Linked list implementation

```
/* void addFirst(Object o)
 * Inserts the given element at the beginning of the list. */
public void addFirst(Object element){
  Node newNode;
  newNode=new Node(element);
  if (Size==0) {// Empty List
    Front=Back=newNode;
  }
  else{
    newNode.next=Front;
    Front=newNode;
  }
   Size++;// update Size
}
```

Linked list implementation

```
/*Object getFirst()
  Returns the first element in the list. */

public Object getFirst() {
   if (Size == 0)
     return null;
   else
      return Front.element;
}
```

Linked list implementation

```
/*void addLast(Object o)
  Appends the given element to the end of the list.*/
public void addLast (Object element){
   Node newNode;
   newNode=new Node(element);
   if (Size==0){// Empty List
      Front=Back=newNode;
   }
   else{
     Back.next=newNode;
     Back=newNode;      // Or Back=Back.next;
}
    Size++;// update Size
}
```

Linked list implementation

```
/* Object getLast()
   Returns the last element in the list.*/

public Object getLast(){
if (Size==0)
    return null;
else
    return Back.element;
}
```

Linked list implementation

```
/*Object get(int index)
  Returns the element at the specified position in the list.*/
 public Object get(int index){
    if (Size==0) return null; //empty list
    else if (index==0) return getFirst(); //first element
    else if (index==Size-1)return getLast();  //last element
    else if (index >0 && index<Size-1){
        Node current=Front;
        for (int i=0;i<index;i++)
           current=current.next;

        return current.element;
     }
     else
       return null; //out of boundary
 }
```

Linked list implementation

```
/*int size()
Returns the number of elements in the list.*/

 public int size(){
   return Size;
}


/*void add(Object o)
Appends the specified element to the end of the list*/
public void add(Object element)
{
     add(size(), element);
}
```

Linked list implementation

```
/*void add(int index, Object element) Inserts the specified
element  at the specified position index in this list.*/
  public void add(int index, Object element) {
    if (index==0) addFirst(element);
    else if (index>=size())addLast(element);
    else{
         Node newNode= new Node(element);
         Node current=Front;  //used to advance to proper position
         for (int i=0;i<index-1;i++)
           current=current.next;

         newNode.next=current.next;
         current.next=newNode;
         Size++;// update size
        }

  }
```

Linked list implementation

```
/*boolean removeFirst()
  Removes  the first element from the list.*/
 public boolean removeFirst(){
   if (Size==0)
      return false; //empty list
   else if (Size==1) //one element inside list
      Front=Back=null;
   else
      Front=Front.next;

   Size--; //update size
   return true;
 }
```

Linked list implementation

```
/*boolean removeLast()
  Removes  the last element from this list.*/
 public boolean removeLast(){
   if (Size==0)
        return false; //empty list
   else if (Size==1) // one element inside the list
        Front=Back=null;
   else{
       Node current= Front;
       for (int i=0;i<Size-2;i++)
          current=current.next;

      current.next=null;
      Back=current;
  }
   Size--; //update size
   return true;
}
```