

```

1  //Write a Java Method to Insert a Value After a Particular Node//
2  import java.util.*;
3  void insertAfterValue(int data,int value)
4  {
5
6      int flag = 0;
7      if(First==null)
8      {
9          System.out.println("Linked is empty");
10     }
11     else
12     {
13         Node dummy = First;
14         while(dummy != null)
15         {
16             if(dummy.data == value)
17             {
18                 flag = 1;
19             }
20             dummy = dummy.next;
21         }
22         if(flag == 0)
23         {
24             System.out.println("The asked value is not inside the linked list");
25         }
26         else
27         {
28             Node n = new Node(data);
29             if(First.data == value && First.next== null)
30             {
31                 First.next=n;
32             }
33             else if(First.data == value)
34             {
35                 n.next=First.next;
36                 First.next=n;
37             }
38             else
39             {
40                 Node temp=First;
41                 while(temp.data!=value)
42                 {
43                     temp=temp.next;
44                 }
45                 n.next=temp.next;
46                 temp.next=n;
47             }
48         }
49     }
50 }

```

```

1  //Write a Java Method to Swap the elements in Singly Linked List//
2  import java.util.*;
3  class Node {
4      int data;
5      Node next;
6
7      Node(int data) {
8          this.data = data;
9          this.next = null;
10     }
11 }
12
13 public class LinkedList {
14     Node head;
15
16     public void swapNodes(int position) {
17         if (head == null || head.next == null)
18             return;
19
20         if (position == 1) {
21             Node prev = null;
22             Node current = head;
23             Node nextNode = head.next;
24
25             head = nextNode;
26             current.next = nextNode.next;
27             nextNode.next = current;
28         } else {
29             int count = 1;
30             Node previousNode = null;
31             Node currentNode = head;
32
33             while (count < position && currentNode != null) {
34                 previousNode = currentNode;
35                 currentNode = currentNode.next;
36                 count++;
37             }
38
39             if (currentNode == null || currentNode.next == null)
40                 return;
41
42             Node nextNode = currentNode.next;
43             previousNode.next = nextNode;
44             currentNode.next = nextNode.next;
45             nextNode.next = currentNode;
46         }
47     }
48
49     public void displayList() {
50         Node tempNode = head;
51         while (tempNode != null) {
52             System.out.print(tempNode.data + " ");
53             tempNode = tempNode.next;
54         }
55         System.out.println();
56     }
57
58     public static void main(String[] args) {
59         LinkedList linkedList = new LinkedList();
60
61         // Create the linked list
62         linkedList.head = new Node(1);
63         Node secondNode = new Node(2);
64         Node thirdNode = new Node(3);
65         Node fourthNode = new Node(4);
66
67         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 " +
78             positionToSwap + ":");
79         linkedList.displayList();
80     }
81 }
```

```
1 //Write a Java Method to Find the Maximum Value out of all the elements the given Linked
  List//
2 import java.util.*;
3 void MaxValue(Node First)
4 {
5     Node temp=First;
6     int MaxValue;
7     MaxValue=temp.data;
8     while(temp!=null)
9     {
10         if(temp.data>MaxValue)
11         {
12             MaxValue=temp.data;
13         }
14         temp=temp.next;
15     }
16     System.out.println("Max Value is"+MaxValue);
17 }
```

```
1 //Write a Java Method to Find the Product of all the elements the given Linked List//
2 import java.util.*;
3 void product(Node First)
4 {
5     Node temp=First;
6     int product=1;
7     while(temp!=null)
8     {
9         product=product*temp.data;
10        temp=temp.next;
11    }
12    System.out.println("Product of All elements is"+product);
13 }
14
```

```
1 //Write a Java Method to Reverse the given Linked List//
2 import java.util.*;
3 void reverse()
4 {
5     Node temp1,temp2,temp3;
6     temp1=First;
7     if(temp1==null)
8     {
9         System.out.println("Singly Linked List is Empty.");
10    }
11    else
12    {
13        temp2=null;
14        while(temp1!=null)
15        {
16            temp3=temp2;
17            temp2=temp1;
18            temp1=temp1.next;
19            temp2.next=temp3;
20        }
21        First=temp2;
22    }
23    System.out.println("The Linked List is Reversed")
24 }
```

```
1 //Write a Java Method to Count the Nodes of a Singly Linked List//
2 import java.util.*;
3 void count()
4 {
5     Node Temp=First;
6     int c=0;
7     if(Temp==null)
8     {
9         System.out.println("Singly Linked List is Empty");
10    }
11    while(Temp!=null)
12    {
13        c=c+1;
14        Temp=Temp.next;
15    }
16    System.out.println("Total Count of Nodes:"+c);
17 }
```

```
1 //Write a Java Method to Concat the given Two Linked List//
2 import java.util.*;
3 void concat(Node First1,Node First2)
4 {
5     Node temp1,temp2;
6     temp1=First1;
7     temp2=First2;
8     while(temp1.next!=null)
9     {
10         temp1=temp1.next;
11     }
12     temp1.next=temp2;
13     System.out.println("The Concated List is");
14     temp1=First1;
15     while(temp1!=null)
16     {
17         System.out.println(temp1.data+"--");
18         temp1=temp1.next;
19     }
20 }
```



```
1  //Write a Java Method to Delete all the elements in Singly Linked List using recursion//
2  class Node {
3      int data;
4      Node next;
5
6      Node(int data) {
7          this.data = data;
8          this.next = null;
9      }
10 }
11
12 public class LinkedList {
13     Node head;
14
15     public void deleteList() {
16         deleteListRecursive(head);
17         head = null; // Reset the head to null after deleting all nodes
18     }
19
20     private void deleteListRecursive(Node node) {
21         if (node == null)
22             return;
23
24         deleteListRecursive(node.next);
25         node.next = null; // Remove the reference to the next node
26     }
27
28     public void displayList() {
29         Node tempNode = head;
30         while (tempNode != null) {
31             System.out.print(tempNode.data + " ");
32             tempNode = tempNode.next;
33         }
34         System.out.println();
35     }
36
37     public static void main(String[] args) {
38         LinkedList linkedList = new LinkedList();
39
40         // Create the linked list
41         linkedList.head = new Node(1);
42         Node secondNode = new Node(2);
43         Node thirdNode = new Node(3);
44         Node fourthNode = new Node(4);
45
46         linkedList.head.next = secondNode;
47         secondNode.next = thirdNode;
48         thirdNode.next = fourthNode;
49
50         System.out.println("Original Linked List:");
51         linkedList.displayList();
52
53         linkedList.deleteList();
54
55         System.out.println("Linked List after deletion:");
56         linkedList.displayList();
57     }
58 }
59
```

```

1 //Write a Java Program to Print Node's data which has only even value//
2 import java.util.Scanner;
3
4 class SingyLinkedList
5 {
6     Scanner sc = new Scanner(System.in);
7     class Node
8     {
9         int data;
10        Node next;
11        Node()
12        {
13            System.out.println("Insert the data part of a new node - ");
14            data = sc.nextInt();
15            next = null;
16        }
17    }
18
19    Node head = null;
20
21    void insertatLast()
22    {
23        Node newNode = new Node();
24        if(head == null)
25        {
26            head = newNode;
27        }
28        else
29        {
30            Node tail = head;
31            while(tail.next != null)
32            {
33                tail = tail.next;
34            }
35            tail.next = newNode;
36        }
37    }
38
39    void displayEven()
40    {
41        if(head == null)
42        {
43            System.out.println("LL is Empty");
44        }
45        else
46        {
47            Node temp = head;
48            while(temp != null)
49            {
50                if((temp.data)%2 == 0)
51                {
52                    System.out.print(temp.data + "-->");
53                }
54                temp = temp.next;
55            }
56            System.out.println("null");
57        }
58    }
59
60    void display()
61    {
62        if(head == null)
63        {
64            System.out.println("LL is Empty");
65        }
66    }
67

```

```

68         }
69     else
70     {
71         Node temp = head;
72         while(temp != null)
73         {
74             System.out.print(temp.data + "-->");
75             temp = temp.next;
76         }
77         System.out.println("null");
78     }
79 }
80 }
81 }
82
83
84 class PrintEvenValuedNodes
85 {
86     public static void main(String args[])
87     {
88         SingyLinkedList s = new SingyLinkedList();
89         s.insertatLast();
90         s.insertatLast();
91         s.insertatLast();
92         s.insertatLast();
93         s.insertatLast();
94         s.insertatLast();
95         System.out.println("SinglyLinkedList is as below -");
96         s.display();
97         System.out.println("SinglyLinkedList with even values is as below -");
98         s.displayEven();
99     }
100 }

```