LinkedList

1. Creation of linked list

```
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> linkedList = new LinkedListADT<>();
        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements: ");
        for (int i = 0; i < n; i++) {
            linkedList.add(sc.nextInt());
        }
        sc.close();
        linkedList.display();
    }
}
```

2. Insertion of element in linked list

```
package com.hands_on;

import user.collection.LinkedListADT;

public class Qn_2 {
    public static void main(String[] args) {
        LinkedListADT<Integer> linkedListADT = new LinkedListADT<>();
        linkedListADT.add(1);
        linkedListADT.add(2);
        linkedListADT.add(3);
        linkedListADT.add(4);
        linkedListADT.add(5);
        linkedListADT.add(6);
        linkedListADT.display();
        System.out.println("Size of the linked list is " + linkedListADT.size());
    }
}
```

3. Insert Node at Specific Position in Linked List

```
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_3 {
    public static void main(String[] args) {
        LinkedListADT<Integer> list = new LinkedListADT<>();
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the size of the list : ");
```

```java
        int n = sc.nextInt();
        System.out.println("Enter the elements of the list : ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        System.out.println("Enter the element and position for insertion :");
        int ele = sc.nextInt();
        int pos = sc.nextInt();
        list.add(ele, pos);
        System.out.println("List after insertion : ");
        list.display();
        sc.close();
    }
}
```

4. Insert in Middle of Linked List

```java
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_4 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();
        System.out.println("Enter the number of elements: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements : ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        System.out.println("Enter the element to insert in the middle: ");
        int m = sc.nextInt();
        int mid = (list.size() / 2);
        list.add(m, mid);
        list.display();
    }
}
```

5. Delete at beginning

```java
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_5 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();
        System.out.println("Enter the size of the list: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements of the list: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        sc.close();
        list.display();
        System.out.println("After deleting the first element");
        list.removeFirst();
        list.display();
    }
}
```

6. Delete a node in singly linked list

```
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_6 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();

        System.out.println("Enter the size of the list: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements of the list: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        list.display();
        System.out.println("Delete at Specific Position : " );
        System.out.print("Enter the element position: ");
        int pos = sc.nextInt();
        list.remove(pos-1);
        list.display();

    }
}
```

7 Delete At the End

```
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();
        System.out.println("Enter the size of the list: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements of the list: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        sc.close();
        list.display();
        System.out.println("The list after deleting the last node.");
        list.remove();
        list.display();
    }
}
```

8. Remove linked list elements

```
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_8 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();
        System.out.println("Enter the size of the list: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements of list: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        System.out.println("Enter the element need to be removed");
        int element = sc.nextInt();
        list.display();
        list.removeValue(element);
        System.out.println("List after removing the element");
        list.display();
    }
}
```

9. Search an element in a Linked List

```
package com.hands_on;

import user.collection.LinkedListADT;
import java.util.Scanner;

public class Qn_9 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();
        System.out.println("Enter the size of the list: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements of list: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        System.out.println("Enter the elements to be searched: ");
        int element = sc.nextInt();
        int pos = list.search(element);
        System.out.println(element+ " is found at " + pos);
    }
}
```

10. Reverse the given list

```
package com.hands_on;

import user.collection.LinkedListADT;

import java.util.Scanner;

public class Qn_10 {
    public static void main(String[] args) {
```

```java
        Scanner sc = new Scanner(System.in);
        LinkedListADT<Integer> list = new LinkedListADT<>();
        System.out.println("Enter the size of the list: ");
        int n = sc.nextInt();
        System.out.println("Enter the elements of the list: ");
        for (int i = 0; i < n; i++) {
            list.add(sc.nextInt());
        }
        sc.close();
        list.display();
        System.out.println("Reverse of linked list: ");
        list.reverse();
        list.display();
    }
}
```

LinkedListADT.java

```java
package user.collection;

class Node <T> {
    T val;
    Node<T> next;
    public Node(T val)
    {
        this.val = val;
        this.next = null;
    }
}

public class LinkedListADT<T> {
    private Node<T> head;

    public LinkedListADT() {
        head = null;
    }
    public void add(T val) {
        Node<T> newNode = new Node<>(val);
        if (head == null) {
            head = newNode;
        } else {
            Node<T> current = head;
            while (current.next != null) {
                current = current.next;
            }
            current.next = newNode;
        }
    }

    public void add( T data, int position) {
        System.out.println("Adding a node at the specified position " + position + " of the list with data " + data + "
");
        Node<T> newNode = new Node<>(data);
        Node<T> current = this.head;
        Node<T> previous = this.head;
        if (position == 1) {
            newNode.next = head;
            this.head = newNode;
            return;
        }
        for( int i = 0; current.next!= null && i < position; i++) {
            previous = current;
            current = current.next;
        }
        previous.next = newNode;
        newNode.next = current;
    }

    public boolean isEmpty() {
        return head == null;
    }

    public T getFirst() {
        return head.val;
    }

    public void display() {
        if (head != null) {
            Node<T> temp = head;
            System.out.print("Head -> ");
            while (temp != null) {
                System.out.print(temp.val + " -> ");
                temp = temp.next;
            }
            System.out.println("NULL");
        } else {
            System.out.println("LinkedListADT is empty");
        }
    }

    public T removeFirst() {
        if (isEmpty()) {
            return null;
        } else {
            Node<T> temp = head;
            head = head.next;
            return temp.val;
        }
    }
    public void remove() {
        if (isEmpty()) {
            System.out.println("LinkedListADT is empty");
        } else {
            Node<T> current = this.head;
            while (current.next.next != null) {
                current = current.next;
            }
            current.next = null;
        }
    }

    public T getLast() {
        if (isEmpty()) {
            return null;
        } else {
            Node<T> temp = head;
            while (temp.next != null) {
                temp = temp.next;
            }
            return temp.val;
        }
    }

    public int size() {
        if (isEmpty()) {
            return 0;
        } else {
            int cnt = 0;
            Node<T> temp = head;
            while (temp.next != null) {
                temp = temp.next;
                cnt++;
            }
            return cnt+1;
        }
    }

    public String getSimpleClassName() {
        return "UserDefined: LinkedListADT";
    }

    public void remove(int position)
    {
        System.out.println("Deleting a node from the specified position " + position +  "
");
        if (head == null) {
            System.out.println("The given list is empty.
");
        }
        else if(position == 0) {
            head = head.next;
        }
        else {
            Node<T> current = head;
            Node<T> prev = head;
            for(int i = 0; current.next != null && i < position; i++) {
                prev = current;
                current = current.next;
            }

            prev.next = current.next;
        }
    }
    public boolean contains(T val) {
        Node<T> temp = head;
        while (temp != null) {
            if (temp.val == val) {
                return true;
            }
            temp = temp.next;
        }
        return false;
    }
    public void removeValue(T val) {
        Node<T> current = head;
        Node<T> prev = head;
        if (head.val == val) {
            head = head.next;
        } else {
            while(current.next != null) {
                if(current.val == val) {
                    prev.next = current.next;
                    current.next = current.next.next;
                }
                if (current.next!= null) {
                    prev = current;
                    current = current.next ;
                }
            }
        }
        if (contains(val)) {
            removeValue(val);
        }
    }
```

```java
    public int search(T val) {
        if (contains(val)) {
            int cnt = 0;
            Node<T> temp = head;
            while(temp != null) {
                if (temp.val == val) {
                    return ++cnt;
                }
                cnt++;
                temp = temp.next;
            }
        }
        return -1;
    }

    public void reverse(){
        Node<T> prev = null;
        Node<T> current = head;
        Node<T> next = null;
        while (current != null) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        head = prev;
    }

    public void merge(LinkedListADT<T> ele){
        Node<T> temp = head;
        while(temp.next != null) {
            temp = temp.next;
        }
        temp.next = ele.head;
    }
}
```