[java集合框架之LinkedList 深度解析（一）](http://www.spring4all.com/article/1067)

**摘要：**

**一、**

LinkedList 是一个继承于AbstractSequentialList的双向链表。它也可以被当作堆栈、队列或双端队列进行操作。  
LinkedList 实现 List接口，能对它进行队列操作。  
LinkedList 实现 Deque接口，即能将LinkedList当作双端队列使用。  
LinkedList 实现了Cloneable接口，即覆盖了函数clone()，能克隆。  
LinkedList 实现java.io.Serializable接口，这意味着LinkedList支持序列化，能通过序列化去传输。  
LinkedList 是非同步的。

**二、**

LinkedList 和 ArrayList 一样，都实现了 List 接口，但其内部的数据结构有本质的不同。LinkedList 是基于链表实现的（通过名字也能区分开来），所以它的插入和删除操作比 ArrayList 更加高效。但也是由于其为基于链表的，所以随机访问的效率要比 ArrayList 差。

**三、**

当你需要使用栈或者队列时，可以考虑使用LinkedList，一方面是因为Java官方已经声明不建议使用Stack类，更遗憾的是，Java里根本没有一个叫做Queue的类（它是个接口名字）。关于栈或队列，现在的首选是ArrayDeque，它有着比LinkedList（当作栈或队列使用时）有着更好的性能。

看一下 LinkedList 的类的定义：

public class LinkedList&lt;E>

extends AbstractSequentialList&lt;E>

implements List&lt;E>, Deque&lt;E>, Cloneable, java.io.Serializable

{}

**LinkedList 源码解读**

**数据结构**  
LinkedList 是基于链表结构实现，所以在类中包含了 first 和 last 两个指针(Node)。Node 中包含了上一个节点和下一个节点的引用，这样就构成了双向的链表。每个 Node 只能知道自己的前一个节点和后一个节点，但对于链表来说，这已经足够了。

transient int size = 0;

transient Node&lt;E> first; *//链表的头指针*

transient Node&lt;E> last; *//尾指针*

*//存储对象的结构 Node, LinkedList的内部类*

private static class Node&lt;E> {

E item;

Node&lt;E> next; *// 指向下一个节点*

Node&lt;E> prev; *//指向上一个节点*

Node(Node&lt;E> prev, E element, Node&lt;E> next) {

this.item = element;

this.next = next;

this.prev = prev;

}

}

**LinkedList属性**  
LinkedList本身的 的属性比较少，主要有三个，一个是size，表名当前有多少个节点；一个是first代表第一个节点；一个是last代表最后一个节点。

transient int size = 0;

*/\*\**

*\* Pointer to first node.*

*\* Invariant: (first == null && last == null) ||*

*\* (first.prev == null && first.item != null)*

*\*/*

transient Node&lt;E> first;

*/\*\**

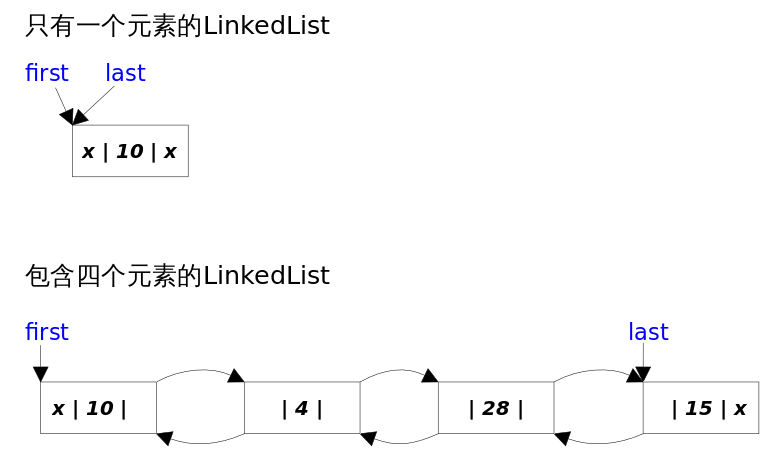
*\* Pointer to last node.*

*\* Invariant: (first == null && last == null) ||*

*\* (last.next == null && last.item != null)*

*\*/*

transient Node&lt;E> last;

**图解:**  
LinkedList底层通过双向链表实现，下节将着重讲解插入和删除元素时双向链表的维护过程。双向链表的每个节点用内部类Node表示。LinkedList通过first和last引用分别指向链表的第一个和最后一个元素。注意这里没有所谓的哑元，当链表为空的时候first和last都指向null。  
LinkedList的实现方式决定了所有跟下标相关的操作都是线性时间，而在首段或者末尾删除元素只需要常数时间。为追求效率LinkedList没有实现同步（synchronized），如果需要多个线程并发访问，可以先采用Collections.synchronizedList()方法对其进行包装。

**哑元**

首先，我们了解下什么是哑元及哑元的作用，某个参数如果在子程序或函数中没有用到，那就被称为哑元。这是程序设计语言中的一个术语，不是FORTRAN独有的，函数的形参又称“哑元”，实参又称“实元”。

**总结：**

本章介绍了一下LinkedList的数据结构和实现的接口，以及LinkedList和ArrayList的对比，什么时候下用LinkedList比较好，同时讲了一下**哑元** 的定义，让大家了解一下。后续文章会结合源码分析，重点讲解LinkedList的构造器和部分重要的常使用方法

[java集合框架之LinkedList 深度解析（一）](http://www.spring4all.com/article/1067)  
[java集合框架之LinkedList 深度解析（二）](http://www.spring4all.com/article/1071)