```java
import java.util.*;

// implementing the dynamic List ADT using 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();
            }
```

```
65
66              Node newNodePtr = new Node();
67              newNodePtr.setData(data);
68              newNodePtr.setNextNodePtr(null);
69              prevNodePtr.setNextNodePtr(newNodePtr);
70
71      }
72
73      public  void insertAtIndex(int insertIndex, int data){
74
75              Node currentNodePtr = headPtr.getNextNodePtr();
76              Node prevNodePtr = headPtr;
77
78              int index = 0;
79
80              while (currentNodePtr != null){
81
82                  if (index == insertIndex)
83                      break;
84
85                  prevNodePtr = currentNodePtr;
86                  currentNodePtr = currentNodePtr.getNextNodePtr();
87                  index++;
88              }
89
90              Node newNodePtr = new Node();
91              newNodePtr.setData(data);
92              newNodePtr.setNextNodePtr(currentNodePtr);
93              prevNodePtr.setNextNodePtr(newNodePtr);
94
95      }
96
97
98      public  int read(int readIndex){
99
100             Node currentNodePtr = headPtr.getNextNodePtr();
101             Node prevNodePtr = headPtr;
102             int index = 0;
103
104             while (currentNodePtr != null){
105
106                 if (index == readIndex)
107                     return currentNodePtr.getData();
108
109                 prevNodePtr = currentNodePtr;
110                 currentNodePtr = currentNodePtr.getNextNodePtr();
111
112                 index++;
113
114             }
115
116         return -1; // an invalid value indicating
117                    // index is out of range
118
119     }
120
121     public  void modifyElement(int modifyIndex, int data){
122
123             Node currentNodePtr = headPtr.getNextNodePtr();
124             Node prevNodePtr = headPtr;
125             int index = 0;
126
127             while (currentNodePtr != null){
128
```

```
                if (index == modifyIndex){
                    currentNodePtr.setData(data);
                    return;
                }

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

                index++;
            }


    }


    public  void deleteElement(int deleteIndex){


            Node currentNodePtr = headPtr.getNextNodePtr();
            Node prevNodePtr = headPtr;
            Node nextNodePtr = headPtr;
            int index = 0;

            while (currentNodePtr != null){

                if (index == deleteIndex){
                    nextNodePtr = currentNodePtr.getNextNodePtr();
                    break;
                }

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

                index++;
            }

            prevNodePtr.setNextNodePtr(nextNodePtr);

    }



    public  void IterativePrint(){

            Node currentNodePtr = headPtr.getNextNodePtr();

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

            System.out.println();

    }


}

class SinglyLinkedList{

    public static void main(String[] args){

    Scanner input = new Scanner(System.in);

```

```
193         int listSize;
194 
195         System.out.print("Enter the number of elements you want to insert: ");
196         listSize = input.nextInt();
197 
198         List integerList = new List(); // Create an empty list
199 
200         for (int i = 0; i < listSize; i++){
201 
202             int value;
203             System.out.print("Enter element # "+i+" : ");
204             value = input.nextInt();
205 
206             integerList.insertAtIndex(i, value);
207         }
208 
209         System.out.print("Contents of the List: ");
210         integerList.IterativePrint();
211 
212         // to read an element at a particular index (before delete)
213 
214         int readIndex;
215         System.out.print("Enter an index to read (before delete): ");
216         readIndex = input.nextInt();
217         System.out.println("Value at " +readIndex +" is: "+integerList.read(readIndex) );
218 
219         // to delete an element at a particular index
220 
221         int deleteIndex;
222         System.out.print("Enter an index to delete: ");
223         deleteIndex = input.nextInt();
224         integerList.deleteElement(deleteIndex);
225 
226         System.out.print("Contents of the List: ");
227         integerList.IterativePrint();
228 
229 
230         // to read an element at a particular index (after delete)
231 
232         System.out.print("Enter an index to read (after delete): ");
233         readIndex = input.nextInt();
234         System.out.println("Value at " + readIndex +" is: " +integerList.read(readIndex) );
235 
236 
237         // to insert an element at a particular index
238         int insertIndex, insertValue;
239         System.out.print("Enter an index to insert: ");
240         insertIndex = input.nextInt();
241         System.out.print("Enter a value to insert: ");
242         insertValue = input.nextInt();
243         integerList.insertAtIndex(insertIndex, insertValue);
244 
245         System.out.print("Contents of the List: ");
246         integerList.IterativePrint();
247 
248 
249         // to read an element at a particular index (after insert)
250 
251         System.out.print("Enter an index to read (after insert): ");
252         readIndex = input.nextInt();
253         System.out.println("Value at " +readIndex +" is: " +integerList.read(readIndex) );
254 
255         // to insert at the end of the list
256         System.out.print("Enter the element you want to insert at the end of the list: ");
```

```
257         insertValue = input.nextInt();
258         integerList.insert(insertValue);
259 
260         System.out.print("Contents of the List: ");
261         integerList.IterativePrint();
262 
263 
264         }
265 
266   }
```

```
Enter the number of elements you want to insert: 5
Enter element # 0 : 78
Enter element # 1 : 96
Enter element # 2 : 23
Enter element # 3 : 12
Enter element # 4 : 44
Contents of the List: 78 96 23 12 44
Enter an index to read (before delete): 3
Value at 3 is: 12
Enter an index to delete: 2
Contents of the List: 78 96 12 44
Enter an index to read (after delete): 3
Value at 3 is: 44
Enter an index to insert: 1
Enter a value to insert: 55
Contents of the List: 78 55 96 12 44
Enter an index to read (after insert): 2
Value at 2 is: 96
Enter the element you want to insert at the end of the list: 49
Contents of the List: 78 55 96 12 44 49
```