

Single linked list

- a single link list is a list with nodes, each node has a data field and a reference to the next node.

```
1- public class Main {
2
3-     static class Node {
4         int data;
5         Node next;
6
7-     public Node(int data) {
8         this.data = data;
9         this.next = null;
10    }
11 }
12
13- public static void main(String[] args) {
14
15     Node first = new Node(10);
16     Node second = new Node(20);
17     Node third = new Node(30);
18     Node fourth = new Node(40);
19     Node fifth = new Node(50);
20
21     first.next = second;
22     second.next = third;
23     third.next = fourth;
24     fourth.next = fifth;
25
26     Node current = first;
27-     while (current != null) {
28         System.out.println(current.data);
29         current = current.next;
30     }
31 }
32 }
33 }
```

Doubly linked list

- this list is as same as the Singly but can be forward or backward Traversal

```
1- class Node {
2     int data;
3     Node next;
4     Node prev;
5
6-     public Node(int data) {
7         this.data = data;
8         this.next = null;
9         this.prev = null;
10    }
11 }
12
13- class GfG {
14
15-     static void forwardTraversal(Node head) {
16         Node curr = head;
17
18-         while (curr != null) {
19             System.out.print(curr.data + " ");
20             curr = curr.next;
21         }
22         System.out.println();
23     }
24
25     static void backwardTraversal(Node tail) {
26-         Node curr = tail;
27
28-         while (curr != null) {
29             System.out.print(curr.data + " ");
30             curr = curr.prev;
31         }
32         System.out.println();
33     }
34
35 }
36
38-     public static void main(String[] args) {
39
40         Node head = new Node(1);
41         Node second = new Node(2);
42         Node third = new Node(3);
43
44         head.next = second;
45         second.prev = head;
46         second.next = third;
47         third.prev = second;
48
49         System.out.println("Forward Traversal:");
50         forwardTraversal(head);
51
52         System.out.println("Backward Traversal:");
53         backwardTraversal(third);
54     }
55 }
```

Circular linked list

- it is as same as the singly but doesn't end with null instead it goes back to the first node

```
1- public class CircularLinkedList {
2-     private Node head;
3-
4-     private class Node {
5-         int data;
6-         Node next;
7-
8-         public Node(int data) {
9-             this.data = data;
10-            this.next = null;
11-        }
12-    }
13-
14-    public void push(int data) {
15-        Node newNode = new Node(data);
16-        if (head == null) {
17-            head = newNode;
18-            newNode.next = head;
19-        } else {
20-            Node temp = head;
21-            while (temp.next != head) {
22-                temp = temp.next;
23-            }
24-            temp.next = newNode;
25-            newNode.next = head;
26-        }
27-    }
28-
29-    public void printList() {
30-        if (head == null) {
31-            System.out.println("List is empty");
32-            return;
33-        }
34-    }
```

```
35-     Node temp = head;
36-     do {
37-         System.out.print(temp.data + " ");
38-         temp = temp.next;
39-     } while (temp != head);
40-     System.out.println();
41- }
42-
43- public static void main(String[] args) {
44-     CircularLinkedList list = new CircularLinkedList();
45-     list.push(12);
46-     list.push(15);
47-     list.push(10);
48-
49-     list.printList();
50- }
51 }
```

Doubly Circular List

- is as same as a doubly link list but can traverse backward and forward with no end, the first node storing the address to the last node, the last node storing the first node

```
1- public class DoublyCircularLinkedList {
2-     private Node head;
3-
4-     private class Node {
5-         int data;
6-         Node prev, next;
7-
8-         public Node(int data) {
9-             this.data = data;
10-            this.prev = null;
11-            this.next = null;
12-        }
13-    }
14-
15-    public void push(int data) {
16-        Node newNode = new Node(data);
17-        if (head == null) {
18-            head = newNode;
19-            newNode.next = newNode;
20-            newNode.prev = newNode;
21-        } else {
22-            Node last = head.prev;
23-            last.next = newNode;
24-            newNode.prev = last;
25-            newNode.next = head;
26-            head.prev = newNode;
27-        }
28-    }
29-
30-    public void printList() {
31-        if (head == null) {
32-            System.out.println("List is empty");
33-            return;
34-        }
35-    }
```

```
36-     Node temp = head;
37-     do {
38-         System.out.print(temp.data + " ");
39-         temp = temp.next;
40-     } while (temp != head);
41-     System.out.println();
42- }
43-
44- public static void main(String[] args) {
45-     DoublyCircularLinkedList list = new DoublyCircularLinkedList();
46-     list.push(12);
47-     list.push(15);
48-     list.push(10);
49-
50-     list.printList();
51- }
52 }
```