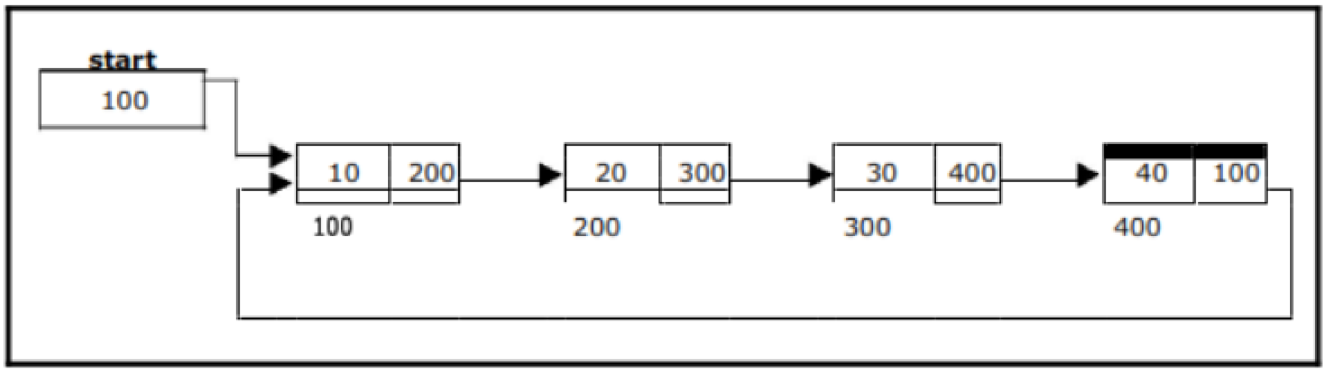
Contents

1. **Circular Linked List** 2
2. **Advantages of Circular Linked Lists:** 2
4. **Creating a Node of Circular LinkedList in JAVA** 3
5. **Method to Insert a New Node (n) at the Beginning of Circular LinkedList** 4
6. **Method to Insert a New Node (n) at Last of Circular LinkedList** 5
7. **Method to Insert a New Node (n) Before A Particular Value in a Circular LinkedList** 6
8. **Method to Delete the First Node in Circular LinkedList** 7
9. **Method to Delete the Last Node in Circular LinkedList** 8
10. **Display Method for Circular LinkedList** 9
11. **Java Program to InstertAtFirst, InstertAtLast, DeleteFirst & DeleteLast in Circular LinkedList** 10

1. Circular Linked List

- Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.
- Circular linked lists are frequently used instead of ordinary linked list because many operations are much easier to implement. In circular linked list no null pointers are used, hence all pointers contain valid address.



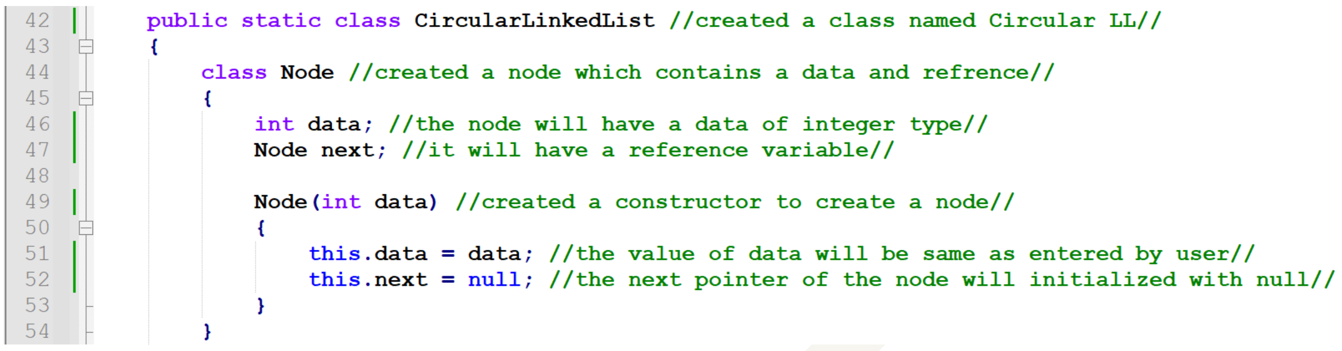
2. Advantages of Circular Linked Lists:

1. Any node can be a starting point. We can traverse the whole list by starting from any point. We just need to stop when the first visited node is visited again.

2. Useful for implementation of queue. Unlike this implementation, we don't need to maintain two pointers for front and rear if we use circular linked list. We can maintain a pointer to the last inserted node and front can always be obtained as next of last.

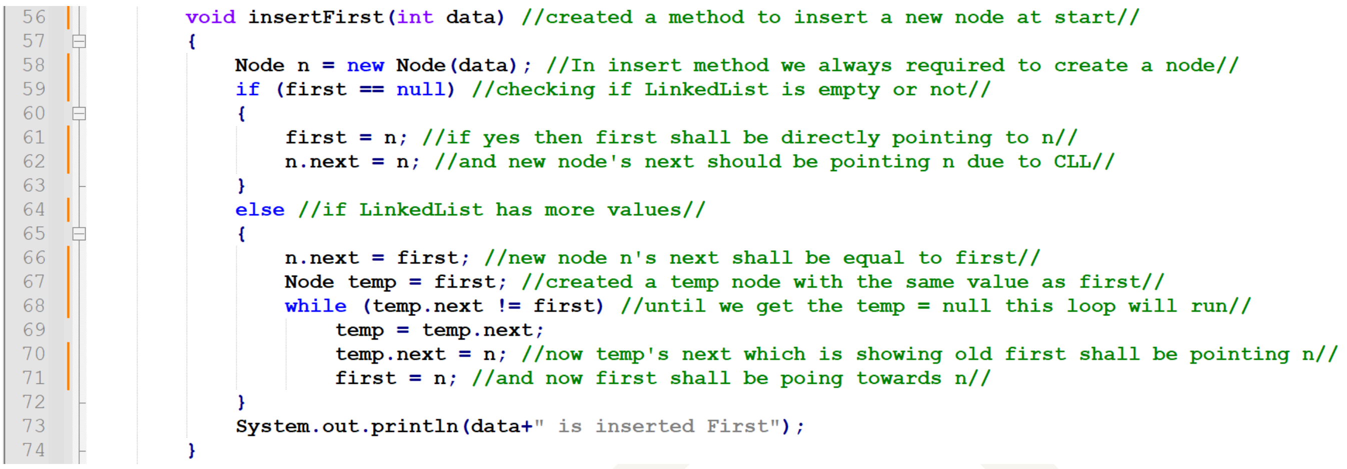
3. Circular lists are useful in applications to repeatedly go around the list. For example, when multiple applications are running on a PC, it is common for the operating system to put the running applications on a list and then to cycle through them, giving each of them a slice of time to execute, and then making them wait while the CPU is given to another application. It is convenient for the operating

4. Creating a Node of Circular LinkedList in JAVA

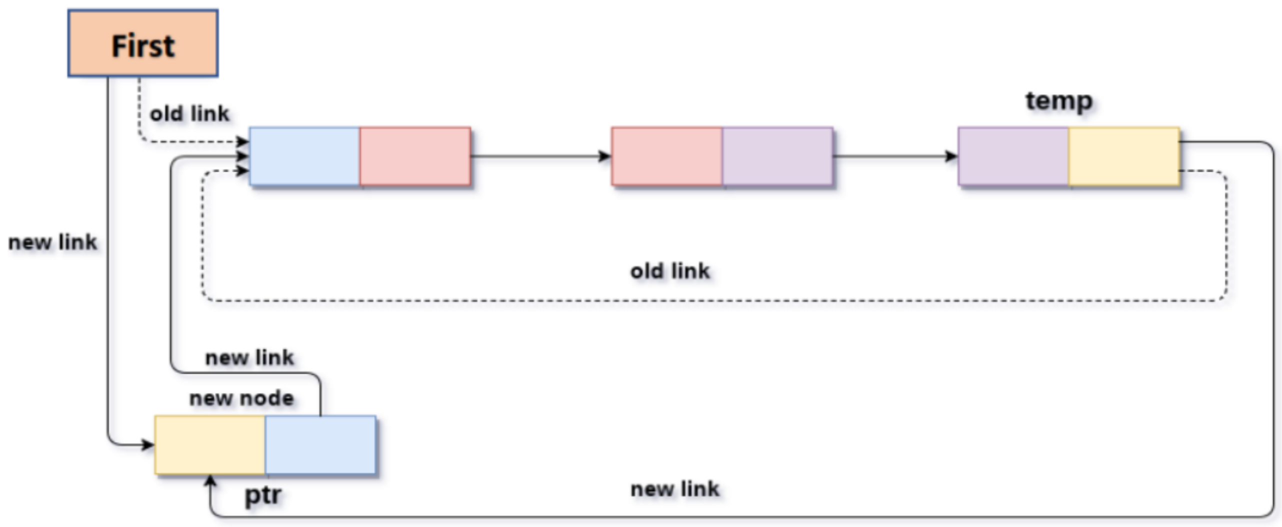


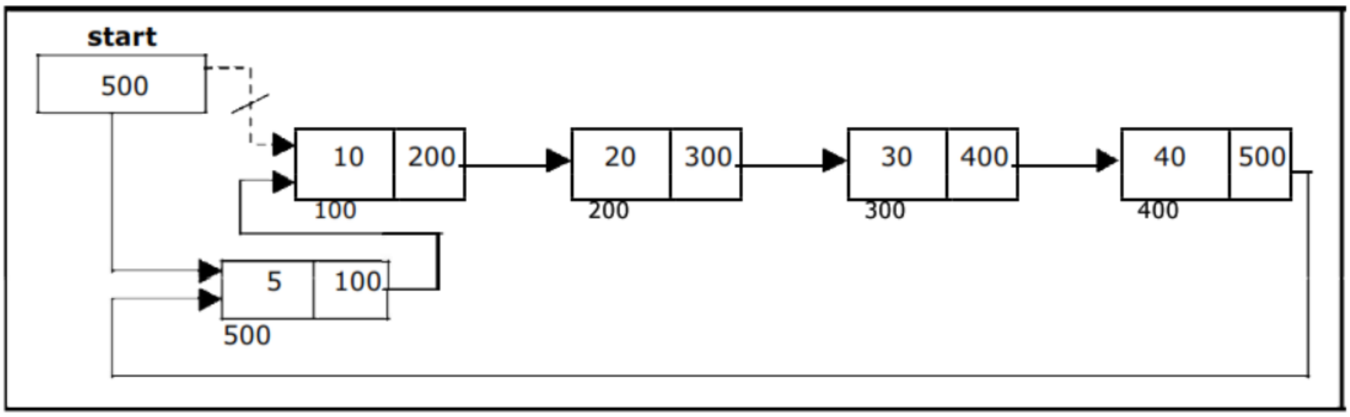
```
    public static class CircularLinkedList //created a class named Circular LL//
    {
        class Node //created a node which contains a data and refrence//
        {
            int data; //the node will have a data of integer type//
            Node next; //it will have a reference variable//

            Node(int data) //created a constructor to create a node//
            {
                this.data = data; //the value of data will be same as entered by user//
                this.next = null; //the next pointer of the node will initialized with null//
            }
        }
```

5. Method to Insert a New Node (n) at the Beginning of Circular LinkedList

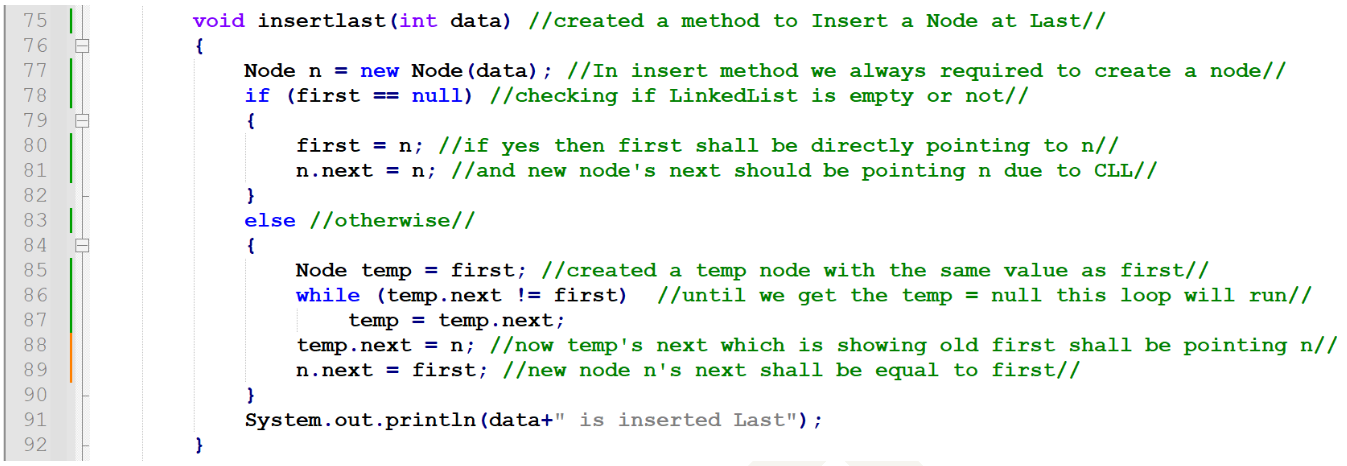


```
void insertFirst(int data) //created a method to insert a new node at start//
{
    Node n = new Node(data); //In insert method we always required to create a node//
    if (first == null) //checking if LinkedList is empty or not//
    {
        first = n; //if yes then first shall be directly pointing to n//
        n.next = n; //and new node's next should be pointing n due to CLL//
    }
    else //if LinkedList has more values//
    {
        n.next = first; //new node n's next shall be equal to first//
        Node temp = first; //created a temp node with the same value as first//
        while (temp.next != first) //until we get the temp = null this loop will run//
            temp = temp.next;
            temp.next = n; //now temp's next which is showing old first shall be pointing n//
            first = n; //and now first shall be poing towards n//
    }
    System.out.println(data+" is inserted First");
}
```

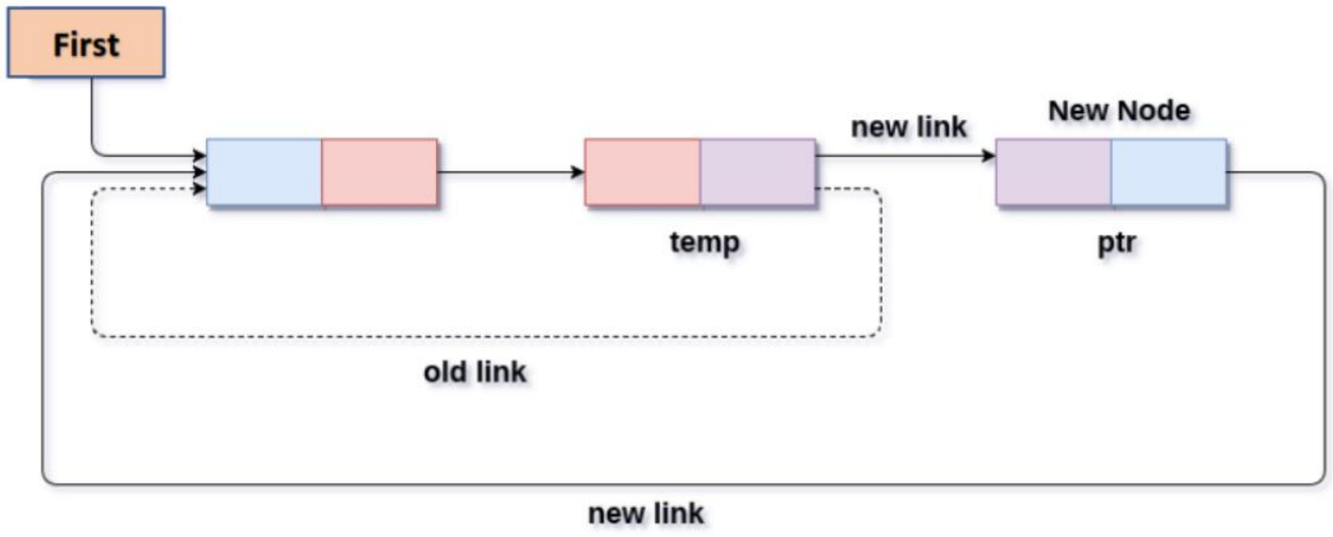


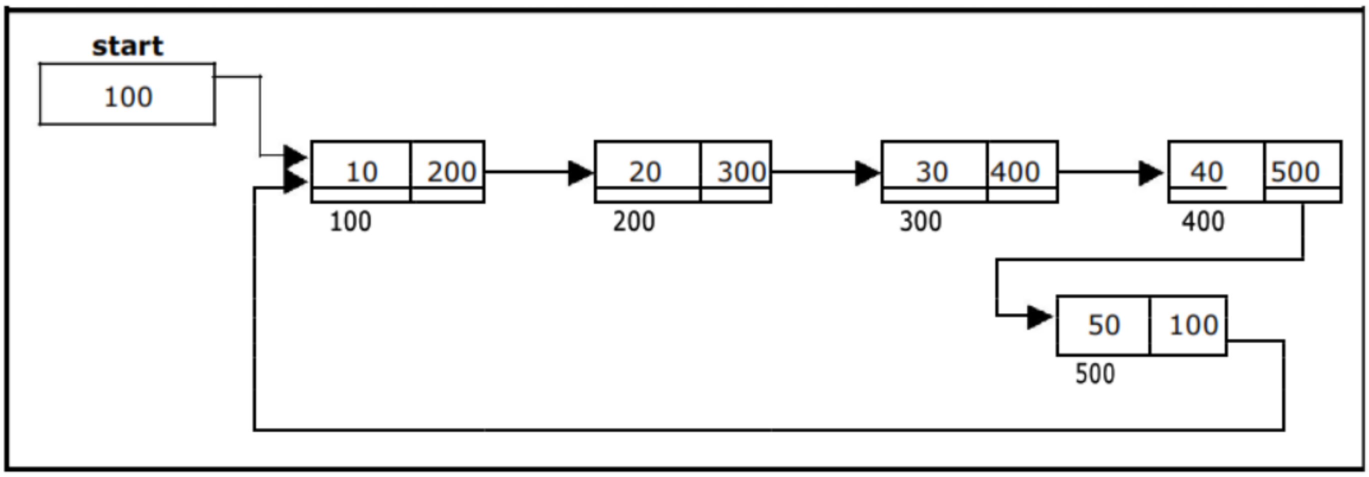


6. Method to Insert a New Node (n) at Last of Circular LinkedList

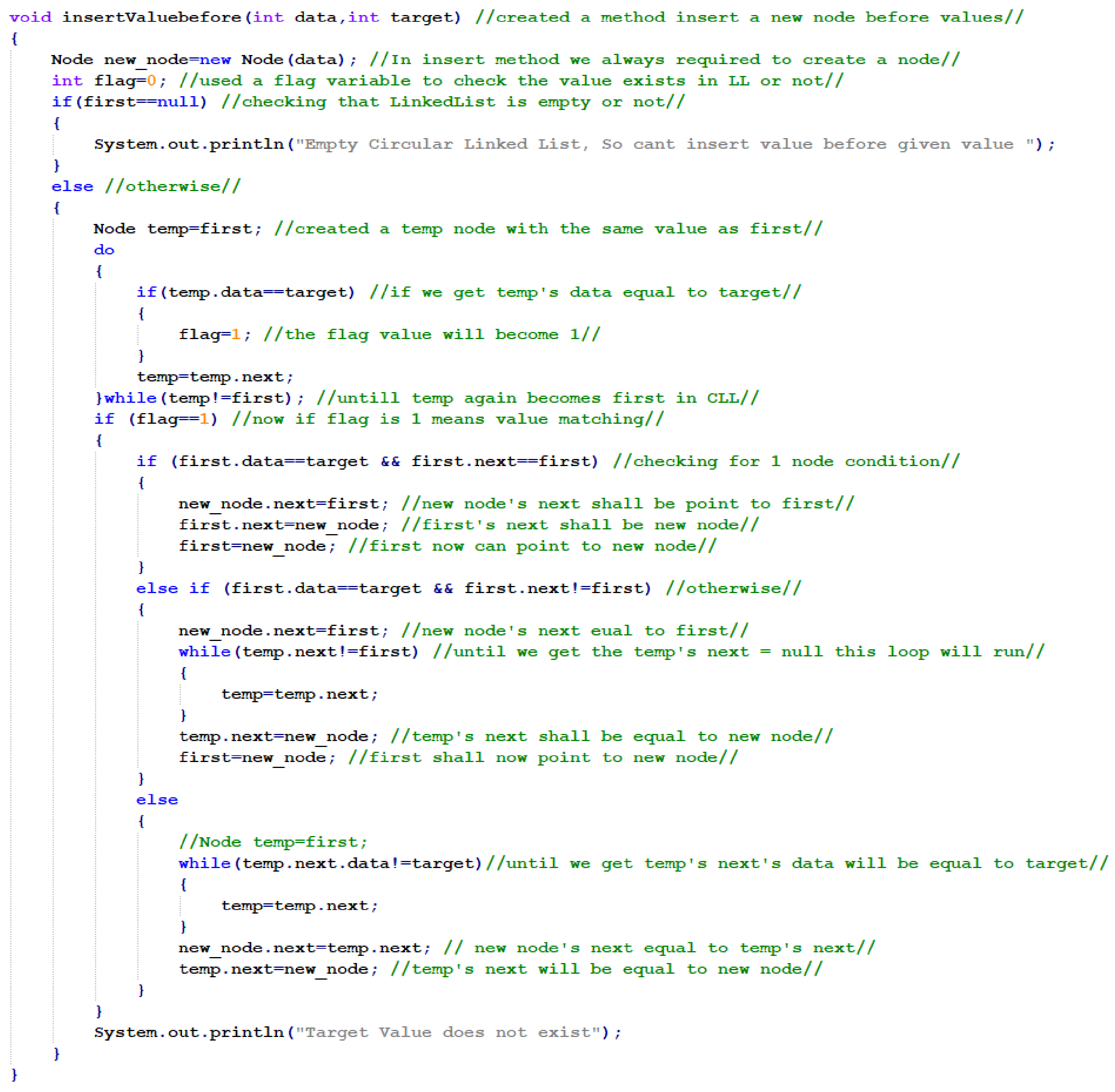


```
void insertlast(int data) //created a method to Insert a Node at Last//
{
    Node n = new Node(data); //In insert method we always required to create a node//
    if (first == null) //checking if LinkedList is empty or not//
    {
        first = n; //if yes then first shall be directly pointing to n//
        n.next = n; //and new node's next should be pointing n due to CLL//
    }
    else //otherwise//
    {
        Node temp = first; //created a temp node with the same value as first//
        while (temp.next != first)  //until we get the temp = null this loop will run//
            temp = temp.next;
        temp.next = n; //now temp's next which is showing old first shall be pointing n//
        n.next = first; //new node n's next shall be equal to first//
    }
    System.out.println(data+" is inserted Last");
}
```



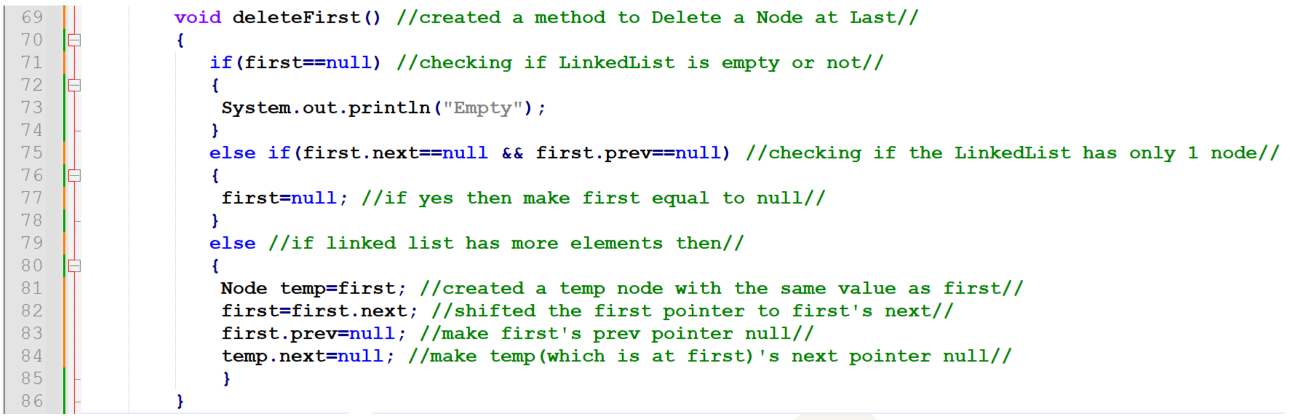


7. Method to Insert a New Node (n) Before A Particular Value in a Circular LinkedList

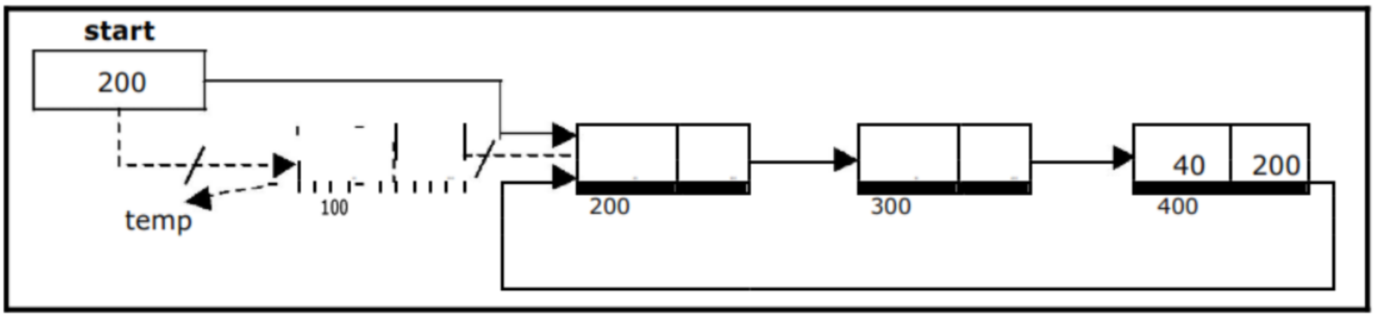


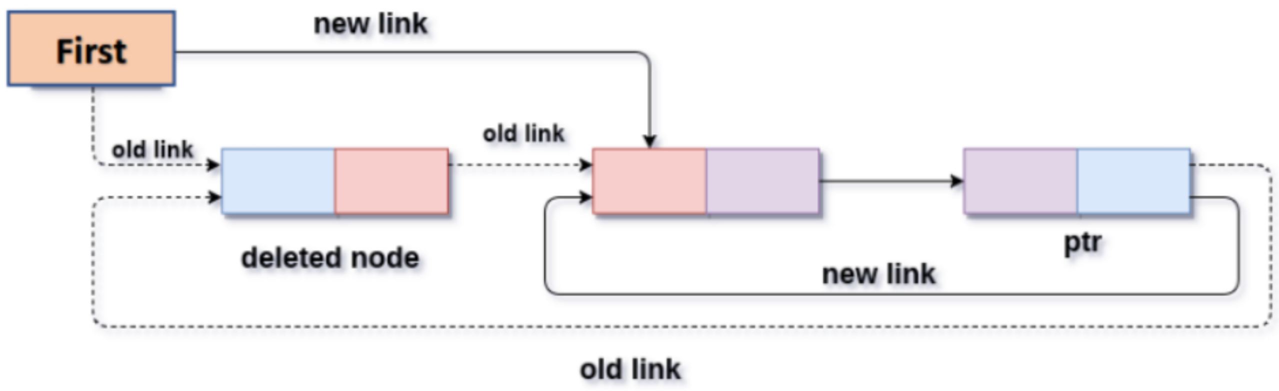
```
void insertValuebefore(int data,int target) //created a method insert a new node before values//
{
    Node new_node=new Node(data); //In insert method we always required to create a node//
    int flag=0; //used a flag variable to check the value exists in LL or not//
    if(first==null) //checking that LinkedList is empty or not//
    {
        System.out.println("Empty Circular Linked List, So cant insert value before given value ");
    }
    else //otherwise//
    {
        Node temp=first; //created a temp node with the same value as first//
        do
        {
            if(temp.data==target) //if we get temp's data equal to target//
            {
                flag=1; //the flag value will become 1//
            }
            temp=temp.next;
        }while(temp!=first); //untill temp again becomes first in CLL//
        if (flag==1) //now if flag is 1 means value matching//
        {
            if (first.data==target && first.next==first) //checking for 1 node condition//
            {
                new_node.next=first; //new node's next shall be point to first//
                first.next=new_node; //first's next shall be new node//
                first=new_node; //first now can point to new node//
            }
            else if (first.data==target && first.next!=first) //otherwise//
            {
                new_node.next=first; //new node's next eual to first//
                while(temp.next!=first) //until we get the temp's next = null this loop will run//
                {
                    temp=temp.next;
                }
                temp.next=new_node; //temp's next shall be equal to new node//
                first=new_node; //first shall now point to new node//
            }
            else
            {
                //Node temp=first;
                while(temp.next.data!=target)//until we get temp's next's data will be equal to target//
                {
                    temp=temp.next;
                }
                new_node.next=temp.next; // new node's next equal to temp's next//
                temp.next=new_node; //temp's next will be equal to new node//
            }
        }
        System.out.println("Target Value does not exist");
    }
}
```

8. Method to Delete the First Node in Circular LinkedList

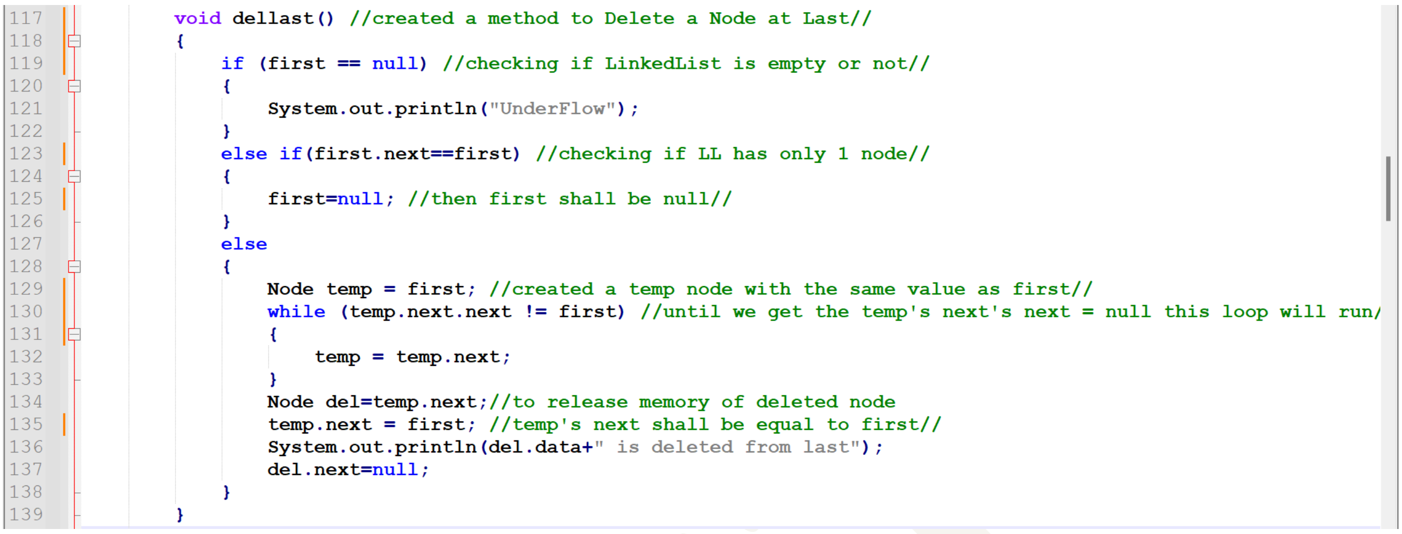


```
void deleteFirst() //created a method to Delete a Node at Last//
{
   if(first==null) //checking if LinkedList is empty or not//
   {
    System.out.println("Empty");
   }
   else if(first.next==null && first.prev==null) //checking if the LinkedList has only 1 node//
   {
    first=null; //if yes then make first equal to null//
   }
   else //if linked list has more elements then//
   {
    Node temp=first; //created a temp node with the same value as first//
    first=first.next; //shifted the first pointer to first's next//
    first.prev=null; //make first's prev pointer null//
    temp.next=null; //make temp(which is at first)'s next pointer null//
    }
}
```

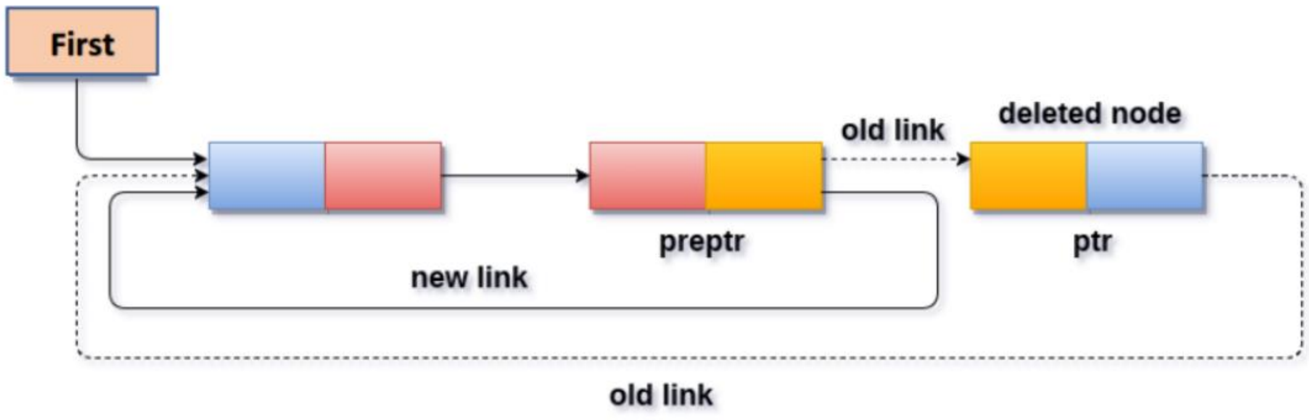


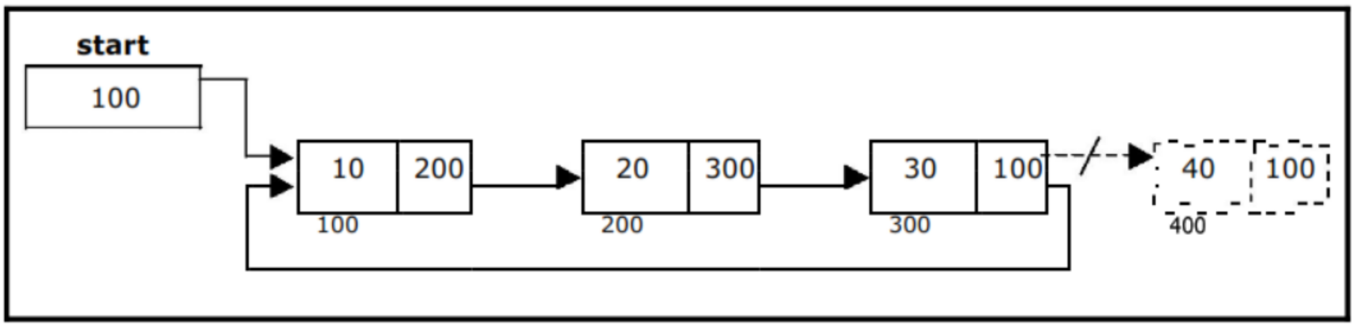


9. Method to Delete the Last Node in Circular LinkedList

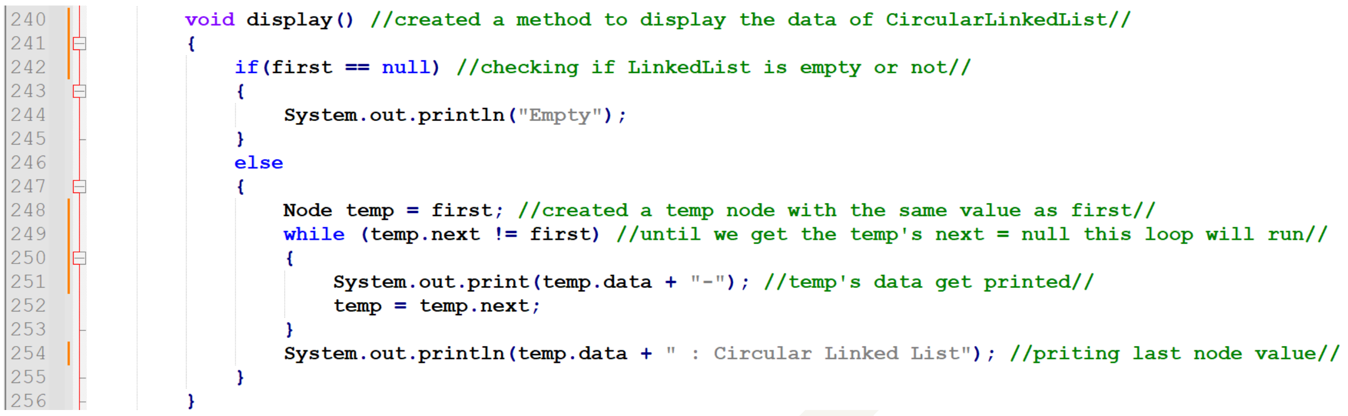


```
        void dellast() //created a method to Delete a Node at Last//
        {
            if (first == null) //checking if LinkedList is empty or not//
            {
                System.out.println("UnderFlow");
            }
            else if(first.next==first) //checking if LL has only 1 node//
            {
                first=null; //then first shall be null//
            }
            else
            {
                Node temp = first; //created a temp node with the same value as first//
                while (temp.next.next != first) //until we get the temp's next's next = null this loop will run/
                {
                    temp = temp.next;
                }
                Node del=temp.next;//to release memory of deleted node
                temp.next = first; //temp's next shall be equal to first//
                System.out.println(del.data+" is deleted from last");
                del.next=null;
            }
        }
```





10. Display Method for Circular LinkedList



```
void display() //created a method to display the data of CircularLinkedList//
{
    if(first == null) //checking if LinkedList is empty or not//
    {
        System.out.println("Empty");
    }
    else
    {
        Node temp = first; //created a temp node with the same value as first//
        while (temp.next != first) //until we get the temp's next = null this loop will run//
        {
            System.out.print(temp.data + "-"); //temp's data get printed//
            temp = temp.next;
        }
        System.out.println(temp.data + " : Circular Linked List"); //priting last node value//
    }
}
```

11. Java Program to InstertAtFirst, InstertAtLast, DeleteFirst & DeleteLast in Circular LinkedList

```java
class Run7
{
    public static void main(String args[]) {
        CircularLinkedList L = new CircularLinkedList();
        L.insertlast(10);
        L.display();
        L.insertlast(20);
        L.display();
        L.delTargetvalue(20);
        L.display();
        L.insertValuebefore(10,10);
        L.display();
        L.insertFirst(10);
        L.display();
        L.insertFirst(20);
        L.display();
        L.insertFirst(30);
        L.display();
        L.insertValuebefore(111,20);
        L.display();
        L.insertFirst(1);
        L.display();
        L.insertFirst(2);
        L.display();
        L.insertFirst(3);
        L.display();
        L.insertlast(10);
        L.display();
        L.insertlast(20);
        L.display();
        L.insertlast(30);
        L.display();
        L.insertValuebefore(111,30);
        L.display();
        L.delTargetvalue(111);
        L.display();
        L.dellast();
        L.display();
        L.delfirst();
        L.display();
    }
    public static class CircularLinkedList //created a class named Circular LL//
    {
        class Node //created a node which contains a data and refrence//
        {
            int data; //the node will have a data of integer type//
            Node next; //it will have a reference variable//

            Node(int data) //created a constructor to create a node//
            {
                this.data = data; //the value of data will be same as entered by user//
                this.next = null; //the next pointer of the node will initialized with
                null//
            }
        }
        Node first = null;
        void insertFirst(int data) //created a method to insert a new node at start//
        {
            Node n = new Node(data); //In insert method we always required to create a
            node//
            if (first == null) //checking if LinkedList is empty or not//
            {
                first = n; //if yes then first shall be directly pointing to n//
                n.next = n; //and new node's next should be pointing n due to CLL//
            }
            else //if LinkedList has more values//
            {
```

```
            n.next = first; //new node n's next shall be equal to first//
            Node temp = first; //created a temp node with the same value as first//
            while (temp.next != first) //until we get the temp's next = null this loop will run//
                    temp = temp.next;
                temp.next = n; //now temp's next which is showing old first shall be pointing n//
                first = n; //and now first shall be poing towards n//
        }
        System.out.println(data+" is inserted First");
    }
    void insertlast(int data) //created a method to Insert a Node at Last//
    {
        Node n = new Node(data); //In insert method we always required to create a node//
        if (first == null) //checking if LinkedList is empty or not//
        {
            first = n; //if yes then first shall be directly pointing to n//
            n.next = n; //and new node's next should be pointing n due to CLL//
        }
        else //otherwise//
        {
            Node temp = first; //created a temp node with the same value as first//
            while (temp.next != first)  //until we get the temp's next = null this loop will run//
                temp = temp.next;
            temp.next = n; //now temp's next which is showing old first shall be pointing n//
            n.next = first; //new node n's next shall be equal to first//
        }
        System.out.println(data+" is inserted Last");
    }
    void delfirst() //created a method to Delete a Node at Last//
    {
        if (first == null) //checking if LinkedList is empty or not//
        {
            System.out.println("UnderFlow");
        }
        else if(first.next==first) //checking if LL has only 1 node//
        {
            first=null; //then first shall be null//
        }
        else //otherwise//
        {
            Node del = first;//to release memory of deleted node
            Node temp = first; //created a temp node with the same value as first//
            while (temp.next != first) //until we get the temp's next = null this loop will run//
            {
                temp = temp.next;
            }
            temp.next = first.next; //then temp's next shall be equal to first's next//
            first = first.next; //first shall be equal to first's next//
            del.next = null;
            System.out.println(del.data+" is deleted from first");
        }
    }
    void dellast() //created a method to Delete a Node at Last//
    {
        if (first == null) //checking if LinkedList is empty or not//
        {
            System.out.println("UnderFlow");
        }
        else if(first.next==first) //checking if LL has only 1 node//
        {
            first=null; //then first shall be null//
            }
            else
            {
                Node temp = first; //created a temp node with the same value as first//
                while (temp.next.next != first) //until we get the temp's next's next =
                null this loop will run//
                {
                    temp = temp.next;
                }
                Node del=temp.next;//to release memory of deleted node
                temp.next = first; //temp's next shall be equal to first//
                System.out.println(del.data+" is deleted from last");
                del.next=null;
            }
        }
        void insertValuebefore(int data,int target) //created a method insert a new node
        before values//
        {
            Node new_node=new Node(data); //In insert method we always required to
            create a node//
            int flag=0; //used a flag variable to check the value exists in LL or not//
            if(first==null) //checking that LinkedList is empty or not//
            {
                System.out.println("Empty Circular Linked List, So cant insert value
                before given value ");
            }
            else //otherwise//
            {
                Node temp=first; //created a temp node with the same value as first//
                do
                {
                    if(temp.data==target) //if we get temp's data equal to target//
                    {
                        flag=1; //the flag value will become 1//
                    }
                    temp=temp.next;
                }while(temp!=first); //untill temp again becomes first in CLL//
                if (flag==1) //now if flag is 1 means value matching//
                {
                    if (first.data==target && first.next==first) //checking for 1 node
                    condition//
                    {
                        new_node.next=first; //new node's next shall be point to first//
                        first.next=new_node; //first's next shall be new node//
                        first=new_node; //first now can point to new node//
                    }
                    else if (first.data==target && first.next!=first) //otherwise//
                    {
                        new_node.next=first; //new node's next eual to first//
                        while(temp.next!=first) //until we get the temp's next = null
                        this loop will run//
                        {
                            temp=temp.next;
                        }
                        temp.next=new_node; //temp's next shall be equal to new node//
                        first=new_node; //first shall now point to new node//
                    }
                    else
                    {
                        //Node temp=first;
                        while(temp.next.data!=target)//until we get temp's next's data
                        will be equal to target//
                        {
                            temp=temp.next;
                        }
                        new_node.next=temp.next; // new node's next equal to temp's
                        next//
                    temp.next=new_node; //temp's next will be equal to new node//
                }
            }
            System.out.println("Target Value does not exist");
        }
    }
    void delTargetvalue(int target) //created a method delete a value//
    {
        int flag=0; //used a flag variable to check the value exists in LL or not//
        if(first==null) //checking that LinkedList is empty or not//
        {
            System.out.println("Empty Circular Linked List, So cant insert value before given value ");
        }
        else //otherwise//
        {
            Node temp=first; //created a temp node with the same value as first//
            do
            {
                if(temp.data==target) //if we get temp's data equal to target//
                {
                    flag=1; //the flag value will become 1//
                }
                temp=temp.next;
            }while(temp!=first); //untill temp again becomes first in CLL//
            if (flag==1) //now if flag is 1 means value matching//
            {
                if (first.data==target && first.next==first) //checking for 1 node condition//
                {
                    first=null;
                }
                else if (first.data==target && first.next!=first) //otherwise//
                {
                    Node del=first; //created a new node del equal to first//
                    while(temp.next!=first) //until we get the temp's next = null this loop will run//
                    {
                        temp=temp.next;
                    }
                    temp.next=first.next; //temp's next equal to first's next//
                    first=first.next; //first shall be now equal to first's next//
                    del.next=null; //del's next becomes null to break link//
                }
                else
                {
                    //Node temp=first;
                    while(temp.next.data!=target) //until we get temp's next's data will be equal to target//
                    {
                        temp=temp.next;
                    }
                    temp.next=temp.next.next; //temp's next shall be equal to temp's next's next//
                }
            }
            System.out.println("Target Value does not exist");
        }
    }
    void display() //created a method to display the data of CircularLinkedList//
    {
        if(first == null) //checking if LinkedList is empty or not//
        {
            System.out.println("Empty");
        }
        else
        {
                Node temp = first; //created a temp node with the same value as first//
                while (temp.next != first) //until we get the temp's next = null this loop will run//
                {
                    System.out.print(temp.data + "-"); //temp's data get printed//
                    temp = temp.next;
                }
                System.out.println(temp.data + " : Circular Linked List"); //priting last node value//
            }
        }
    }
}
```