

CircularList.java

```
1 import java.util.Scanner;
2
3 class Circular
4 {
5     class Node {
6         int data;
7         Node next;
8
9         Node(int d) {
10             data = d;
11             next = null;
12         }
13     }
14     Node root, last;
15     Circular()
16     {
17         root=last=null;
18     }
19     public void insertLeft(int e)
20     {
21         Node n=new Node(e);
22         if(root==null)
23         {
24             root=last=n;
25             last.next=root;
26         }
27         else
28         {
29             n.next=root;
30             root=n;
31             last.next=root;
32         }
33     }
34     public void deleteLeft()
35     {
36         if(root==null)
37         {
38             System.out.println("Empty List");
39         }
40         else
41         {
42             Node t=root;
43             root=root.next;
44             last.next=root;
45             System.out.println(t.data+ "Removed ");
46         }
47     }
48     public void insertRight(int e)
49     {
50         Node n=new Node(e);
51         if(root==null)
52         {
53             root=last=n;
54             last.next=root;
55         }
56         else
57         {
58             last.next=n;
59             last=n;
60             last.next=root;
61         }
62     }
```

```

63     public void deleteRight()
64     {
65         if(root==null)
66         {
67             System.out.println("Empty List");
68         }
69         else
70         {
71             Node t=root;
72             Node t2=root;
73             while(t!=last)
74             {
75                 t2=t;
76                 t=t.next;
77             }
78             last=t2;
79             last.next=root;
80             System.out.println(t.data+ "Remove");
81         }
82     }
83 }
84 public void printList()
85 {
86     if(root==null)
87     {
88         System.out.println("Empty List");
89     }
90     else
91     {
92         Node t=root;
93         do
94         {
95             System.out.println(t.data);
96             t=t.next;
97         }while(t!=root);
98     }
99 }
100 }
101 public class CircularList {
102     public static void main(String args[]) {
103         int val, ch;
104         Scanner s = new Scanner(System.in);
105         Circular c = new Circular();
106         do {
107             System.out.println("\n1.Insert Left \n2.Delete Left \n3.Insert Right \n4.Delete
Right \n5.Print List \nEnter choice :");
108             ch = s.nextInt();
109             switch (ch) {
110                 case 1:
111                     System.out.println("Insert Left Node");
112                     val = s.nextInt();
113                     c.insertLeft(val);
114                     break;
115                 case 2:
116                     System.out.println("Delete Left Node");
117                     c.deleteLeft();
118                     break;
119                 case 3:
120                     System.out.println("Insert Right Node");
121                     val = s.nextInt();
122                     c.insertRight(val);
123                     break;

```

CircularList.java

```
124         case 4:
125             System.out.println("Delete Right Node");
126             c.deleteRight();
127             break;
128         case 5:
129             System.out.println("Print LinkList");
130             c.printList();
131             break;
132     }
133 }
134 }while(ch!=0);
135 }
136 }
137 }
138 }
```

DoublyList.java

```

1 import java.util.Scanner;
2
3 class Dll
4 {
5     class Node
6     {
7         int data;
8         Node left,right;
9         Node(int e)
10        {
11            data=e;
12            left=null;
13            right=null;
14        }
15    }
16    Node root,last;
17    Dll()
18    {
19        root=last=null;
20    }
21    public void insertLeft(int e)
22    {
23        Node n=new Node(e);
24        if(root==null)
25        {
26            root=last=n;
27        }
28        else
29        {
30            root.left=n;
31            n.right=root;
32            root=n;
33        }
34    }
35    public void deleteLeft()
36    {
37        if(root==null)
38        {
39            System.out.println("Empty List");
40        }
41        else
42        {
43            Node t=root;
44            root=root.right;
45            root.left=null;
46            System.out.println(t.data+ "Remove");
47        }
48    }
49    public void insertRight(int e)
50    {
51        Node n=new Node(e);
52        if(root==null)
53        {
54            root=last=n;
55        }
56        else
57        {
58            last.right=n;
59            n.left=root;
60            last=n;
61        }
62    }

```

```

63     public void deleteRight()
64     {
65         if(root==null)
66         {
67             System.out.println("Empty List");
68         }
69         else
70         {
71             Node t=last;
72             last=last.left;
73             last.right=null;
74             System.out.println(t.data+ "Remove");
75         }
76     }
77     public void printStart() {
78         Node t=root;
79         while(t!=null)
80         {
81             System.out.println(t.data);
82             t=t.right;
83         }
84     }
85     public void printLast() {
86         Node t=last;
87         while(t!=null)
88         {
89             System.out.println(t.data);
90             t=t.left;
91         }
92     }
93 }
94 }
95 public class DoublyList {
96     public static void main(String args[]) {
97         int value, ch;
98         Scanner s = new Scanner(System.in);
99         Dll d = new Dll();
100         do {
101             System.out.println("\n1.Insert Left \n2.Delete Left \n3.Insert Right \n4.Delete
Right \n5.Print Start \n6.Print Last \nEnter choice :");
102             ch = s.nextInt();
103             switch (ch) {
104                 case 1:
105                     System.out.println("Insert Left Node");
106                     value = s.nextInt();
107                     d.insertLeft(value);
108                     break;
109                 case 2:
110                     System.out.println("Delete Left Node");
111                     d.deleteLeft();
112                     break;
113                 case 3:
114                     System.out.println("Insert Right Node");
115                     value = s.nextInt();
116                     d.insertRight(value);
117                     break;
118                 case 4:
119                     System.out.println("Delete Right Node");
120                     d.deleteRight();
121                     break;
122                 case 5:
123                     System.out.println("Print LinkList Start");

```

DoublyList.java

```
124         d.printStart();
125         break;
126     case 6:
127         System.out.println("Print LinkList Last");
128         d.printLast();
129         break;
130     }
131 }while(ch!=0);
132
133 }
134 }
135
```