

Reverce.java

```

1 import java.util.Scanner;
2
3 class LinkListDS
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;
15
16     LinkListDS()
17     {
18         root = null;
19     }
20     public void insertNode(int e) {
21         Node n = new Node(e);
22         if (root == null) {
23             root = n;
24         } else {
25             n.next = root;
26             root = n;
27         }
28     }
29
30
31     public void deleteNode() {
32         if (root == null) {
33             System.out.println("Link List Empty");
34         } else {
35             Node t = root;
36             root = root.next;
37             System.out.println(t.data + "Remove");
38         }
39     }
40
41     public void printLink() {
42         if (root == null) {
43             System.out.println("Empty List");
44         } else {
45             Node t = root;
46             while (t != null) {
47
48                 System.out.println(t.data);
49                 t = t.next;
50             }
51         }
52     }
53
54     public void sortList(Stack a) {
55         if (root==null) {
56             System.out.println("List is Empty");
57         } else {
58             Node t = root;
59             while (t != null) {
60                 a.push(t.data);
61                 t = t.next;
62             }

```

Reverce.java

```

63         Node s = root;
64         while (s != null) {
65             s.data = a.pop();
66             s = s.next;
67         }
68     }
69     System.out.println();
70 }
71 }
72
73 class Stack {
74     int s[];
75     int top;
76     int maxsize;
77
78     Stack() {
79
80     }
81
82     public Stack(int size) {
83         maxsize = size;
84         s = new int[maxsize];
85         top = -1;
86     }
87
88
89     public void push(int data) {
90         top++;
91         s[top] = data;
92     }
93
94     public int pop() {
95         int popped = s[top];
96         top--;
97         return popped;
98     }
99
100    public int peek() {
101        return s[top];
102    }
103
104    public void print() {
105        System.out.print("Stack :");
106        for (int i = top; i > -1; i--) {
107            System.out.print(s[i] + " ");
108        }
109        System.out.println();
110    }
111 }
112 public class Reverce {
113     public static void main(String args[]) {
114         int val, ch;
115         Scanner s = new Scanner(System.in);
116         LinkListDS q = new LinkListDS();
117         Stack t=new Stack();
118         do {
119             System.out.println(
120                 "\n1.Insert Node \n2.Delete Node \n3.Pirnt List \nEnter choice :");
121             ch = s.nextInt();
122
123             switch (ch) {
124                 case 1:

```

Reverce.java

```
125         System.out.println("Enter Left Node");
126         val = s.nextInt();
127         q.insertNode(val);
128         break;
129     case 2:
130         System.out.println("Enter Right Node");
131         q.deleteNode();
132         break;
133     case 3:
134         System.out.println("LinkList");
135         q.printLink();
136         System.out.println("Rev");
137         q.sortList(t);
138         break;
139     }
140 }
141 }while(ch!=0);
142 }
143 }
144 }
```

MergeList.java

```

1 import java.util.Scanner;
2 class LinkList
3 {
4     class Node {
5         int data;
6         Node next;
7
8         Node(int d) {
9             data = d;
10            next = null;
11        }
12    }
13    Node root;
14
15    LinkList()
16    {
17        root = null;
18    }
19    public void insertNode(int e) {
20        Node n = new Node(e);
21        if (root == null) {
22            root = n;
23        } else {
24            n.next = root;
25            root = n;
26        }
27    }
28
29
30    public void deleteNode() {
31        if (root == null) {
32            System.out.println("Link List Empty");
33        } else {
34            Node t = root;
35            root = root.next;
36            System.out.println(t.data + "Remove");
37        }
38    }
39
40    public void sortLink()
41    {
42        if (root == null) {
43            System.out.println("Empty List");
44        }
45        else {
46            int temp=-1;
47            for(Node p=root;p!=null;p=p.next)
48            {
49                for(Node t=root ,t2=root.next;t!=null && t2!=null;t=t.next,t2=t2.next)
50                {
51                    if(t.data>t2.data)
52                    {
53                        temp=t.data;
54                        t.data=t2.data;
55                        t2.data=temp;
56                    }
57                }
58            }
59        }
60    }
61    }
62    System.out.println();

```

MergeList.java

```

63         }
64     }
65 }
66
67 public void printLink() {
68     if (root == null) {
69         System.out.println("Empty List");
70     } else {
71         Node t = root;
72         while (t != null) {
73
74             System.out.println(t.data);
75             t = t.next;
76         }
77     }
78 }
79 }
80
81 public Node Start() {
82     return (root == null) ? null : root;
83 }
84 }
85
86
87 public class MergeList
88 {
89
90     public static void main(String args[]) {
91         int val, ch;
92         Scanner s = new Scanner(System.in);
93         LinkedList first = new LinkedList();
94         LinkedList second = new LinkedList();
95         LinkedList third = new LinkedList();
96
97         do {
98             System.out.println("\n1.First List Node \n2.Second List Node \n3.Delete First
Node \n4.Delete Second Node \n5.Print Merged LinkedList");
99             ch = s.nextInt();
100             switch (ch) {
101                 case 1:
102                     System.out.println("Enter First List Node");
103                     val = s.nextInt();
104                     first.insertNode(val);
105                     break;
106
107                 case 2:
108                     System.out.println("Enter Second List Node");
109                     val = s.nextInt();
110                     second.insertNode(val);
111                     break;
112
113                 case 3:
114                     System.out.println("Delete First Node");
115                     first.deleteNode();
116                     break;
117                 case 4:
118                     System.out.println("Delete Second Node");
119                     second.deleteNode();
120                     break;
121                 case 5:
122                     System.out.println("Print Merge LinkedList-->");
123

```

MergeList.java

```

124      System.out.println("First LinkList\n");
125      first.printLink();
126      System.out.println("Second LinkList\n");
127      second.printLink();
128
129      LinkedList.Node a, b;
130
131      a = first.Start();
132      b = second.Start();
133
134      while (a != null && b != null) {
135          if (a.data < b.data) {
136              third.insertNode(a.data);
137              a = a.next;
138          } else {
139              third.insertNode(b.data);
140              b = b.next;
141          }
142      }
143
144      if (a != null) {
145          while (a != null) {
146              third.insertNode(a.data);
147              a = a.next;
148          }
149      }
150      if (b != null) {
151          while (b != null) {
152              third.insertNode(b.data);
153              b = b.next;
154          }
155      }
156
157      System.out.println("Print Merged LinkList\n");
158      third.printLink();
159      break;
160  }
161
162  } while (ch != 0);
163 }
164 }
165
166
167

```

SortList.java

```

1 import java.util.Scanner;
2 class List
3 {
4     class Node {
5         int data;
6         Node next;
7
8         Node(int d) {
9             data = d;
10            next = null;
11        }
12    }
13    Node root;
14
15    List()
16    {
17        root = null;
18    }
19    public void insertNode(int e) {
20        Node n = new Node(e);
21        if (root == null) {
22            root = n;
23        } else {
24            n.next = root;
25            root = n;
26        }
27    }
28
29
30    public void deleteNode() {
31        if (root == null) {
32            System.out.println("Link List Empty");
33        } else {
34            Node t = root;
35            root = root.next;
36            System.out.println(t.data + "Remove");
37        }
38    }
39
40    public void sortLink()
41    {
42        if (root == null) {
43            System.out.println("Empty List");
44        }
45        else {
46            int temp=-1;
47            for(Node p=root;p!=null;p=p.next)
48            {
49                for(Node t=root ,t2=root.next;t!=null && t2!=null;t=t.next,t2=t2.next)
50                {
51                    if(t.data>t2.data)
52                    {
53                        temp=t.data;
54                        t.data=t2.data;
55                        t2.data=temp;
56                    }
57                }
58            }
59
60        }
61    }
62    System.out.println();

```

SortList.java

```

63         }
64     }
65 }
66
67 public void printLink() {
68     if (root == null) {
69         System.out.println("Empty List");
70     } else {
71         Node t = root;
72         while (t != null) {
73
74             System.out.println(t.data);
75             t = t.next;
76         }
77     }
78 }
79 }
80
81 }
82
83
84 public class SortList
85 {
86
87     public static void main(String args[]) {
88         int val, ch;
89         Scanner s = new Scanner(System.in);
90         List q = new List();
91
92         do {
93             System.out.println("\n1.Insert Node \n2.Delete Node \n3.Print Link List \n
Enter choice :");
94             ch = s.nextInt();
95             switch (ch) {
96                 case 1:
97                     System.out.println("Enter Left Node");
98                     val = s.nextInt();
99                     q.insertNode(val);
100                     break;
101
102                 case 2:
103                     System.out.println("Enter Delete Left Node");
104                     q.deleteNode();
105                     break;
106
107                 case 3:
108                     System.out.println("Befor Sorting");
109                     q.printLink();
110                     q.sortLink();
111                     System.out.println("After Sorting");
112                     q.printLink();
113                     break;
114             }
115         } while (ch != 0);
116     }
117 }
118 }
119

```