Simple LinkedList:

```
import java.util.Scanner;

public class LinkedList { //Singly Linked List
	static class Node {
		int data;
		Node next;

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

	Node head = null;

	public void addLeft(int data) {
		Node newNode = new Node(data);
		if (head == null) {
			head = newNode;
			return;
		}
		Node temp = head;
		head = newNode;
		head.next = temp;
	}

	public void addRight(int data) {
		Node newNode = new Node(data);
		if (head == null) {
			head = newNode;
			return;
		}
		Node curNode = head;
		while (curNode.next != null) {
			curNode = curNode.next;
		}
		curNode.next = newNode;
	}

	public void delLeft() {
		if (head == null) {
			System.out.println("Empty List");
			return;
```

```
        }
        head = head.next;
    }

    public void delRight() {
        if (head == null) {
            System.out.println("Empty List");
            return;
        }
        Node secondLastNode = head;
        Node lastNode = head.next;
        while (lastNode.next != null) {
            lastNode = lastNode.next;
            secondLastNode = secondLastNode.next;
        }
        secondLastNode.next = null;
    }

    public void search() {
        Node p = head;
        int s, position = 0;
        Scanner input = new Scanner(System.in);
        System.out.print("Which element you want to remove: ");
        s = input.nextInt();

        while (p != null) {
            if (p.data == s) {
                System.out.println("element found");
                break;
            }
            ++position;
            p = p.next;
        }
        delete(position);
    }


    public void insertNthPosition() {
        Node temp = head;
        Scanner input = new Scanner(System.in);

        System.out.print("Enter position: ");
        int position = input.nextInt();

        System.out.print("Enter data: ");
        int data = input.nextInt();

        Node newNode = new Node(data);
```

```
        if (position < 0) {
            System.out.println("Invalid Position");
        } else if (position == 1) {
            addLeft(data);
        } else {
            try {
                int i = 1;
                while (i < position - 1) {
                    temp = temp.next;
                    ++i;
                }

                newNode.next = temp.next;
                temp.next = newNode;
            } catch (Exception e) {
                addRight(data);
            }

        }

    }

    public void delete(int position) {
        Node temp = head;
        if (head == null) {
            return;
        }
        if (position == 0) {
            head = temp.next;
            return;
        }

        for (int i = 0; temp != null && i < position - 1; i++) {
            temp = temp.next;
        }

        if (temp == null || temp.next == null) {
            return;
        }
        temp.next = temp.next.next;
    }

    public void display() {
        Node curNode = head;
        if (head == null) {
            System.out.println("empty list");
        } else {
            while (curNode != null) {
                System.out.print(curNode.data + " ");
```

```
                curNode = curNode.next;
            }
            System.out.println();
        }
    }

    public static void main(String[] args) {
        LinkedList myList = new LinkedList();
        myList.addLeft(5);
        myList.addRight(6);
        myList.addLeft(4);
        myList.addRight(7);
        myList.addRight(8);
        myList.display();
        myList.insertNthPosition();
        //myList.search();
        myList.display();
        myList.delLeft();
        myList.delRight();
    }
}
```

Doubly LinkedList:

```
import java.util.Scanner;

public class DoublyLinkedList {
    static class Node {
        int data;
        Node next;
        Node previous;
        Node (int data) {
            this.data = data;
            this.next = null;
            this.previous = null;
        }
    }

    Scanner input = new Scanner(System.in);
    Node head, temp = null;

    public void create(int data) {
        Node newNode = new Node(data);
        newNode.previous = null;
        newNode.next = null;
```

```
        if (head == null) {
            head = temp = newNode;
        } else {
            temp.next = newNode;
            newNode.previous = temp;
            temp = newNode;
            temp.next = null;
        }
    }

    public void addLeft(int data) {
        Node newNode = new Node(data);
        newNode.previous = null;
        newNode.next = null;
        if (head == null) {
            head = temp = newNode;
            return;
        }
        head.previous = newNode;
        newNode.next = head;
        head = newNode;
    }

    public void addRight(int data) {
        Node newNode = new Node(data);
        newNode.previous = null;
        newNode.next = null;
        temp.next = newNode;
        newNode.previous = temp;
        temp = newNode;
    }

    public void insertAtNthPosition() {
        System.out.print("Enter Position: ");
        int position = input.nextInt();

        System.out.print("Enter Data: ");
        int data = input.nextInt();

        Node newNode = new Node(data);
        temp = head;
        int i = 1;

        if (position < 0) {
            System.out.println("Invalid Position");
        } else if (position == 1) {
            addLeft(data);
        } else {
            while (i < position - 1) {
```

```
                temp = temp.next;
                ++i;
            }
            try {
                newNode.previous = temp;
                newNode.next = temp.next;
                temp.next = newNode;
                newNode.next.previous = newNode;
            } catch (Exception e) {
                addRight(data);
            }

        }
    }

    public void deleteLeft() {
        if (head == null) {
            System.out.println("Empty List");
        } else {
            head = head.next;
            head.previous = null;
        }
    }

    public void deleteRight() {
        if (head == null) {
            System.out.println("Empty List");
        } else {
            temp.previous.next = null;
            temp = temp.previous;
        }
    }

    public void deleteAtNthPosition() {
        int i = 1;
        temp = head;
        System.out.print("Enter Position: ");
        int position = input.nextInt();

        if (head == null) {
            System.out.println("Empty List");
        } else if (position < 0) {
            System.out.println("Invalid Position");
        } else if (position == 1) {
            deleteLeft();
        } else {
            while (i < position) {
                temp = temp.next;
                ++i;
```

```
                }
                try {
                    temp.previous.next = temp.next;
                    temp.next.previous = temp.previous;
                } catch (Exception e) {
                    deleteRight();
                }

            }
        }

        public void display() {
            Node curNode = head;
            if (curNode == null) {
                System.out.println("Empty List");
            } else {
                while (curNode != null) {
                    System.out.print(curNode.data + " ");
                    curNode = curNode.next;
                }
                System.out.println();
            }
        }

        public static void main(String[] args) {
            DoublyLinkedList myList = new DoublyLinkedList();
            myList.create(2);
            myList.addLeft(1);
            myList.addRight(4);
            myList.addRight(5);
            myList.addRight(6);
            myList.addLeft(3);

            myList.display();
            myList.deleteRight();
            myList.deleteLeft();
            myList.deleteAtNthPosition();
            myList.insertAtNthPosition();

            myList.display();
        }
    }
```

Circular LinkedList:

```
import java.util.Scanner;

public class CircularLinkedList {
	static class Node {
		int data;
		Node next;

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

	Node head = null;
	Scanner input = new Scanner(System.in);


	public void insertLeft() {
		System.out.print("Enter data: ");
		int data = input.nextInt();
		Node newNode = new Node(data);
		if (head == null) {
			head = newNode;
			head.next = head;
			return;
		}
		Node temp = head;
		while (temp.next != head) {
			temp = temp.next;
		}

		temp.next = newNode;
		newNode.next = head;
		head = newNode;
	}

	public void insertRight() {
		System.out.print("Enter data: ");
		int data = input.nextInt();
		Node newNode = new Node(data);
		if (head == null) {
			head = newNode;
			head.next = head;
			return;
		}
		Node temp = head;
		while (temp.next != head) {
			temp = temp.next;
		}
```

```
        temp.next = newNode;
        newNode.next = head;
    }

    public void insertAtNthPosition() {
        System.out.print("Enter position: ");
        int position = input.nextInt();
        System.out.print("Enter data: ");
        int data = input.nextInt();
        Node newNode = new Node(data);
        if (head == null) {
            head = newNode;
            head.next = head;
            return;
        }

        if (position == -1) {
            System.out.println("Invalid Position");
        } else if (position == 1) {
            insertLeft();
        } else {
            Node temp = head;
            int i = 1;
            while (i < position - 1) {
                temp = temp.next;
                ++i;
            }
            if (i == position - 1) {
                newNode.next = temp.next;
                temp.next = newNode;
            } else {
                System.out.println("Invalid position");
            }
        }
    }

    public void deleteLeft() {
        if (head == null) {
            System.out.println("Empty list");
        } else {
            Node temp = head;
            while (temp.next != head) {
                temp = temp.next;
            }
            head = head.next;
            temp.next = head;
        }
    }
```

```
public void deleteRight() {
        Node temp = head;
        Node previous = null;
        if (head == null) {
                System.out.println("Empty List");
        } else {
                while (temp.next != head) {
                        previous = temp;
                        temp = temp.next;
                }
                previous.next = head;
                temp.next = null;
        }
}

public void deletePosition() {
        Scanner input = new Scanner(System.in);
        int position, i = 1;

        System.out.print("Enter Position: ");
        position = input.nextInt();
        if (position < 1) {
                System.out.println("Invalid Position");
        } else if (position == 1) {
                deleteLeft();
        } else {
                Node temp = head, previous = null;
                while (i < position - 1) {
                        temp = temp.next;
                        ++i;
                }
                temp.next = temp.next.next;
        }
}

public void reverseCLL() {
        if (head == null || head.next == null) {
                System.out.println("Empty List");
        } else {
                Node previous = null;
                Node current = head;
                Node nextNode;
                while (current != null) {
                        nextNode = current.next;
                        current.next = previous;
                        previous = current;
                        current = nextNode;
                }
```

```
                head = previous.next;

            }
        }

        void display() {
            if (head == null) {
                System.out.println("Empty List");
                return;
            }
            Node temp = head;
            do {
                System.out.print(temp.data + " ");
                temp = temp.next;
            } while (temp != head);
            System.out.println();
        }

        public static void main(String[] args) {
            CircularLinkedList list = new CircularLinkedList();
            list.insertLeft();
            list.insertLeft();
            list.insertLeft();
            list.insertRight();
            list.insertRight();
            list.insertRight();
            list.display();
            list.deletePosition();
//              list.deleteRight();
//              list.deleteRight();
            //list.reverseCLL();
            //list.insertAtNthPosition();
            list.display();
        }
}
```