DATA STRUCTURES (017013292)
Semester – II
Chapter Name: DOUBLY AND CIRCULAR LINK LIST

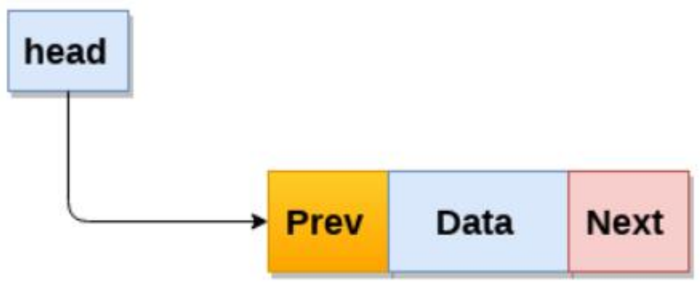
Contents

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

1. Doubly Linked List

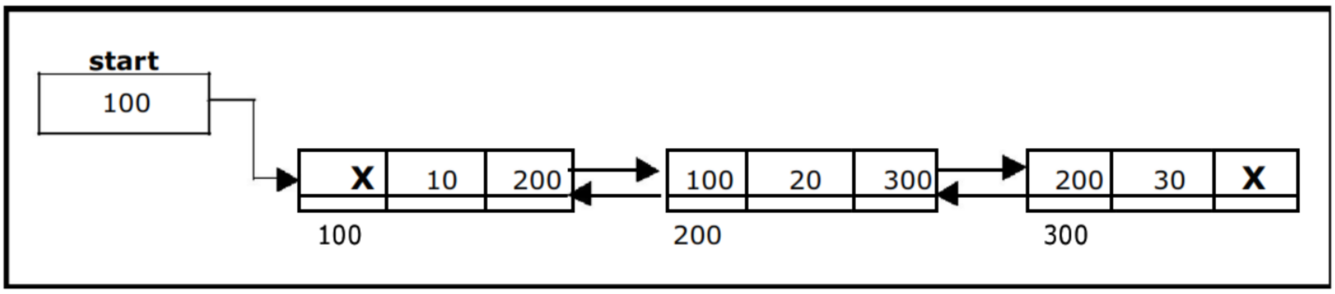
- Doubly linked list is a complex type of linked list in which a node contains a pointer to the previous as well as the next node in the sequence.

- Therefore, in a doubly linked list, a node consists of three parts: node data, pointer to the next node in sequence (next pointer), pointer to the previous node (previous pointer). A sample node in a doubly linked list is shown in the figure.

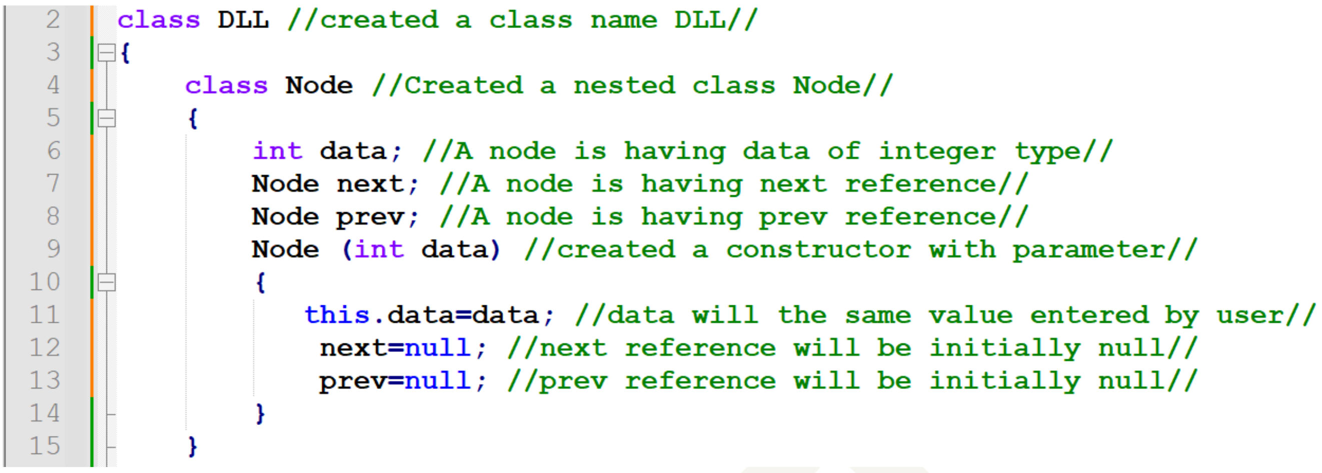


Node

A doubly linked list containing three nodes having numbers having number 10,20 & 30 in their data part, is shown in the following image.



2. Creating a Node of Doubly LinkedList in JAVA



```
class DLL //created a class name DLL//
{
    class Node //Created a nested class Node//
    {
        int data; //A node is having data of integer type//
        Node next; //A node is having next reference//
        Node prev; //A node is having prev reference//
        Node (int data) //created a constructor with parameter//
        {
           this.data=data; //data will the same value entered by user//
            next=null; //next reference will be initially null//
            prev=null; //prev reference will be initially null//
        }
    }
```

3. Advantages of Doubly LinkedList over Singly LinkedList

1. A DLL can be traversed in both forward and backward direction.
2. The delete operation in DLL is more efficient if pointer to the node to be deleted is given.
3. We can quickly insert a new node before a given node.

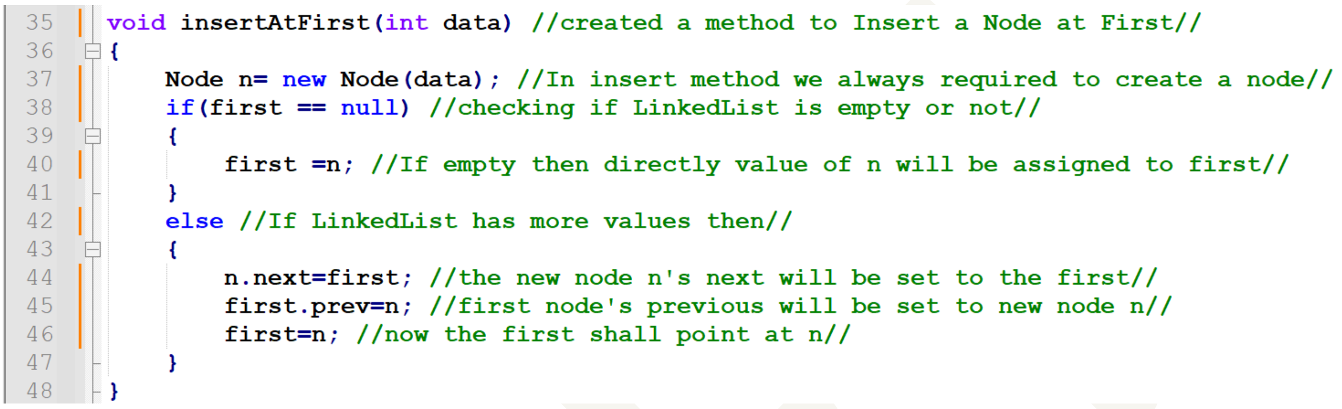
In singly linked list, to delete a node, pointer to the previous node is needed. To get this previous node, sometimes the list is traversed. In DLL, we can get the previous node using previous pointer.

4. Disadvantages of Doubly LinkedList over Singly LinkedList

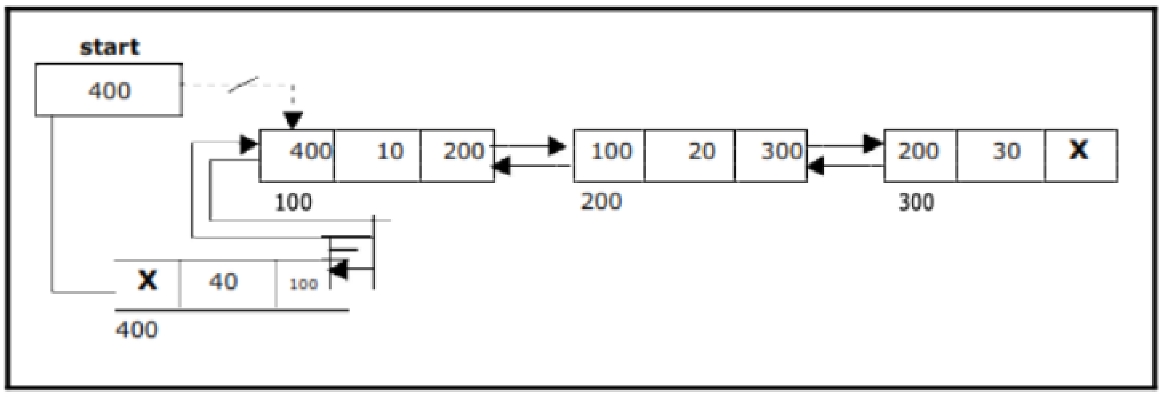
1. Every node of DLL Require extra space for a previous pointer. It is possible to implement DLL with single pointer.
2. All operations require an extra pointer previous to be maintained. For example, in insertion, we need to modify previous pointers together with next pointers. For example in following functions for insertions at different positions, we need 1 or 2 extra steps to set previous pointer.

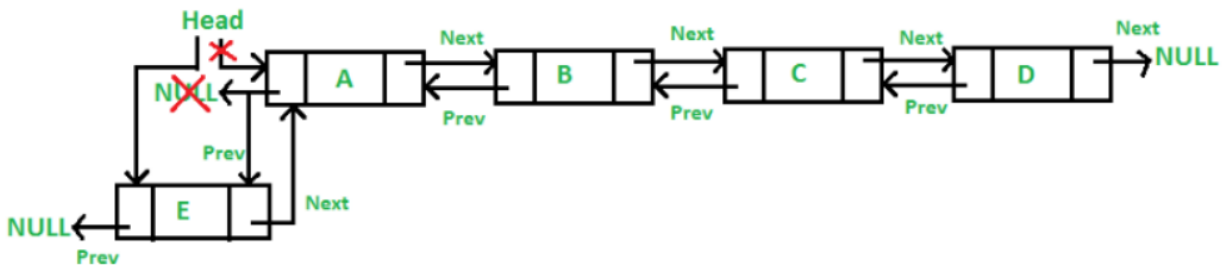
Node = Prev + Data + Next

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

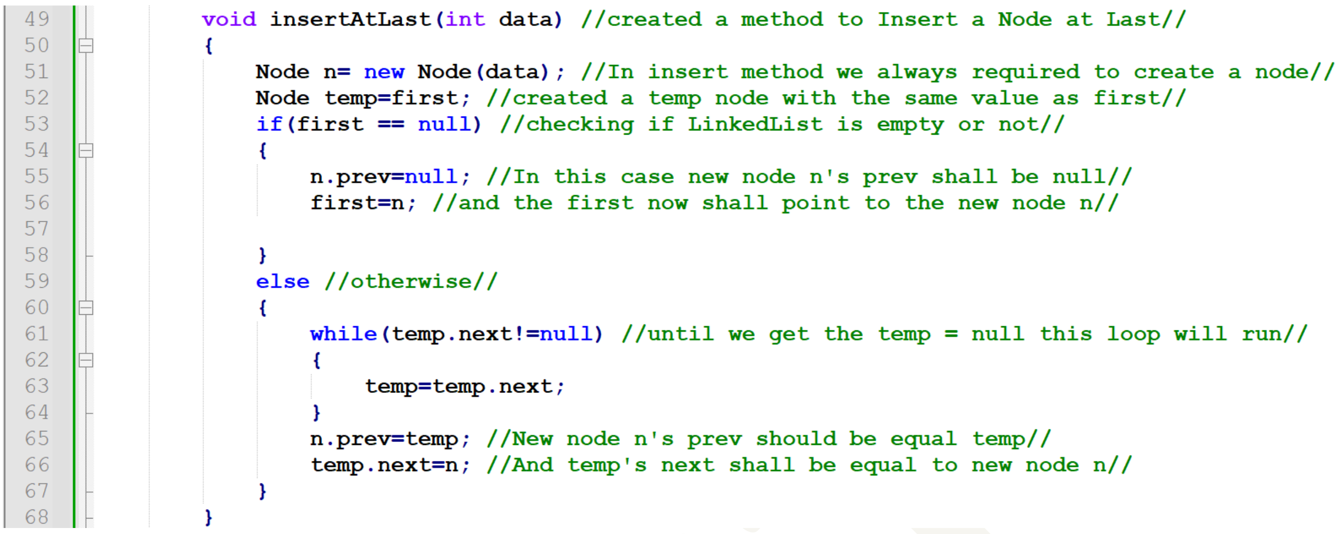


```
void insertAtFirst(int data) //created a method to Insert a Node at First//
{
    Node n= new Node(data); //In insert method we always required to create a node//
    if(first == null) //checking if LinkedList is empty or not//
    {
        first =n; //If empty then directly value of n will be assigned to first//
    }
    else //If LinkedList has more values then//
    {
        n.next=first; //the new node n's next will be set to the first//
        first.prev=n; //first node's previous will be set to new node n//
        first=n; //now the first shall point at n//
    }
}
```

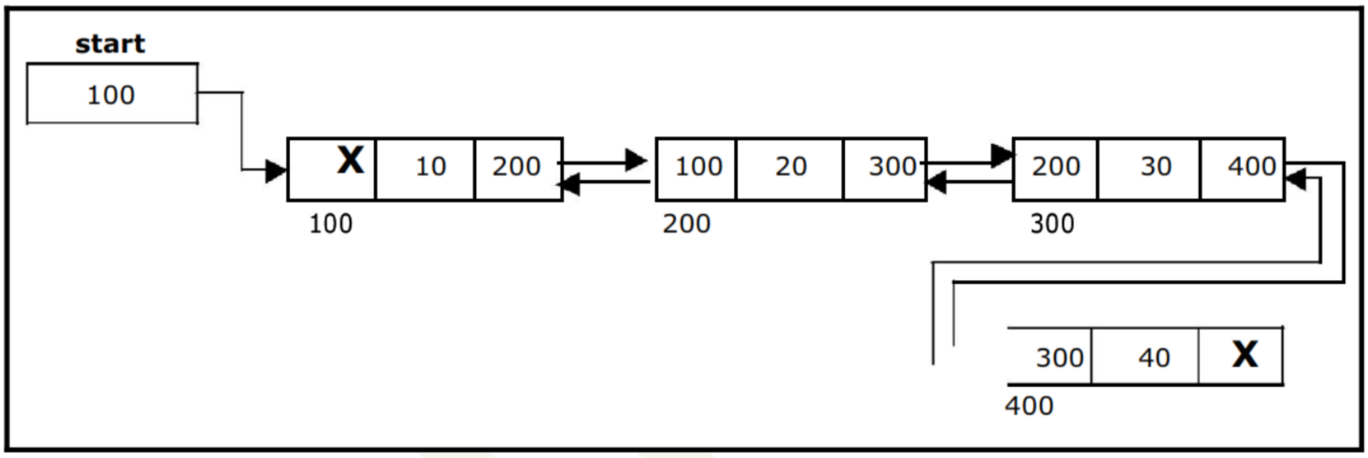


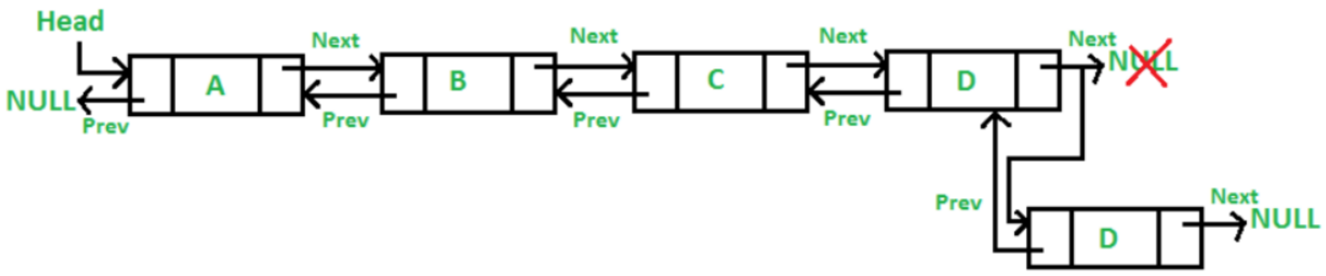


6. Method to Insert a New Node (n) at Last of Doubly LinkedList

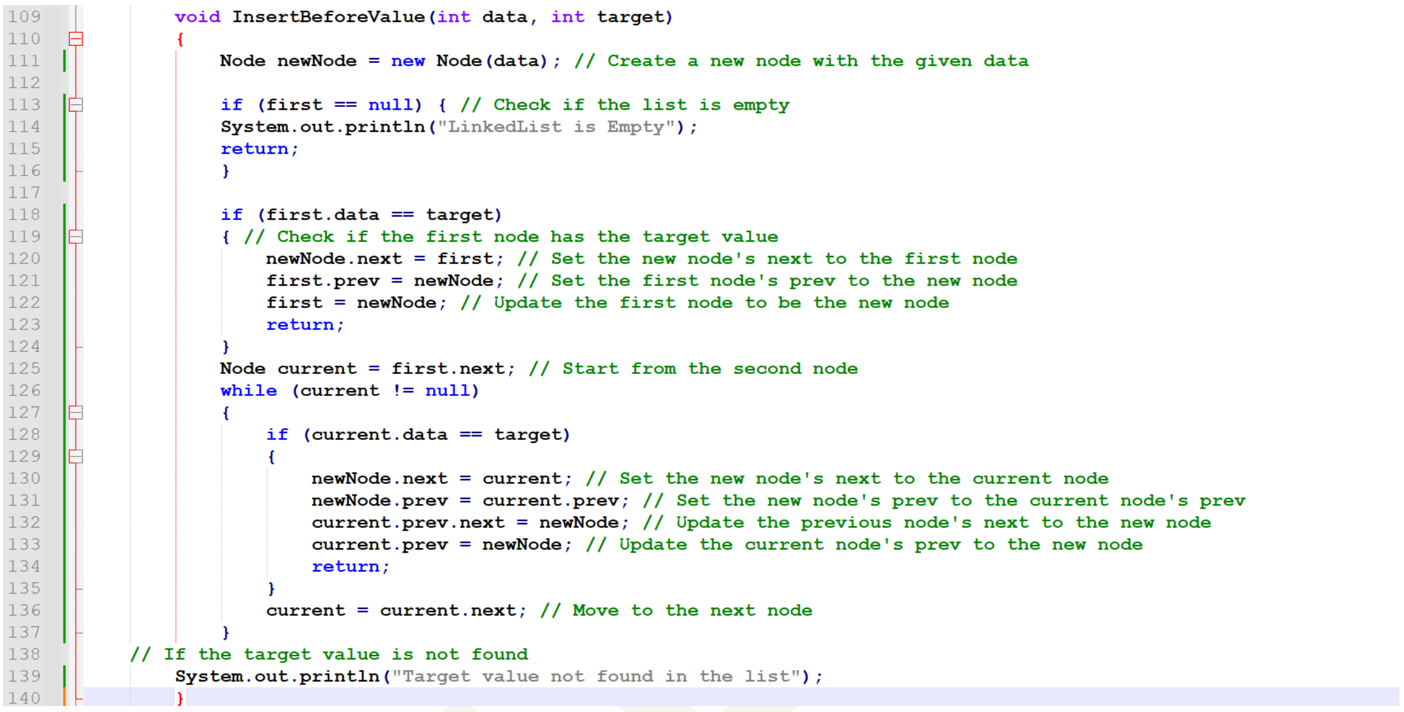


```
void insertAtLast(int data) //created a method to Insert a Node at Last//
{
    Node n= new Node(data); //In insert method we always required to create a node//
    Node temp=first; //created a temp node with the same value as first//
    if(first == null) //checking if LinkedList is empty or not//
    {
        n.prev=null; //In this case new node n's prev shall be null//
        first=n; //and the first now shall point to the new node n//

    }
    else //otherwise//
    {
        while(temp.next!=null) //until we get the temp = null this loop will run//
        {
            temp=temp.next;
        }
        n.prev=temp; //New node n's prev should be equal temp//
        temp.next=n; //And temp's next shall be equal to new node n//
    }
}
```



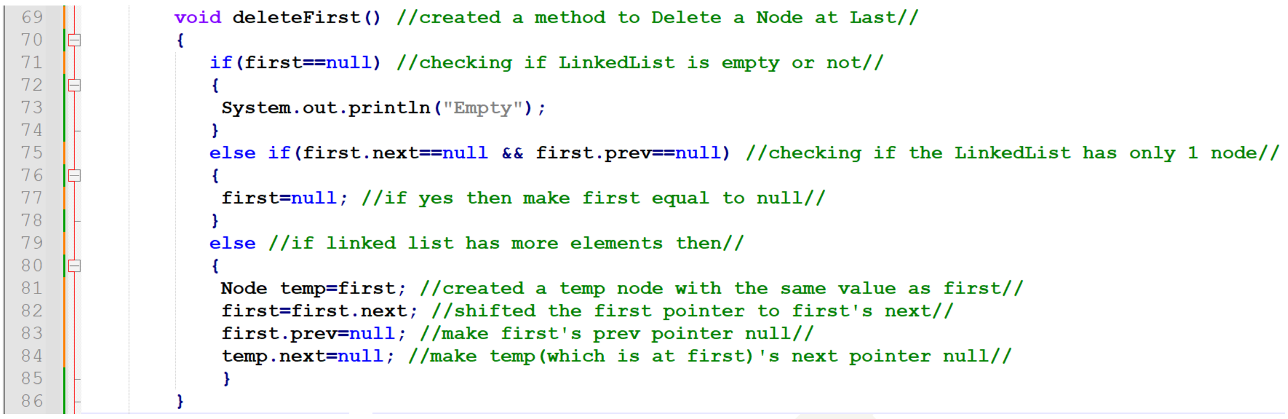


7. Method to Insert a New Node Before Particular Target Value in Doubly Linked List

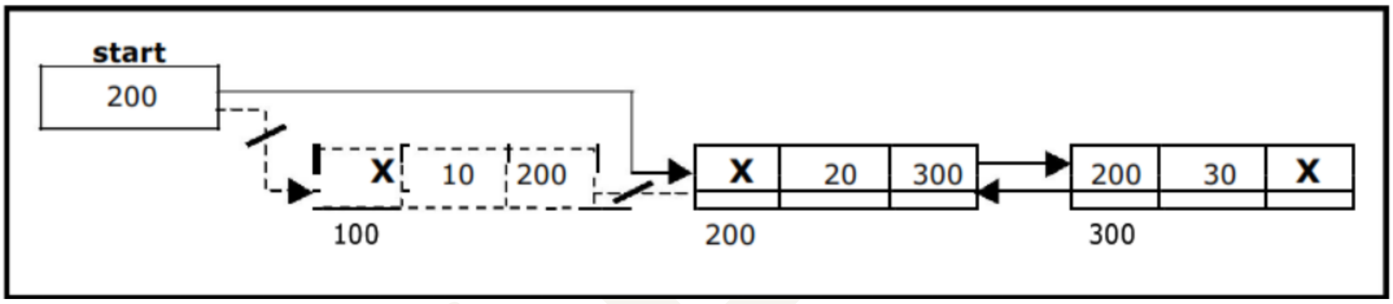


```java
    void InsertBeforeValue(int data, int target)
    {
        Node newNode = new Node(data); // Create a new node with the given data

        if (first == null) { // Check if the list is empty
        System.out.println("LinkedList is Empty");
        return;
        }

        if (first.data == target)
        { // Check if the first node has the target value
            newNode.next = first; // Set the new node's next to the first node
            first.prev = newNode; // Set the first node's prev to the new node
            first = newNode; // Update the first node to be the new node
            return;
        }
        Node current = first.next; // Start from the second node
        while (current != null)
        {
            if (current.data == target)
            {
                newNode.next = current; // Set the new node's next to the current node
                newNode.prev = current.prev; // Set the new node's prev to the current node's prev
                current.prev.next = newNode; // Update the previous node's next to the new node
                current.prev = newNode; // Update the current node's prev to the new node
                return;
            }
            current = current.next; // Move to the next node
        }
// If the target value is not found
    System.out.println("Target value not found in the list");
    }
```

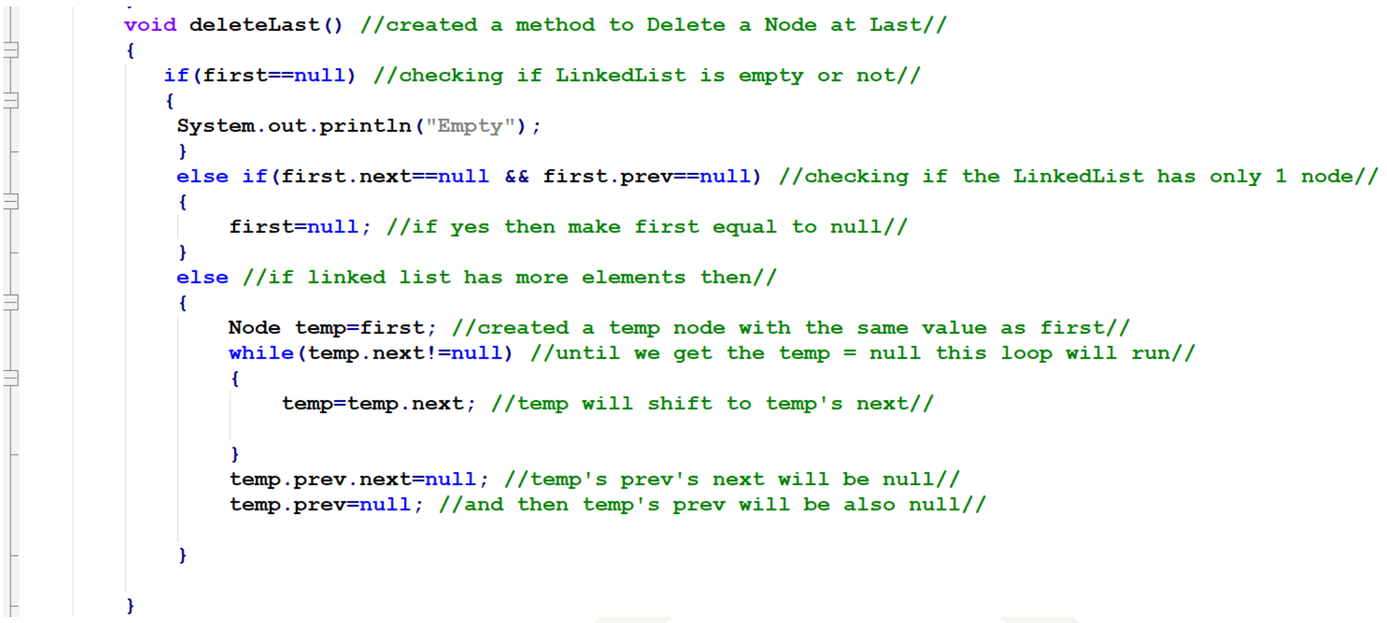
8. Method to Delete the First Node in Doubly LinkedList



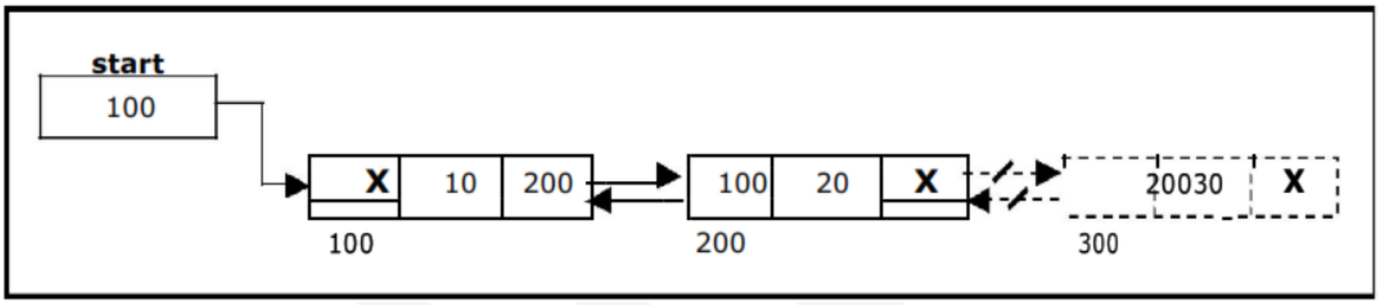
```
        void deleteFirst() //created a method to Delete a Node at Last//
        {
           if(first==null) //checking if LinkedList is empty or not//
           {
            System.out.println("Empty");
           }
           else if(first.next==null && first.prev==null) //checking if the LinkedList 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//
            }
        }
```



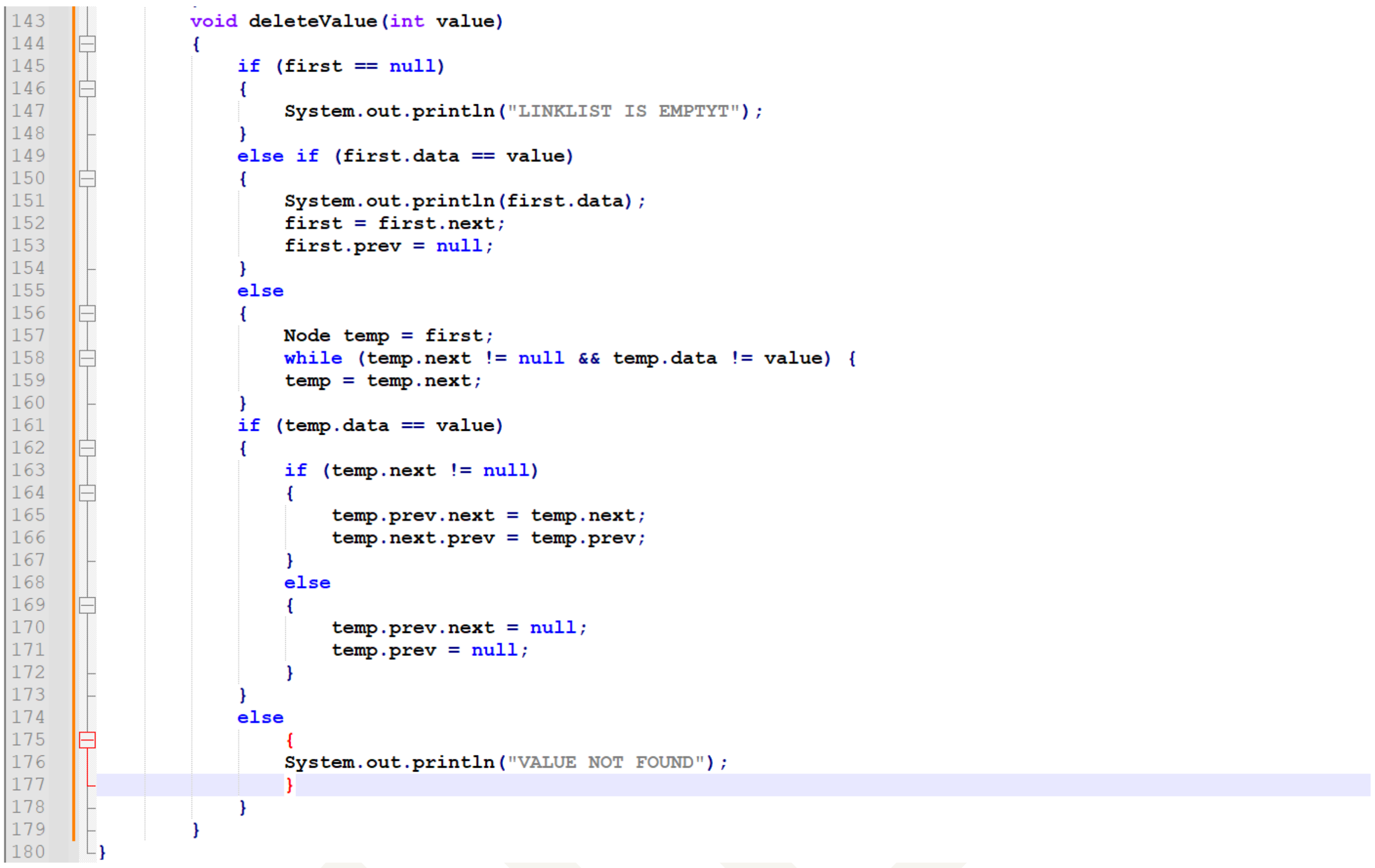
9. Method to Delete the Last Node in Doubly LinkedList



```
void deleteLast() //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//
        while(temp.next!=null) //until we get the temp = null this loop will run//
        {
            temp=temp.next; //temp will shift to temp's next//

        }
        temp.prev.next=null; //temp's prev's next will be null//
        temp.prev=null; //and then temp's prev will be also null//

    }

}
```

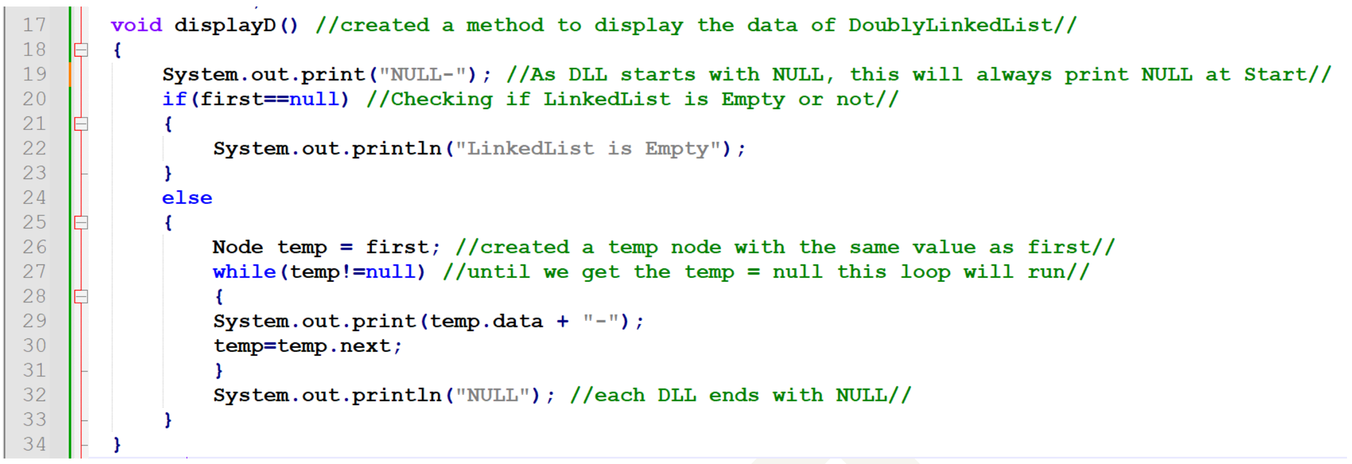


10. Method to Delete a Particular Value in Doubly LinkedList



```
        void deleteValue(int value)
        {
            if (first == null)
            {
                System.out.println("LINKLIST IS EMPTYT");
            }
            else if (first.data == value)
            {
                System.out.println(first.data);
                first = first.next;
                first.prev = null;
            }
            else
            {
                Node temp = first;
                while (temp.next != null && temp.data != value) {
                temp = temp.next;
            }
            if (temp.data == value)
            {
                if (temp.next != null)
                {
                    temp.prev.next = temp.next;
                    temp.next.prev = temp.prev;
                }
                else
                {
                    temp.prev.next = null;
                    temp.prev = null;
                }
            }
            else
                {
                System.out.println("VALUE NOT FOUND");
                }
            }
        }
}
```

11. Display Method for Doubly LinkedList



```
void displayD() //created a method to display the data of DoublyLinkedList//
{
    System.out.print("NULL-"); //As DLL starts with NULL, this will always print NULL at Start//
    if(first==null) //Checking if LinkedList is Empty or not//
    {
        System.out.println("LinkedList is Empty");
    }
    else
    {
        Node temp = first; //created a temp node with the same value as first//
        while(temp!=null) //until we get the temp = null this loop will run//
        {
        System.out.print(temp.data + "-");
        temp=temp.next;
        }
        System.out.println("NULL"); //each DLL ends with NULL//
    }
}
```

12. Java Program to InstertAtFirst, InstertAtLast, DeleteFirst & DeleteLast in Doubly LinkedList

```java
import java.util.*;
class DLL //created a class name DLL//
{
    class Node //Created a nested class Node//
    {
        int data; //A node is having data of integer type//
        Node next; //A node is having next reference//
        Node prev; //A node is having prev reference//
        Node (int data) //created a constructor with parameter//
        {
           this.data=data; //data will the same value entered by user//
            next=null; //next reference will be initially null//
            prev=null; //prev reference will be initially null//
        }
    }
    Node first=null;
        void displayD() //created a method to display the data of DoublyLinkedList//
        {
            System.out.print("NULL-"); //As DLL starts with NULL, this will always print
            NULL at Start//
            if(first==null) //Checking if LinkedList is Empty or not//
            {
                System.out.println("LinkedList is Empty");
            }
            else
            {
                Node temp = first; //created a temp node with the same value as first//
                while(temp!=null) //until we get the temp = null this loop will run//
                {
                System.out.print(temp.data + "-");
                temp=temp.next;
                }
                System.out.println("NULL"); //each DLL ends with NULL//
            }
        }
        void insertAtFirst(int data) //created a method to Insert a Node at First//
        {
            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 empty then directly value of n will be assigned to first//
            }
            else //If LinkedList has more values then//
            {
                n.next=first; //the new node n's next will be set to first//
                first.prev=n; //first prev will be set to n//
                first=n; //now the first shall point at n//
            }
        }
        void insertAtLast(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//
            Node temp=first; //created a temp node with the same value as first//
            if(first == null) //checking if LinkedList is empty or not//
            {
                n.prev=null; //In this case new node n's prev shall be null//
                first=n; //and the first now shall point to the new node n//

            }
            else //otherwise//
            {
                while(temp.next!=null) //until we get the temp = null this loop will
                run//
                {
                    temp=temp.next;
            }
            n.prev=temp; //New node n's prev should be equal temp//
            temp.next=n; //And temp's next shall be equal to new node n//
        }
    }
    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//
        }
    }
    void deleteLast() //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//
            while(temp.next!=null) //until we get the temp = null this loop will run//
            {
                temp=temp.next; //temp will shift to temp's next//

            }
            temp.prev.next=null; //temp's prev's next will be null//
            temp.prev=null; //and then temp's prev will be also null//

        }

    }
    void InsertBeforeValue(int data, int target)
    {
        Node newNode = new Node(data); // Create a new node with the given data

        if (first == null) { // Check if the list is empty
        System.out.println("LinkedList is Empty");
        return;
        }

        if (first.data == target)
        { // Check if the first node has the target value
            newNode.next = first; // Set the new node's next to the first node
            first.prev = newNode; // Set the first node's prev to the new node
            first = newNode; // Update the first node to be the new node
            return;
        }
        Node current = first.next; // Start from the second node
```

```
                while (current != null)
                {
                    if (current.data == target)
                    {
                        newNode.next = current; // Set the new node's next to the current node
                        newNode.prev = current.prev; // Set the new node's prev to the current node's prev
                        current.prev.next = newNode; // Update the previous node's next to the new node
                        current.prev = newNode; // Update the current node's prev to the new node
                        return;
                    }
                    current = current.next; // Move to the next node
                }
        // If the target value is not found
            System.out.println("Target value not found in the list");
            }
            void deleteValue(int value)
            {
                if (first == null)
                {
                    System.out.println("LINKLIST IS EMPTYT");
                }
                else if (first.data == value)
                {
                    System.out.println(first.data);
                    first = first.next;
                    first.prev = null;
                }
                else
                {
                    Node temp = first;
                    while (temp.next != null && temp.data != value) {
                    temp = temp.next;
                }
                if (temp.data == value)
                {
                    if (temp.next != null)
                    {
                        temp.prev.next = temp.next;
                        temp.next.prev = temp.prev;
                    }
                    else
                    {
                        temp.prev.next = null;
                        temp.prev = null;
                    }
                }
                else
                    {
                    System.out.println("VALUE NOT FOUND");
                    }
                }
            }
    }
    class Run11 //created the run class to call the method by making objects//
    {
        public static void main(String args[])
        {
            DLL d = new DLL();
            d.displayD();
            d.insertAtFirst(12);
            d.displayD();
            d.insertAtLast(22);
            d.displayD();
        d.insertAtFirst(11);
        d.displayD();
        d.insertAtLast(222);
        d.displayD();
        d.InsertBeforeValue(1,222);
        d.displayD();
        d.deleteFirst();
        d.displayD();
        d.deleteLast();
        d.displayD();
    }
}
```