链滴

jdk 源码 --LinkedList

作者：fyzzz

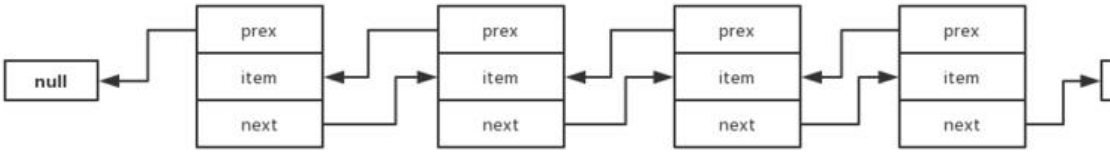
原文链接：https://ld246.com/article/1562322266191

来源网站：链滴

本文基于jdk1.8_171

LinkedList介绍

之前看了ArrayList，内部是一个数组。这次看了LinkedList，作用和ArrayList一样，但是内部是链表式。链表结构如下图：



数组和链表的区别

直接看例子：

数组： 假设有10个人去看电影，想要挨着坐，那就需要找同一排连续的10个位置坐（座位号01-10。如果想要找5号先生，那直接去第五个座位即可。但是，如果此时来了第十一个人，想要坐在4号和号先生的位置中间，那么原本5号先生和后面的所有人都要往后挪个位置，然后11号先生才能落座。

链表： 还是10个人去看电影，不用挨着坐，每个人记着自己前一个人和后一个人坐哪里就行，第一人记下一个人和最后一个人记上一个人。此时还是来了第十一个人，想要坐在4号和5号先生的位置中（不是空间意义的中间），那么5号先生更换自己上家为11号，4号先生更换下家为11号，11号记自上家是4号，下家是5号即可。但是如果需要找到5号的位置，那你需要先找1号，问他下家2号在哪里然后找2号下家3号。。。一直找到5号。

所以结论：数组查找元素比链表快，但往中间插入元素链表比数组快。

内部类

```
/**
* 链表内部元素
*/
class Node<E>{
    //当前元素
    E item;
    //上一个元素
    Node<E> prev;
    //下一个元素
    Node<E> next;
    Node(Node<E> prev,E element,Node<E> next){
        this.item = element;
        this.next = next;
        this.prev = prev;
    }
}
```

成员变量

```
//元素个数
```

```
transient int size = 0;
//链表第一个元素
transient Node<E> first;
//链表最后一个元素
transient Node<E> last;
```

常用方法

1、addFirst(E e)

```
public void addFirst(E e){
    linkFirst(e);
}

private void linkFirst(E e){
    //缓存第一个元素
    final Node<E> f = first;
    //新建元素
    final Node<E> newNode = new Node<>(null, e, f);
    //第一个元素设置为新元素
    first = newNode;
    if(f == null)
        //如果链表原先是空的，那添加元素后最后一个元素也是新元素
        last = newNode();
    else
        //如果链表原先不为空，那么原先第一个元素的前一位指向新元素
        f.prev = newNode;
    //链表元素个数加一
    size++;
    //集合修改次数加一
    modCount++;
}
```

addLast类似，不赘述。

2、add(E e)

```
public boolean add(E e){
    //add方法就是在链表末尾添加元素
    linkLast(e);
    return true;
}
```

3、add(int index,E element)

```
public void add(int index,E element){
    //这一步就是检查index是否越界，越界就抛出异常
    checkElementIndex(index);
    //如果index是size，就加在链表最后，否则就加在指定元素之前
    if(index == size)
        linkLast(element);
    else
```

```
        //node(index)方法解析往下看
        linkBefore(element,node(index));
}

void linkBefore(E e,Node<E> succ){
    //assert succ !=null
    //缓存指定元素的前一个元素
    final Node<E> pred = succ.prev;
    //要插入的新元素
    final Node<E> newNode = new Node<>(pred, e, succ);
    //指定元素的前一位变成新元素
    succ.prev = newNode;
    if(pred == null){
        //如果指定元素是第一个，那么新元素就变成第一个元素
        first = newNode;
    } else {
        //缓存的pred的下一个元素变成新元素
        pred.next = newNode;
    }
    size++;
    modCount++;
}
```

4、get(int index)

```
public E get(int index){
    checkElementIndex(index);
    //node(index)方法解析往下看
    return node(index).item;
}

Node<E> node(int index){
    if(index < (size >> 1)){
        //如果index小于size的一半，则从头开始找
        //先找到第一个元素
        Node<E> x = first;
        //往后找index次，我看的时候第一次没理解，举了例才明白，比如index=1，size=4等等
        for(int i=0; i < index; i++)
            x = x.next;
        return x;
    } else {
        //从尾开始找
        //先找到最后一个元素
        Node<E> x = last;
        //往前找index次
        for(int i = size - 1; i > index; i--)
            x = x.prev;
        return x;
    }
}
```

结语

LinkedList有自身的优点，虽然实际运用不如ArrayList频繁，但我们依然要了解它的特性以及实现方
。