new head of the circular linked list.
  - The tail is the last node that points back to the first note making the linked list circular because the traversal can be done from start till end.
  - To add a node in the beginning means to change the node tail is referring to newly added node. By
    doing this the new node will be inserted in the beginning of the
    circular linked list.

Algorithm

- addFirst(element)

- IF Head==Null
  - Head -> newNode
  - Tail -> newNode
  - newNode->next = Head
  - EXIT
- ELSE
  - Node Temp = head
  - newNode->next = Temp
  - head = newNode
  - tail->next = Head
  - EXIT

Insertion At Beginning in Circular Linked List in Java



Insertion at the End of a Circular Linked List

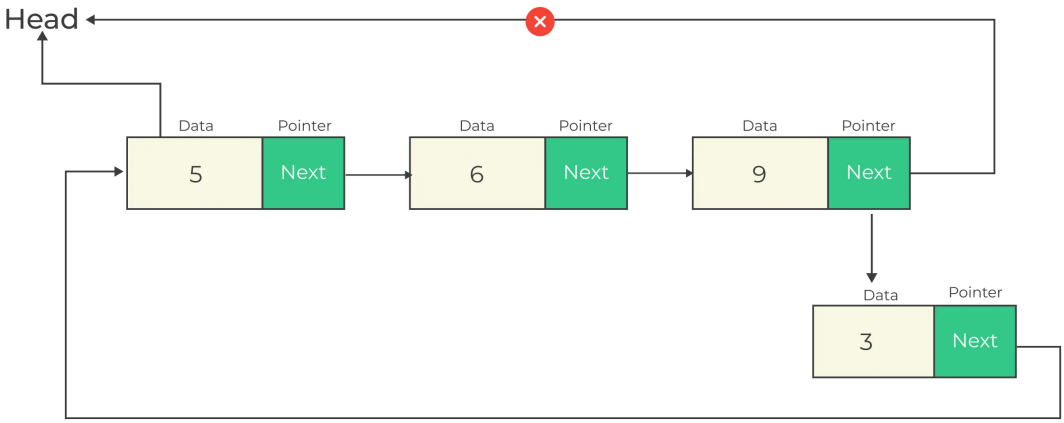
- The circular linked list is empty –
  - If the circular linked is empty which means the head is null then the new node will be added as head.
  - Both the head and the tail will point towards the head because there is only one element.

- The circular linked list is not empty –
  - A head is the first node of the circular linked list which can be considered as the starting of the list and points towards the next element.
  - The tail is the last element of the circular linked list and it point towards the head making the list appear circular.
  - If the list is not empty which means if the list has elements the last node will be the tail.
  - To insert an element in the end of circular linked list, the present tail should be replaced by the new node.
  - By changing the node referred by the current tail to the newly added node, it has become the new tail.
  - Now, make the new tail point towards the head of the circular linked list and the element is added in the end of the circular linked list.

Algorithm

- insertEnd(element)
- IF Head==Null
  - Head -> newNode
  - Tail -> newNode
  - newNode->next = Head
  - EXIT
- ELSE
  - tail->next = newNode
  - tail = newNode
  - tail->next = Head
  - EXIT

Insertion At End in Circular Linked List in Java



Insertion at the n^{th} node of a Circular Linked List

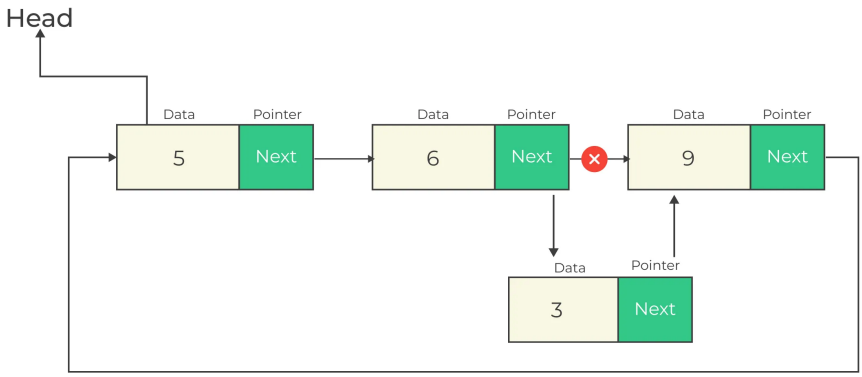
- While inserting a new node in the anywhere between the head and the rails we need to consider the three nodes.
- The (n-1)th node must be pointing towards the current nth node, the current nth node must be pointing towards the current (n+1)th node.
- First of all the node (n-1)th address of the next node must be copied in the
  address part of the new node. Since it is referring to the node we will consider as n+1 after the insertion is successful.
- Change the address of (n-1)th node to the address of the new node.
- In this way, insertion at any position can be done in a circular linked list.

Algorithm

- addLast(element)
- IF Head==Null
  - Head -> newNode
  - Tail -> newNode
  - newNode->next = Head
  - EXIT

- ELSE
  - tail->next = newNode
  - tail = newNode
  - tail->next = Head
  - EXIT

Insertion in Between The Nodes in Circular Linked List in java




```
package linklist;

public class CircularLinkedList {
  public class Node {
    int element;
    Node next;

    public Node(int element) {
      this.element = element;
    }
  }

  public Node head = null;
  public Node tail = null;
  int size = 0;

  public void insertBegin(int element) {
    Node newEle = new Node(element);
    if (head == null) {
      head = newEle;
      tail = newEle;
      newEle.next = head;
```

```
    } else {
      tail.next = newEle;
      tail = newEle;
      tail.next = head;
    }
  }

  public void insertEnd(int element) {
    Node newEle = new Node(element);
    if (head == null) {
      head = newEle;
      tail = newEle;
      newEle.next = head;
    } else {
      tail.next = newEle;
      newEle.next = head;
      tail = newEle;
    }
    size++;
  }

  public void insertAfter(int n, int data) {
    int size = calcSize(head);

    // Can only insert after 1st position
    // Can't insert if position to insert is greater than size of Linked List
    if (n < 1 || n > size) {
      System.out.println("Can't insert\n");

    }
    if (n == 1) {
      insertBegin(data);
      return;
    }

    Node newNode = new Node(data);
    // required to traverse
    Node temp = head;

    // traverse to the nth node
    while (--n > 1)
      temp = temp.next;

    newNode.next = temp.next;
    temp.next = newNode;

  }

  public int calcSize(Node node) {
    int size = 0;
    while (node != tail) {
      node = node.next;
      size++;
    }
    return size + 1;
  }

  public void print() { // print function
    Node current = head;
```

```
    if (head == null) {
      System.out.println("List is empty");
    } else {
      System.out.println("Nodes of the circular linked list: ");
      do {
        System.out.print(" " + current.element);
        current = current.next;
      } while (current != head);
      System.out.println();
    }
  }

  public static void main(String[] args) {
    CircularLinkedList Obj = new CircularLinkedList();
    Obj.insertBegin(11);
    Obj.insertBegin(22);
    Obj.insertBegin(33);
    Obj.insertBegin(44);
    Obj.insertBegin(55);

    Obj.print();

    Obj.insertEnd(110);
    Obj.insertEnd(220);
    Obj.insertEnd(330);
    Obj.insertEnd(440);

    Obj.print();
    Obj.insertAfter(2,77);
    Obj.insertAfter(3,88);
    Obj.print();
  }
}
```

Deletion in Circular Linked List

We can perform three different types of **Deletion** in circular linked list, these different types of **Deletion** are:-

- **Deletion** at beginning .
- **Deletion** at specific position .
- **Deletion** at end .

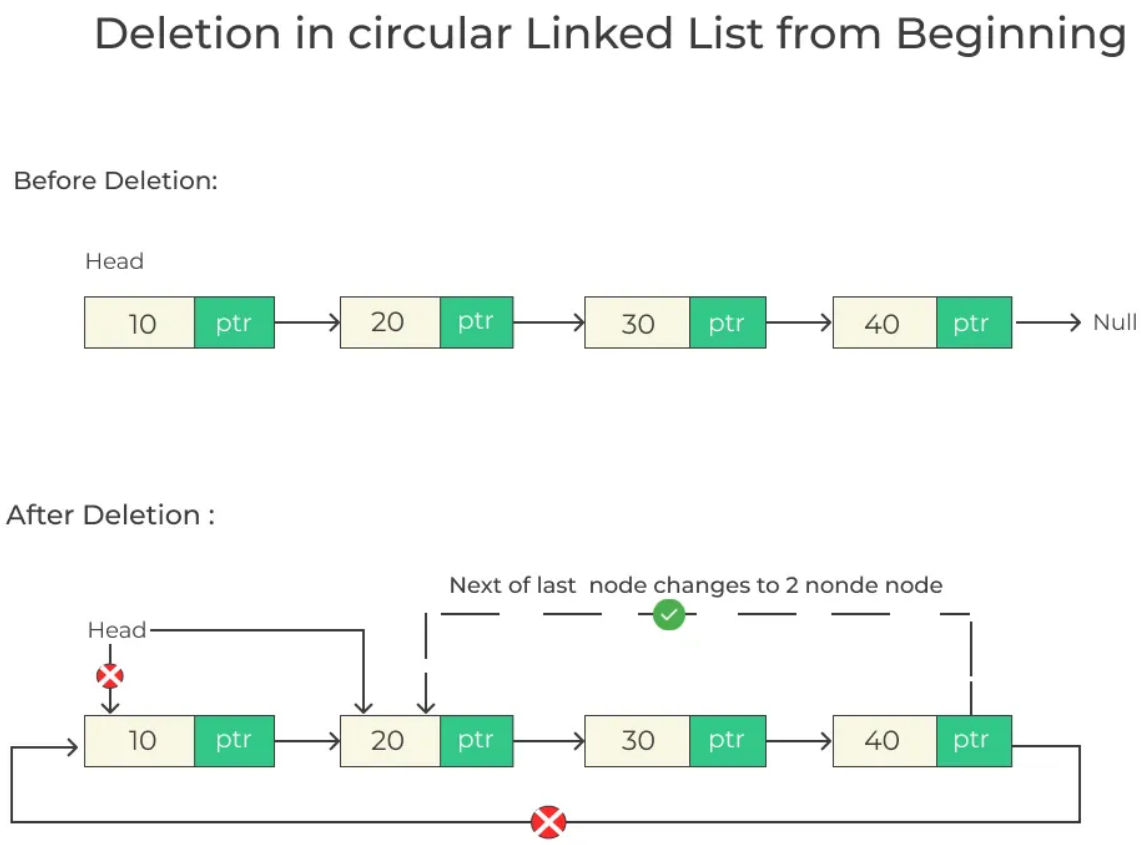
Deletion from the Beginning of a Circular Linked List

- In a circular linked list a reference variable head points to the first node. The first node stores the information of the next node of the circular linked.

- The tail is the last node that points
  back to the first node making the linked list circular because the
  traversal can be done from start till end.
- To delete an element
  in the beginning of the linked list. The first node should be deleted
  and the reference should point towards the second node making it the
  first element of the circular linked list.
- To delete a node in
  the beginning means to change the node tail is referring to second node. By
  doing this the second node will become the first node of the
  circular linked list and the node from the beginning will be deleted.

Algorithm

- deleteFirst()
- IF head == null
  - return
- ELSE IF head != tail
  - head = head -> next
  - tail -> next = head
- ELSE
  - head = tail = null



Deletion at the n^{th} node of a Circular Linked List

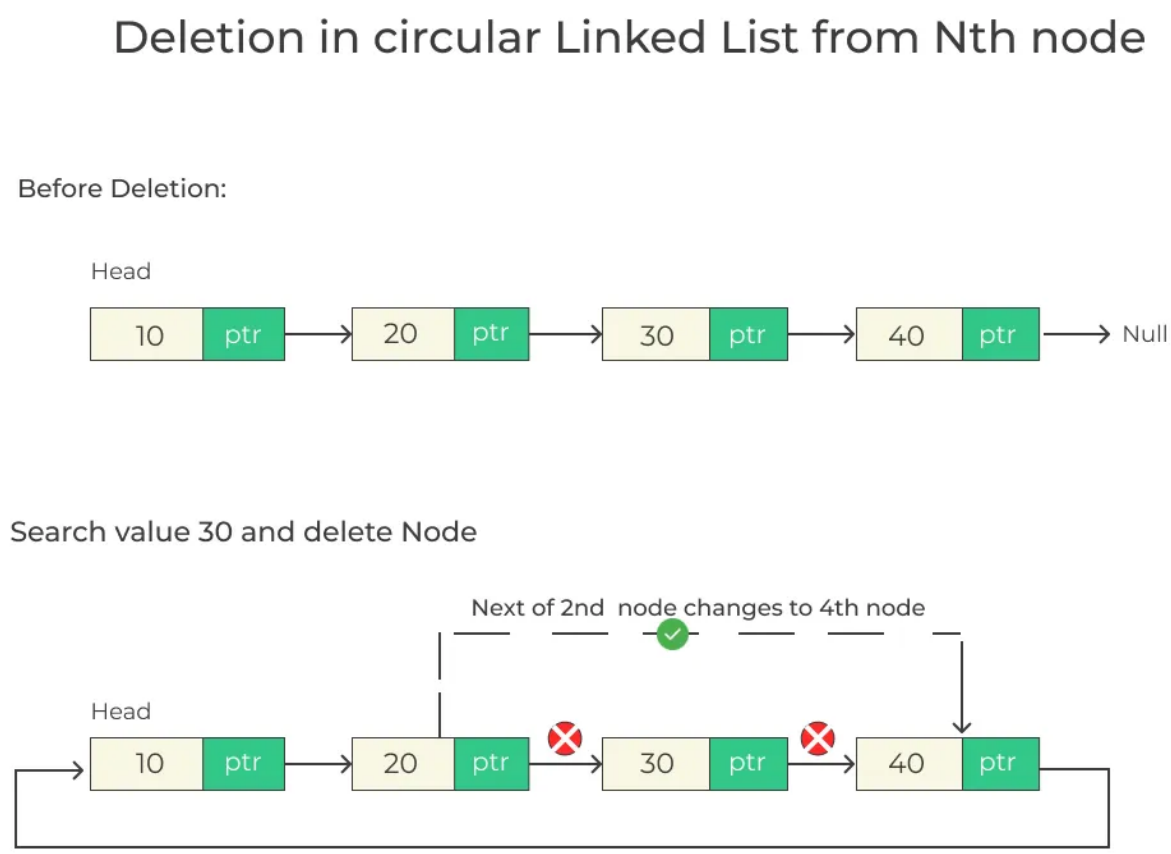
- If the list is empty we cannot delete element from it hence return as deletion is not possible.
- If there is data present in the nodes of the list, then
- Traverse till the specific position from where you want to delete an element.
- Then make next pointer of previous list as next of next node.
- Free the node that is present at specific position.

Algorithm

- deleteNth()
- IF head == null
  - return
- ELSE
  - WHILE (–n>0)

- previous = temp;
- temp = temp.next;

- previous.next = temp.next;



Deletion from the End of a Circular Linked List

- A head is the reference of the first node of the circular linked list which can be considered as the starting of thelist and points towards the next element.
- The tail is the last element of the circular linked list and it point towards the first element making the list appear circular.
- If the list is not empty which means if the list has elements the last node will be the tail.
- To delete an element in the end of circular linked list, the present tail should be removed.

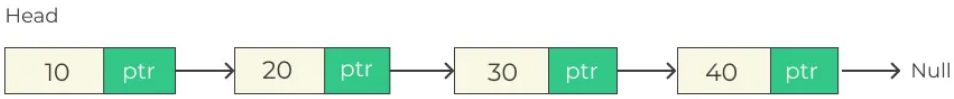
- By changing the node referred by the second last node to the first element , it has become the new tail.
- In this way the element is deleted from the end of the circular linked list.

Algorithm

- deleteLast()
- IF head == null
  - return
- ELSE IF head != tail
  - Node current = head
  - WHILE current->next != tail
    - current = current-> next
    - tail = current;
    - tail -> next = head
- ELSE
  - head = tail = null

Deletion in circular Linked List from last

Before Deletion:



After Deletion :