```java
//Write a Java Method to Insert a Value After a Particular Node//
import java.util.*;
void insertAfterValue(int data,int value)
    {

        int flag = 0;
        if(First==null)
        {
            System.out.println("Linked is empty");
        }
        else
        {
            Node dummy = First;
            while(dummy != null)
            {
                if(dummy.data == value)
                {
                flag = 1;
                }
                dummy = dummy.next;
            }
                if(flag == 0)
                {
                    System.out.println("The asked value is not inside the linked list");
                }
                else
                {
                    Node n = new Node(data);
                    if(First.data == value && First.next== null)
                    {
                        First.next=n;
                    }
                    else if(First.data == value)
                    {
                        n.next=First.next;
                        First.next=n;
                    }
                    else
                    {
                        Node temp=First;
                        while(temp.data!=value)
                        {
                            temp=temp.next;
                        }
                        n.next=temp.next;
                        temp.next=n;
                    }
                }
        }
    }
```

```java
//Write a Java Method to Swap the elements in Singly Linked List//
import java.util.*;
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

public class LinkedList {
    Node head;

    public void swapNodes(int position) {
        if (head == null || head.next == null)
            return;

        if (position == 1) {
            Node prev = null;
            Node current = head;
            Node nextNode = head.next;

            head = nextNode;
            current.next = nextNode.next;
            nextNode.next = current;
        } else {
            int count = 1;
            Node previousNode = null;
            Node currentNode = head;

            while (count < position && currentNode != null) {
                previousNode = currentNode;
                currentNode = currentNode.next;
                count++;
            }

            if (currentNode == null || currentNode.next == null)
                return;

            Node nextNode = currentNode.next;
            previousNode.next = nextNode;
            currentNode.next = nextNode.next;
            nextNode.next = currentNode;
        }
    }

    public void displayList() {
        Node tempNode = head;
        while (tempNode != null) {
            System.out.print(tempNode.data + " ");
            tempNode = tempNode.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        LinkedList linkedList = new LinkedList();

        // Create the linked list
        linkedList.head = new Node(1);
        Node secondNode = new Node(2);
        Node thirdNode = new Node(3);
        Node fourthNode = new Node(4);

        linkedList.head.next = secondNode;
```

```
68            secondNode.next = thirdNode;
69            thirdNode.next = fourthNode;
70
71            System.out.println("Original Linked List:");
72            linkedList.displayList();
73
74            int positionToSwap = 2;
75            linkedList.swapNodes(positionToSwap);
76
77            System.out.println("Linked List after swapping nodes at position " +
              positionToSwap + ":");
78            linkedList.displayList();
79        }
80    }
81
```

```java
//Write a Java Method to Find the Maximum Value out of all the elements the given Linked List//
import java.util.*;
void MaxValue(Node First)
{
    Node temp=First;
    int MaxValue;
    MaxValue=temp.data;
    while(temp!=null)
    {
        if(temp.data>MaxValue)
        {
            MaxValue=temp.data;
        }
        temp=temp.next;
    }
    System.out.println("Max Value is"+MaxValue);
}
```

```java
//Write a Java Method to Find the Product of all the elements the given Linked List//
import java.util.*;
void product(Node First)
{
    Node temp=First;
    int product=1;
    while(temp!=null)
    {
        product=product*temp.data;
        temp=temp.next;
    }
    System.out.println("Product of All elements is"+product);
}

```

```java
//Write a Java Method to Reverse the given Linked List//
import java.util.*;
void reverse()
{
    Node temp1,temp2,temp3;
    temp1=First;
    if(temp1==null)
    {
        System.out.println("Singly Linked List is Empty.");
    }
    else
    {
        temp2=null;
        while(temp1!=null)
        {
            temp3=temp2;
            temp2=temp1;
            temp1=temp1.next;
            temp2.next=temp3;
        }
        First=temp2;
    }
    System.out.println("The Linked List is Reversed")
}
```

```java
//Write a Java Method to Count the Nodes of a Singly Linked List//
import java.util.*;
void count()
{
    Node Temp=First;
    int c=0;
    if(Temp==null)
    {
        System.out.println("Singly Linked List is Empty");
    }
    while(Temp!=null)
    {
        c=c+1;
        Temp=Temp.next;
    }
    System.out.println("Total Count of Nodes:"+c);
}
```

```java
//Write a Java Method to Concate the given Two Linked List//
import java.util.*;
void concat(Node First1,Node First2)
{
    Node temp1,temp2;
    temp1=First1;
    temp2=First2;
    while(temp1.next!=null)
    {
        temp1=temp1.next;
    }
    temp1.next=temp2;
    System.out.println("The Concated List is");
    temp1=First1;
    while(temp1!=null)
    {
        System.out.println(temp1.data+"--");
        temp1=temp1.next;
    }
}
```

```java
//Write a Java Method to Delete all the elements in Singly Linked List using recursion//
class Node {
    int data;
    Node next;

    Node(int data) {
        this.data = data;
        this.next = null;
    }
}

public class LinkedList {
    Node head;

    public void deleteList() {
        deleteListRecursive(head);
        head = null; // Reset the head to null after deleting all nodes
    }

    private void deleteListRecursive(Node node) {
        if (node == null)
            return;

        deleteListRecursive(node.next);
        node.next = null; // Remove the reference to the next node
    }

    public void displayList() {
        Node tempNode = head;
        while (tempNode != null) {
            System.out.print(tempNode.data + " ");
            tempNode = tempNode.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        LinkedList linkedList = new LinkedList();

        // Create the linked list
        linkedList.head = new Node(1);
        Node secondNode = new Node(2);
        Node thirdNode = new Node(3);
        Node fourthNode = new Node(4);

        linkedList.head.next = secondNode;
        secondNode.next = thirdNode;
        thirdNode.next = fourthNode;

        System.out.println("Original Linked List:");
        linkedList.displayList();

        linkedList.deleteList();

        System.out.println("Linked List after deletion:");
        linkedList.displayList();
    }
}

```

```java
//Write a Java Program to Print Node's data which has only even value//
import java.util.Scanner;

class SingyLinkedList
{
    Scanner sc = new Scanner(System.in);
    class Node
    {
        int data;
        Node next;
        Node()
        {
            System.out.println("Insert the data part of a new node - ");
            data = sc.nextInt();
            next = null;
        }

    }

    Node head = null;

    void insertatLast()
    {
        Node newNode = new Node();
        if(head == null)
        {
            head = newNode;
        }
        else
        {
            Node tail = head;
            while(tail.next != null)
            {
                tail = tail.next;
            }
            tail.next = newNode;
        }
    }

    void displayEven()
    {
        if(head == null)
        {
            System.out.println("LL is Empty");
        }
        else
        {
            Node temp = head;
            while(temp != null)
            {
                if((temp.data)%2 == 0)
                {
                    System.out.print(temp.data + "-->");
                }
                temp = temp.next;
            }
            System.out.println("null");

        }
    }


    void display()
    {
        if(head == null)
        {
            System.out.println("LL is Empty");
```

```
        }
        else
        {
            Node temp = head;
            while(temp != null)
            {
                System.out.print(temp.data + "-->");
                temp = temp.next;
            }
            System.out.println("null");

        }
    }
}


class PrintEvenValuedNodes
{
    public static void main(String args[])
    {
        SingyLinkedList  s = new SingyLinkedList();
        s.insertatLast();
        s.insertatLast();
        s.insertatLast();
        s.insertatLast();
        s.insertatLast();
        s.insertatLast();
        System.out.println("SinglyLinkedList is as below -");
        s.display();
        System.out.println("SinglyLinkedList with even values is as below -");
        s.displayEven();
    }
}
```