The LinkedList class provides unique name method to get(), remove(), and insert() as element at the beginning and end of the list; these operation allowed LinkedList to be used as a Stack, Queue, DQueue. The implementation is not also synchronized.

LinkedList have extra 6 methods in comparison of ArrayList and Vector.

- public void addFirst(Object o);
- public void addLast(Object o);
- public Object getFirst();
- public Object getLast();
- public void removeFirst();
- public void removeLast();

Program: Perfroming operation with pre-defined class.

```
import java.util.LinkedList;

    class LinkListExample{

        public static void main(String...a){

            //LinkedList declaration
            LinkedList<String> linkList = new LinkedList<>();

            // Add(String elements) is used for adding the elements to LinkedList
            linkList.add("Item 1");
            linkList.add("Item 2");
            linkList.add("Item 3");
            linkList.add("Item 4");
            linkList.add("Item 6");

            // Display the LinkedList elements
            System.out.println("Actual Elements: " + linkList);

            //Let's add first elements and last elements
            linkList.addFirst("First Item");
            linkList.addLast("Last Item");

            System.out.println("After Adding: " + linkList);

            // Now let's get and set values in LinkedList.
            Object firstItem = linkList.get(0);
            System.out.println("First Element is: " + firstItem);

            linkList.set(0, "Item Changed");
            firstItem = linkList.get(0);
            System.out.println("First Element After Changes: " + firstItem);


            // Now' let's remove the first and last element.
            linkList.remove();    //By default remove the first element.
            linkList.removeLast();
            System.out.println("After Deletion: " + linkList);

            // Now let's add the element to a position and remove from position.
            linkList.add(4, "Item 5");
            System.out.println("Added Element at a position: " + linkList);
        linkList.remove(2);
    System.out.println("After removing from position: " + linkList);

    System.out.println("First element via getFirst(): " + linkList.getFirst());
    System.out.println("First element via getLast(): " + linkList.getLast());

    }
}
```

// Program 2: LinkList operation on user-defined class.

```
    import java.util.LinkedList;
    import java.util.Iterator;

    public class LinkedListExample{

        public static void main(String...a){

            // LinkedList declaration
            LinkedList<Emp> linkList = new LinkedList<>();

            // Let's add the elements in LinkedList
            linkList.add(new Emp(1001, "Vineet"));
            linkList.add(new Emp(1002, "Prem"));
            linkList.add(new Emp(1003, "Javed"));
            linkList.add(new Emp(1004, "Sayeed"));
            linkList.add(new Emp(1005, "Parul"));
            linkList.add(new Emp(1006, "Lovely"));

            // Display the LinkedList elements
            Iterator<Emp> itr = linkList.iterator();
            while(itr.hasNext()){
                Emp e = itr.next();
                System.out.println(e.id + ", " + e.name);
            }

            // Now Let's add elements at first and last.
            linkList.addFirst(new Emp(1000, "Kotlin"));
            linkList.addLast(new Emp(1007, "Papaya"));

            System.out.println("\nAfter Adding: ");
            for(Emp e: linkList){
                System.out.println(e.id + ", " + e.name);
            }

            // This is how i get and set value.
            linkList.set(0, new Emp(1000, "Aniqa"));
            Emp e1 = (Emp) linkList.get(0);
            System.out.println("\nChanges via set(0): " + e1.id + ", " + e1.name);

            // Now let's remove first and last Element.
            linkList.remove();
            linkList.removeLast();

            System.out.println("\nAfter removing first and las elements:");
            itr = linkList.iterator();
            while(itr.hasNext()){
```

```
            Emp e = itr.next();
            System.out.println(e.id + ", " + e.name);
        }

        // Add to position and remove from a position.
        linkList.add(0, new Emp(999, "Harsh"));
        linkList.remove(linkList.size()-1);

        System.out.println("\nAfter adding via position: ");
        for(Emp e: linkList){
            System.out.println(e.id + ", " + e.name);
        }

        Emp fe = (Emp) linkList.getFirst();
        System.out.println("\nFirst element via getFirst(): " + fe.id + ", " +
            fe.name);

        fe = (Emp) linkList.getLast();
        System.out.println("\nFirst element via getLast(): " + fe.id + ", " +
            fe.name);

    }
}

// User Defined Class
class Emp{
    int id;
    String name;

    //Constructor
    Emp(int id, String name){
        this.id = id;
        this.name = name;
    }
}
```

Program 3. Access LinkedList Via Loops

```
import java.util.LinkedList;
import java.util.Iterator;

public class LinkListLoop{

    public static void main(String...a){

        LinkedList<String> linkedList = new LinkedList<>();
        linkedList.add("Apple");
        linkedList.add("Orange");
        linkedList.add("Mango");
        linkedList.add("Papaya");

        // via for loop
        System.out.println("\nVia Loop:");
        for(int i=0; i< linkedList.size(); i++){
            System.out.println(linkedList.get(i));
        }
        // Via advance for loop
        System.out.println("\nVia Advance Loop:");
        for(String s1: linkedList){
            System.out.println(s1);
        }

        // Via while loop
        int num = 0;
        System.out.println("\nVia While Loop:");
        while(linkedList.size() > num){
            System.out.println(linkedList.get(num));
            ++num;
        }

        // Via Iterator
        System.out.println("\nVia Iterator:");
        Iterator itr = linkedList.iterator();
        while(itr.hasNext()){
            System.out.println(itr.next());
        }
    }
}
```

Program 4: Ways to remove elements from linkedList. All remove() methods of LinkedList.

```
import java.util.LinkedList;

public class AllRemoveMethod{

    public static void main(String...args){
        LinkedList<String> linkedList = new LinkedList<> ();
        linkedList.add("String One");
        linkedList.add("String Two");
        linkedList.add("String Three");
        linkedList.add("String Four");
        linkedList.add("String Five");
        linkedList.add("String Six");
        linkedList.add("String Seven");
        linkedList.add("String Eight");
        linkedList.add("String Eight");
        linkedList.add("String Nine");

        System.out.println("Actual List: " + linkedList);

        System.out.println("\n remove(): " + linkedList.remove());
        System.out.println(" Changed List: " + linkedList);

        System.out.println("\n remove(index): " + linkedList.remove(2));
        System.out.println(" Changed List: " + linkedList);

        System.out.println("\n remove(Object): " + linkedList.remove("String
Six"));

        System.out.println(" Changed List: " + linkedList);
        System.out.println("\n removeFirst(): " + linkedList.removeFirst());
        System.out.println(" Changed List: " + linkedList);

        System.out.println("\n removeLast(): " + linkedList.removeLast());
        System.out.println(" Changed List: " + linkedList);

        System.out.println("\n removeFirstOccurrence(): " +
    linkedList.removeFirstOccurrence("String Eight"));

        System.out.println(" Changed List: " + linkedList);

        System.out.println("\n removeLastOccurrence(): " +
    linkedList.removeLastOccurrence("String Three"));

        System.out.println(" Changed List: " + linkedList);
      }
}
```

Program 5: Ways to access last elements of LinkedList

```
    import java.util.LinkedList;

    public class MyLastElement{

      public static void main(String...a){

        LinkedList<String> linkedList = new LinkedList<>();
        linkedList.add("String One");
        linkedList.add("String Two");
        linkedList.add("String Three");
        linkedList.add("String Four");
        linkedList.add("String Five");

        System.out.println("Actual List: " + linkedList);

        /* getLast(): returns the last element in the list, but if list is
    empty then it returns exception.*/

        System.out.println("\n via getLast(): " + linkedList.getLast());

        /* peekLast(): returns the last element in the list, but if list if
    empty then it returns null.*/

        System.out.println("\n via peekLast(): " + linkedList.peekLast());
      }
}
```

Program 6: Reverse the iterator in LinkedList

```
    import java.util.LinkedList;
    import java.util.Iterator;

    class ReverseIterator{
      public static void main(String...a){
            LinkedList<String> name = new LinkedList<>();
            name.add("String One");
            name.add("String Two");
            name.add("String Three");
            name.add("String Four");
            name.add("String Five");

            System.out.println("Actual List: " + name);

            // Now let's iterate in descending order.
            System.out.println("\nAfter Reverse Iteration: ");
            Iterator itr = name.descendingIterator();
            while(itr.hasNext()){
                System.out.println(itr.next());
            }
        }
}
```

Program 7: Push (add) Pop (remove) elements

```
    import java.util.LinkedList;

    public class MyPushPop{

        public static void main(String...a){

            LinkedList<String> name = new LinkedList<>();
            name.add("String One");
            name.add("String Two");
            name.add("String Three");
            name.add("String Four");
            name.add("String Five");

            System.out.println("Actual List: " + name);

            // Now let's push the element
            name.push("Push Element");
            System.out.println("Push Changes: " + name);

            // Now let's pop the element.
            name.pop();
            System.out.println("Pop Changes: " + name);
        }
}
```

Program 8: Create My Own LinkedList: MyLinkedList.java

```java
 class Link{
     int iData;
     public Link next;

     // Constructor
     public Link(int iData){
         this.iData = iData;
     }

     @Override
     public String toString(){
         return "{ " + iData + " }";
     }
 }


//LinkedList

 class LinkedList{

     private Link first;

     //Constructor
     public LinkedList(){
         first = null;
     }

     public boolean isEmpty(){
         return (first == null);   // returns true if first is null.
     }

     public void insertFirst(int id){

         Link newLink = new Link(id);
         newLink.next = first;
         first = newLink;
     }

     public Link deleteFirst(){
         Link temp = first;
         first = first.next;
         return temp;
     }

     @Override
     public String toString(){
         String str = "";
         Link current = first;
         while(current != null){
             str+= current.toString();
             current = current.next;
         }

         return str;
     }
 }


 // Class that using all those classes.
 public class MyLinkedList{

     public static void main(String...a){
         LinkedList theList = new LinkedList();
         theList.insertFirst(22);
         theList.insertFirst(44);
         theList.insertFirst(66);
         theList.insertFirst(88);

         System.out.println(theList);

         // Let's delete the first element...
         System.out.println("Deleting firstElement..." + theList.deleteFirst());
         System.out.println("After Deletion: " + theList);

         // Now Let's delete the all elements
         while(!theList.isEmpty()){
             Link aLink = theList.deleteFirst();
             System.out.println("Deleted item is: " + aLink);
         }

         System.out.println("");
         System.out.println("Elements rest are: " + theList);
     }
 }
```

Program: Finding Linked List Element

```
class Link{
    public int iData;
    public Link next;

    public Link(int id){
        iData=id;
    }

    public String toString(){
        return "{"+iData+"}";
    }
}

class LinkList{

    private Link first;

    public LinkList(){
        first=null;
    }

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

    public void insertFirst(int id){
        Link newLink=new Link(id);
        newLink.next=first;
        first=newLink;
    }
    public Link delete(int key){
        Link current=first;
        Link previous=first;

        while(current.iData!=key){
            if(current.next==null)
                return null;
            else
            {
                previous=current;
                current=current.next;
            }
        }

        if(current==first)first=first.next;
        else previous.next=current.next;
        return current;
    }

    public Link find(int key){
        Link current=first;
        while(current.iData!=key){
            if(current.next==null)
                return null;
            else
                current=current.next;
        }
        return current;
    }

    public String toString(){
        String str="";
        Link current=first;
        while(current!=null){
            str+=current.toString();
            current=current.next;
        }
        return str;
    }
}

//class
public class FindingLinkedList{

    public static void main(String arg[]){

        LinkList theList=new LinkList();
        theList.insertFirst(22);
        theList.insertFirst(44);
        theList.insertFirst(66);
        theList.insertFirst(88);

        System.out.println("Actual List: " + theList);

        // Let's find the element
        Link In = theList.find(44);
        System.out.println("Element Available: " + In);

        Link aLink = theList.delete(44);
        System.out.println("After Deletion: " + theList);

    }
}
```