Made by :- Amrit Agrawal

Singly Linked List :-

```
1. Add
2. Add First
3. Delete First
4. Get First
5. Get Last
6. Display
7. Add Last
8. Delete Last
9. Size
10. Get Node Value
11. Delete In Specified Index
12. Add In Specified Index
13. Display Reverse
14. Reverse LinkedList
15. Detect loop in a linked list
0 to exit
```

```
import java.util.HashSet;
import java.util.Scanner;
public class singlelinklist
{
	private class Node
	{
		Object Data;
		Node next;
		Node(Object Data)
		{
			this.Data=Data;
		}
	}
	static Node head;
	public boolean add(Object data)//1
	{
		Node n=new Node(data);
		if(head==null)
		{
			head=n;
			return true;
		}
		Node t=head;
		while (t.next!=null)
		{
			t=t.next;
		}
		t.next=n;
		return true;
	}
	public void addfirst(Object data)//2
	{
		Node n=new Node(data);
		n.next=head;
		head=n;
	}
	public Object deletefirst() //3
	{
		if(head==null)
		{
			System.out.println("list is empty");
			return null;
		}
		Object data=head.Data;
		head=head.next;
		return data;
	}
	public Object getFirst()//4
	{
		if(head==null)
		{
			System.out.println("list is empty");
			return null;
		}
		return head.Data;
	}
	public Object getLast()//5
	{
		if(head==null)
		{
			System.out.println("list is empty");
			return null;
		}
		Node t=head;
		while(t.next!=null)
		{
			t=t.next;
		}
		return t.Data;
	}
	public void display() //6
	{
		Node t=head;
		while (t!=null)
		{
			System.out.print(t.Data);
			if(t.next!=null)
				System.out.print("--->");
			t=t.next;
		}
	}
	public void addAtEnd(Object data) //7
	{
		Node node = new Node(data);

		if (head == null)
		{
			head = new Node(data);
			return;
		}
		node.next = null;
		Node last = head;
		while (last.next != null)
			last = last.next;

		last.next = node;
		return;
	}
	public Object deletelast()//8
	{
		if(head==null)
		{
			System.out.println("list is empty");
			return null;
		}
		Node t=head;
		Node t1=head;
		while (t.next!=null)
		{
			t1=t;
			t=t.next;
		}
		if(t1.next==null)
			head=null;
		else
			t1.next=null;
		return t.Data;
	}
	public int size() //9
	{
		int count =0;
		Node t=head;
		while (t!=null)
		{
			t=t.next;
			count++;
		}
		return count;
	}
	public Object getNode(Node head,int n)//10
	{
		int nodes = 0;
		Node i = head;
		while(i != null)
		{
			i = i.next;
			nodes++;
		}
		nodes -= n;
		while(--nodes > 0)
		{
```

```
                head = head.next;
            }
            return head.Data;
        }
        public Object deleteSpNode(int in)//11
        {
            if(in<0||in>size())
            {
                System.out.println("Index not Range");
                return null;
            }
            if(in==0)
            {
                Object data=head.Data;
                head=head.next;
                return data;
            }
            Node t=head;
            while (in>1)
            {
                t=t.next;
                in--;
            }
            Object data=t.next.Data;
            t.next=t.next.next;
            return data;
        }
        public void addspeindex(Object data,int in)//12
        {
            if(in<0||in>size())
            {
                System.out.println("Index not in the Range");
                return;
            }
            Node n=new Node(data);
            if(in==0)
            {
                addfirst(data);
                return;
            }
            Node t=head;
            while (in>1)
            {
                t=t.next;
                in--;
            }
            n.next=t.next;
            t.next=n;
        }
        public void displayRev(Node n)//13
        {
            if(n.next!=null)
                displayRev(n.next);
            if(n.next!=null)
                System.out.print("<===");
```

```
		System.out.print(n.Data);
	}
	public Object reverseLinklist(Node n)//important//14
	{
		Node prev=null;
		Node curr=n;
		Node next=null;
		while (curr!=null)
		{
			next=curr.next;
			curr.next=prev;
			prev=curr;
			curr=next;
		}
		n = prev;
		return n;
	}
	public boolean detectLoop(Node h) //15//important
	{
		HashSet<Node> s = new HashSet<Node>();
		while (h != null)
		{
			if (s.contains(h))
				return true;
			s.add(h);
			h = h.next;
		}
		return false;
	}
	@Override
	public String toString()
	{
		String st="[";
		Node t=head;
		while (t!=null)
		{
			st=st+t.Data;
			if (t.next!=null)
			{
				st=st+"=>";
			}
			t=t.next;
		}
		return st+"]";
	}
	public static void main(String[] args)
	{
		System.out.println("Welcome to The Single linked List\n");
		System.out.println("Enter your choices ");
		System.out.println("1. Add");
		System.out.println("2. Add First");
		System.out.println("3. Delete First");
		System.out.println("4. Get First");
		System.out.println("5. Get Last");
		System.out.println("6. Display");
		System.out.println("7. Add Last");
		System.out.println("8. Delete Last");
		System.out.println("9. Size");
		System.out.println("10. Get Node Value");
		System.out.println("11. Delete In Specified Index");
		System.out.println("12. Add In Specified Index");
		System.out.println("13. Display Reverse");
		System.out.println("14. Reverse LinkedList");
		System.out.println("15. Detect loop in a linked list");
		System.out.println("0 to exit");

		singlelinklist s1=new singlelinklist();
		Scanner sc=new Scanner(System.in);
		int choice=0;
		do {
			System.out.println();
			System.out.println("Please enter your choice::");
			choice=sc.nextInt();
			switch (choice)
			{
			case 1:
				System.out.println("Enter the number of Data you want to 
Add");

				int value=sc.nextInt();
				System.out.println("Enter the data you want to add");
				for (int i = 0; i <value; i++)
				{
					Object d=sc.next();
					s1.add(d);
				}
				System.out.println("Data is Added \n");
				System.out.println("Press 6 For Display the Data");
				break;
			case 2:
				System.out.println("Enter the data you want to Add");
				Object o=sc.next();
				s1.addfirst(o);
				System.out.println(o+" is Added on Node first");
				break;
			case 3:
				System.out.println("First Node "+s1.deletefirst()+" is 
Deleted ");

				break;
			case 4:
				System.out.println("The First Node is ==>"+s1.getFirst());
				break;
			case 5:
				System.out.println("The last Node is "+s1.getLast());
				break;
			case 6:
				s1.display();
				break;
			case 7:
				System.out.println("Enter the data you want to Add");
				Object obj4=sc.next();
```

```
                    s1.addAtEnd(obj4);
                    System.out.println("The Node is Added at the Last \n");
                    s1.display();
                    break;
                case 8:
                    s1.deletelast();
                    System.out.println("The last Node is Deleted");
                    break;
                case 9:
                    System.out.println("The Size of Linked List is 
"+s1.size());
                    break;
                case 10:
                    System.out.println("Enter the Node no");
                    int n3=sc.nextInt();
                    System.out.println("The value in "+n3+" Node is 
"+s1.getNode(head,n3));
                    break;
                case 11:
                    System.out.println("Enter the index");
                    int in=sc.nextInt();
                    s1.deleteSpNode(in);
                    if(in<s1.size())
                    System.out.println("The "+in+" Node is Deleted");
                    break;
                case 12:
                    System.out.println("Enter the data and then index ");
                    Object obj=sc.next();
                    int n2=sc.nextInt();
                    s1.addspeindex(obj, n2);
                    if(n2<s1.size())
                        System.out.println("Data is succesfully Added in 
"+n2+" Postion");
                    break;
                case 13:
                    System.out.println("Reverse Display of Linked List \n");
                    s1.displayRev(head);
                    break;
                case 14:
                    head=(Node)s1.reverseLinklist(head);
                    System.out.println("Linked List is Reverse \n");
                    s1.display();
                    break;
                case 15:
                    //s1.head.next.next.next.next = s1.head; //This is to 
creating the loop
                    if (s1.detectLoop(head))
                        System.out.println("Loop Detect");
                    else
                        System.out.println("No Loop");
                    break;
                case 0:
                    System.out.println("Thank you");
                    break;
                default:System.out.println("No such choice available..");
```

```
                    break;
                    }
                    if(choice==0)
                    {
                        sc.close();
                        break;
                    }
                } while (true);
                sc.close();
        }
}
```