## 一. ArrayList

1. 概念

ArrayList是一个有序可重复的List，并且支持动态扩容。

2. 原理

ArrayList的底层是一个动态数组，当我们进行元素的增删改查的时候其实是对动态数组的操作。

（1）添加元素

（2）添加元素到指定位置

添加元素会首先将要添加的元素的位置之后的元素整体后移一位；

然后把元素添加到指定的位置；



图 1

（3）删除元素

删除元素直接将待删除元素的位置之后的所元素依次前移；

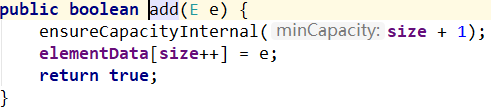


图 2

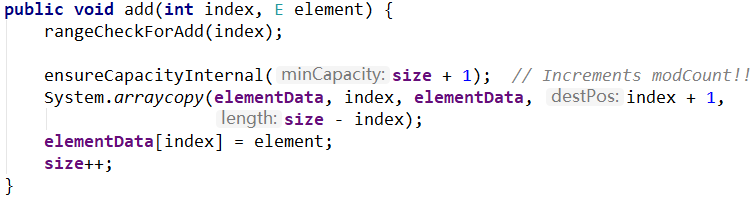
（4）查询元素

3. 源码

（1）添加元素

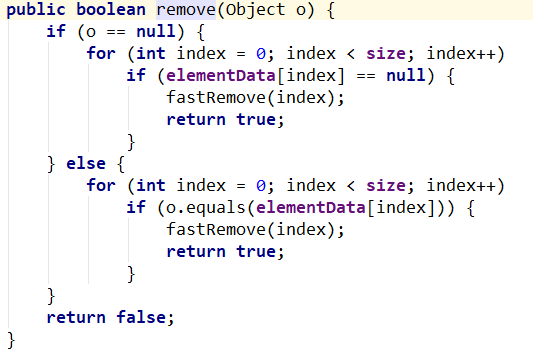


（2）添加元素到指定位置：

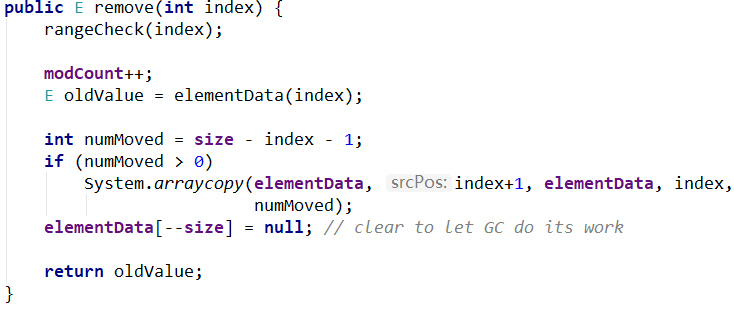


（3）删除元素

指定元素的：



指定位置的：



（4）查询元素

4. 优点

由于底层是动态数组，所以查询特别快，只需要获得指定某个位置的元素即可

5. 缺点

增加或者删除会整体移动部分元素，所以速度会慢。

6. 适用场景

查询多于增删的场景

7. 线程安全性

非线程安全。（fail-fast机制）迭代不能修改集合，否则报错。

## 二. LinkedList

1. 概念

LinkedList在用法上类似于ArrayList，但是它们的底层原理不同。

2. 原理

LinkedList的底层是一个双向链表。

该类内部有Node节点，Node会有指向上一个节点的指针、存储数据的element、指向下一个节点的指针，如图1所示。

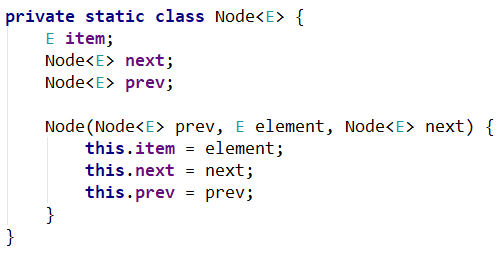


图 1

（1）添加元素

（2）添加元素到指定位置

首先，会从链表的头部或者尾部开始遍历（index/2靠前还是靠后），找到指定位置



图 2

然后进行添加操作



图 3

（3）删除指定位置的元素

首先，删除元素会同样遍历到指定的位置



图 4

然后进行删除



图 5

（4）查找元素

3. 源码

（1）添加元素

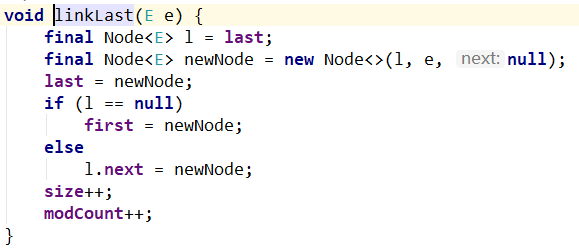
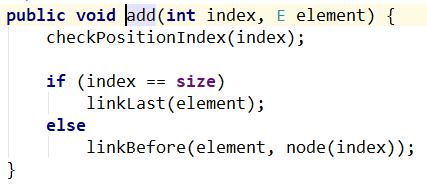


图 6

（2）添加元素到指定位置



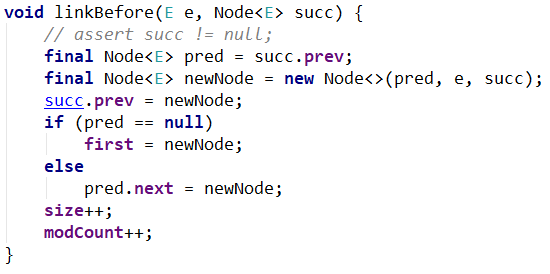


图 7

（3）删除指定位置的元素

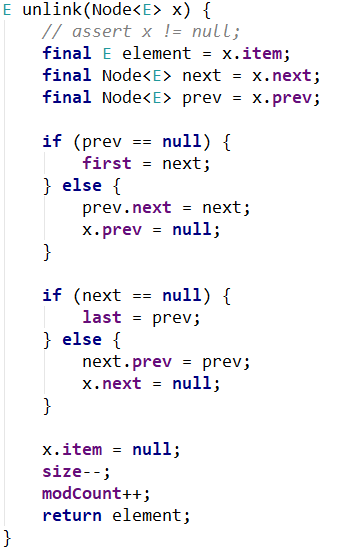


图 8

（4）查找元素

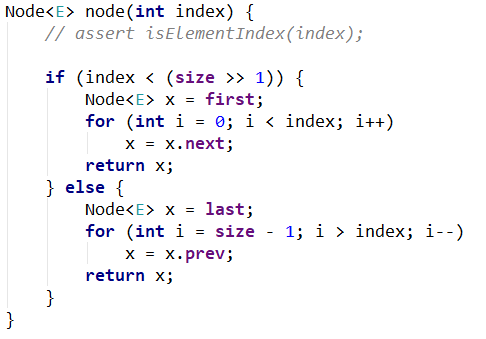


图 9

4. 优点

由于底层是链表，所以插入删除特别快，只需要改变指针的指向就可以了。

注：越中间的元素，越会影响性能，未必一定比ArrayList快

5. 缺点

查询会遍历链表，所以速度慢

6. 适用场景

查询少于增删的场景

7. 线程安全性

非线程安全。（fail-fast机制）迭代不能修改集合，否则报错。