

```
class Node {  
    private String data;  
    private Node next;  
  
    public Node(String data) {  
        this.data = data;  
    }  
  
    public void setData(String data) {  
        this.data = data;  
    }  
  
    public void setNext(Node node) {  
        this.next = node;  
    }  
  
    public String getData() {  
        return this.data;  
    }  
  
    public Node getNext() {  
        return this.next;  
    }  
}
```

```
class LinkedList {  
  
    private Node head;  
    private Node tail;  
  
    public Node getHead() {
```

```

        return this.head;
    }

    public Node getTail() {
        return this.tail;
    }

    public void addAtEnd(String data) {
        // Create a new node
        Node node = new Node(data);

        // Check if the list is empty,
        // if yes, make the node as the head and the tail
        if (this.head == null)
            this.head = this.tail = node;
        else {
            // If the list is not empty, add the element at the end
            this.tail.setNext(node);
            // Make the new node as the tail
            this.tail = node;
        }
    }

    public void addAtBeginning(String data) {
        // Create a new node
        Node node = new Node(data);

        // Check if the list is empty,
        // if yes, make the node as the head and the tail
        if (this.head == null)

```

```

        this.head = this.tail = node;
    else {
        // If the list is not empty, add the element at the beginning
        node.setNext(this.head);
        // Make the new node as the head
        this.head = node;
    }
}

```

```

public void display() {
    // Initialize temp to the head node
    Node temp = this.head;
    // Traverse the list and print data of each node
    while (temp != null) {
        System.out.println(temp.getData());
        temp = temp.getNext();
    }
}

```

```

public Node find(String data) {
    Node temp = this.head;
    // Traverse the list and return the node
    // if the data of it matches with the searched data
    while (temp != null) {
        if (temp.getData().equals(data))
            return temp;
        temp = temp.getNext();
    }
    return null;
}

```

```

public void insert(String data, String dataBefore) {
    Node node = new Node(data);
    // Check if the list is empty,
    // if yes, make the node as the head and the tail
    if (this.head == null)
        this.head = this.tail = node;
    else {
        // Find the node after which the data has to be inserted
        Node nodeBefore = this.find(dataBefore);
        if (nodeBefore != null) {
            // Insert the new node after nodeBefore
            node.setNext(nodeBefore.getNext());
            nodeBefore.setNext(node);
            // If nodeBefore is currently the tail node,
            // make the new node as the tail node
            if (nodeBefore == this.tail)
                this.tail = node;
        } else
            System.out.println("Node not found");
    }
}

```

```

public void delete(String data) {
    // Check if the list is empty,
    if (this.head == null)
        System.out.println("List is empty");
    else {
        // Find the node to be deleted
        Node node = this.find(data);
        // If the node is not found
    }
}

```

```

        if (node == null)

            System.out.println("Node not found");

        // If the node to be deleted is the head node
        else if (node == this.head) {

            this.head = this.head.getNext();

            node.setNext(null);

            // If the node to be deleted is also the tail node
            if (node == this.tail)

                tail = null;

        } else {

            // Traverse to the node present before the node to be deleted
            Node nodeBefore = null;

            Node temp = this.head;

            while (temp != null) {

                if (temp.getNext() == node) {

                    nodeBefore = temp;

                    break;

                }

                temp = temp.getNext();

            }

            // Remove the node
            nodeBefore.setNext(node.getNext());

            // If the node to be deleted is the tail node,
            // then make the previous node as the tail
            if (node == this.tail)

                this.tail = nodeBefore;

            node.setNext(null);

        }

    }

}

```

```

public static void main(String args[]) {

    LinkedList list = new LinkedList();

    list.addAtEnd("Milan");

    list.addAtEnd("Venice");

    list.addAtEnd("Munich");

    list.addAtEnd("Prague");

    list.addAtEnd("Vienna");

    list.display();

    System.out.println("-----");

    list.delete("Venice");

    list.display();

    /*
    * if(list.find("Munich")!=null) System.out.println("Node found"); else
    * System.out.println("Node not found");
    */

}
}

```

Output:

Milan

Venice

Munich

Prague

Vienna

-----

Milan

Munich

Prague

Vienna