链滴

学习笔记 |List,Set 和 Map

作者：ellenbboe

原文链接：https://ld246.com/article/1565336608441

来源网站：链滴

List

- 存取的顺序和取出的顺序是一致的,可重复添加

LinkedList

- 双向链表
- 实现了Deque的接口,可以用操作栈和队列的方式操作LinkedList
- 由于是链表,支持高效的插入和删除
- 增加删除和链表操作是差不多的,get和set的时候判断下表和长度的一半的关系来确定从头部还是尾进行遍历

ArrayList

非线程安全

- 底层是Object数组,因为有扩容机制,所以可以实现动态增长
- 初始化是根据输入的参数进行的,默认返回空的Object数组

扩容

- 初始容量为10,第一次扩容假如需要的容量小于默认容量时容量保持在10

扩容发生在add的时候

- 通过我需要的容量(实际需要)和当前的容量(数组的长度)进行比较,不够的话进行扩容
- 扩容后新的数组是原来数组的1.5倍大小
- 扩容完成后将原来数组的数组复制到新的数组中,增加和删除的时候是用c/c++的方法来对数组进行制

如果想要arraylist实现同步,可以使用synchronizedList方法

Vector

- 底层与arraylist一样,也是数组
- 通过synchronized对方法进行同步,来使线程安全,存在性能损失
- vector底层数组不够用的时候是扩容一倍

查询多用ArrayList，增删多用LinkedList

fast-fail机制

在遍历集合的时候对集合进行删除或者添加,会报错

比如根据集合size长度遍历,在这个过程中删除集合某一元素后size会变

利用**Copy-On-Write**实行读写分离,避免上述情况

CopyOnWriteArrayList

- 在修改时,复制出一个新数组,修改的操作在新数组中完成,最后将新数组交由array变量指向.
- 写加锁，读不加锁
- 读的是原来的数组,写的是数组的副本
- 一切修改完成后将新集合赋值给引用变量

缺点是由于在新数组里更改,新的数组需要时间返回,只能保证最终一致性

Set

元素不重复

- HashSet:底层是HashMap,无序,非线性同步
- TreeSet:底层TreeMap,有序
- LinkedSet:底层是LinkedHashMap

Map

存储键值对

Hashtable

- 方法有synchronize关键字修饰,所以线程安全,其他和HashMap1.6版本似乎一样,也是数组加上链表

HashMap和ConcurrentHashMap

Link

TreeMap

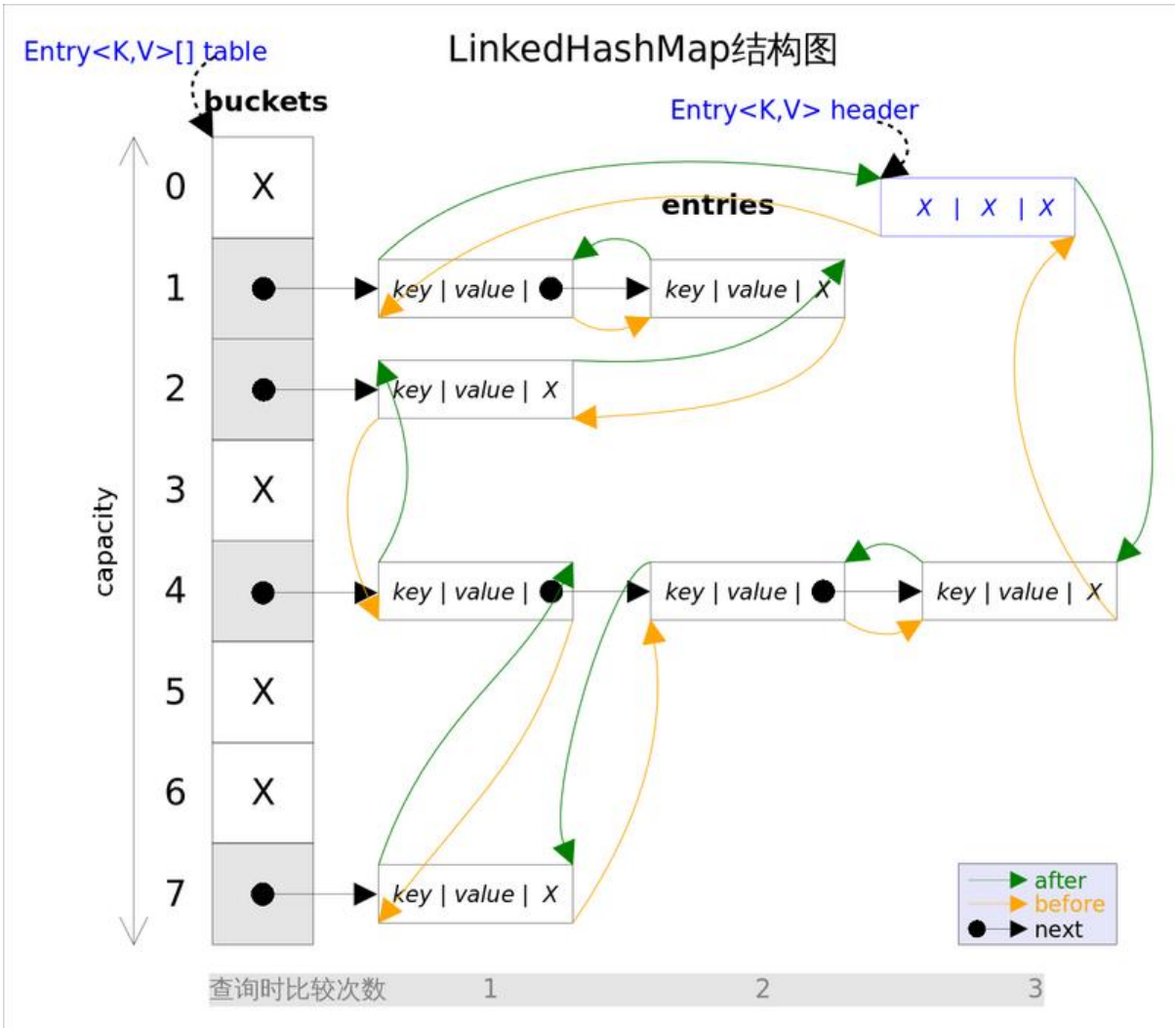
底层是红黑树

- 由于实现了nevigableMap接口而nevigable接口继承了sortedmap接口,使得treemap中的数据是序的,使用comparator和comprable来排序key,主要是用comparator,假如comparator为空,就使用然排序,key不能为null
- 自定义排序通过传入comparator对象实现
- 插入就根据Comparator进行比较,然后构建成节点插入到红黑树中,然后调整红黑树
- 查找还是根据Comparator进行比较,寻找合适位置的节点,假如没有就返回null
- 删除还是根据Comparator进行比较,寻找合适位置的节点,删掉节点然后调整红黑树

Comparable和Comparator

- 实现Comparator接口的类是一种比较工具,一种规则,需Overwrite Compare方法,比较两个对象
- 实现Comparable接口的类就是一种普通的类,需OverWrite CompareTo方法,与另一对象进行比较

LinkedHashMap



- 怀有List和Map的特点
- 继承HashMap
- 底层是HashMap和双向链表
- 基础节点组成:HashMap节点同时加上双向指针
- 基本上是在HashMap的基础上维护一个双向链表
- 遍历有插入顺序和访问顺序两种方式,LinkedHashMap遍历的是内部维护的双向链表,插入顺序就是通的顺序,访问顺序会将最近访问的结点转移到链表的末尾,然后在遍历链表

原文链接：学习笔记 |List,Set 和 Map