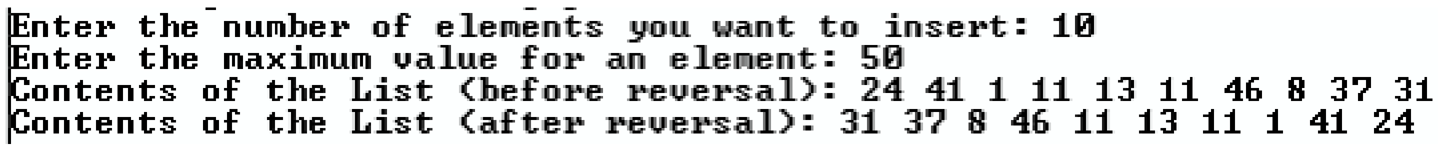
```java
import java.util.*;
// reversing a singly linked list

class Node{

    private int data;
    private Node nextNodePtr;


    public Node(){}

    public void setData(int d){
            data = d;
    }

    public  int getData(){
            return data;
    }

    public  void setNextNodePtr(Node nodePtr){
            nextNodePtr = nodePtr;
    }

    public  Node getNextNodePtr(){
        return nextNodePtr;
    }

}

class List{

    private Node headPtr;


    public  List(){
        headPtr = new Node();
        headPtr.setNextNodePtr(null);
    }


    public Node getHeadPtr(){
        return headPtr;
    }

    public  boolean isEmpty(){

        if (headPtr.getNextNodePtr() == null)
            return true;

        return false;
    }


    public  void insert(int data){

            Node currentNodePtr = headPtr.getNextNodePtr();
            Node prevNodePtr = headPtr;

            while (currentNodePtr != null){
                prevNodePtr = currentNodePtr;
                currentNodePtr = currentNodePtr.getNextNodePtr();
            }

            Node newNodePtr = new Node();
            newNodePtr.setData(data);
            newNodePtr.setNextNodePtr(null);
            prevNodePtr.setNextNodePtr(newNodePtr);

    }

    public  void insertAtIndex(int insertIndex, int data){

            Node currentNodePtr = headPtr.getNextNodePtr();
            Node prevNodePtr = headPtr;

            int index = 0;

            while (currentNodePtr != null){

                if (index == insertIndex)
                    break;

                prevNodePtr = currentNodePtr;
                currentNodePtr = currentNodePtr.getNextNodePtr();
                index++;
            }

            Node newNodePtr = new Node();
            newNodePtr.setData(data);
            newNodePtr.setNextNodePtr(currentNodePtr);
            prevNodePtr.setNextNodePtr(newNodePtr);

    }



    public  void IterativePrint(){

            Node currentNodePtr = headPtr.getNextNodePtr();

            while (currentNodePtr != null){
                System.out.print(currentNodePtr.getData()+" ");
                currentNodePtr = currentNodePtr.getNextNodePtr();
            }

            System.out.println();

    }


    public  void reverseList(){

            Node currentNodePtr = headPtr.getNextNodePtr();
            Node prevNodePtr = null;
            Node nextNodePtr = currentNodePtr;

            while (currentNodePtr != null){

                nextNodePtr = currentNodePtr.getNextNodePtr(); // Step 1
                currentNodePtr.setNextNodePtr(prevNodePtr); // Step 2
                prevNodePtr = currentNodePtr;  // Step 3
                currentNodePtr = nextNodePtr;  // Step 4

            }

            headPtr.setNextNodePtr(prevNodePtr);

        }
```

```java



}

class ReverseSinglyLinkedList{

    public static void main(String[] args){

    Scanner input = new Scanner(System.in);

    int listSize;
    System.out.print("Enter the number of elements you want to insert: ");
    listSize = input.nextInt();

    List integerList = new List(); // Create an empty list

    int maxValue;
    System.out.print("Enter the maximum value for an element: ");
    maxValue = input.nextInt();

    Random randGen = new Random(System.currentTimeMillis());

    for (int i = 0; i < listSize; i++){

        int value = randGen.nextInt(maxValue);

        integerList.insertAtIndex(i, value);
    }

    System.out.print("Contents of the List (before reversal): ");
    integerList.IterativePrint();

    integerList.reverseList();

    System.out.print("Contents of the List (after reversal): ");
    integerList.IterativePrint();

    }

}
```



```
Enter the number of elements you want to insert: 10
Enter the maximum value for an element: 50
Contents of the List (before reversal): 24 41 1 11 13 11 46 8 37 31
Contents of the List (after reversal): 31 37 8 46 11 13 11 1 41 24
```