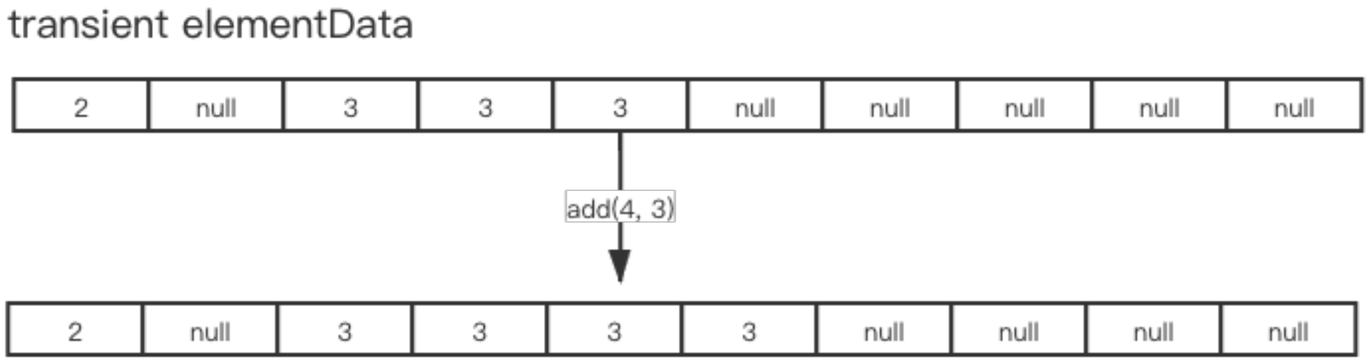
Java 集合源码剖析

目录

ArrayList


transient elementData
2
null
3
3
3
null
null
null
null
null
add(4, 3)
2
null
3
3
3
3
null
null
null
null

```java
ArrayList<String> list = new ArrayList<>();
list.add("1");
list.add("2");
list.add("3");
list.add("3");
list.add("4");
// 1.fori 删除，删不干净
for (int i = 0; i < list.size(); i++) {
    if ("3".equals(list.get(i))) {
        list.remove( o: "3");
    }
}
// 2.foreach 删除，ConcurrentModificationException
for (String i : list) {
    if ("3".equals(i)) {
        list.remove(i);
    }
}
// 3.迭代器删除，最佳
Iterator<String> iterator = list.iterator();
while (iterator.hasNext()) {
    if ("3".equals(iterator.next())) {
        iterator.remove();
    }
}
System.out.println(Arrays.toString(list.toArray()));
```

```java
List<String> list = Collections.singletonList("2333");
// UnsupportedOperationException
list.add("233");
```

Vector

1.初始化容量为 10，扩容时加倍

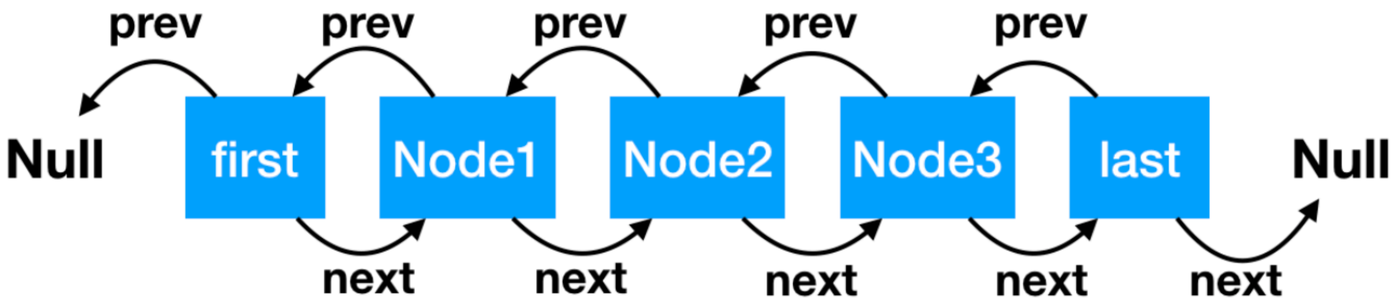
2.加锁的 ArrayList

Stack

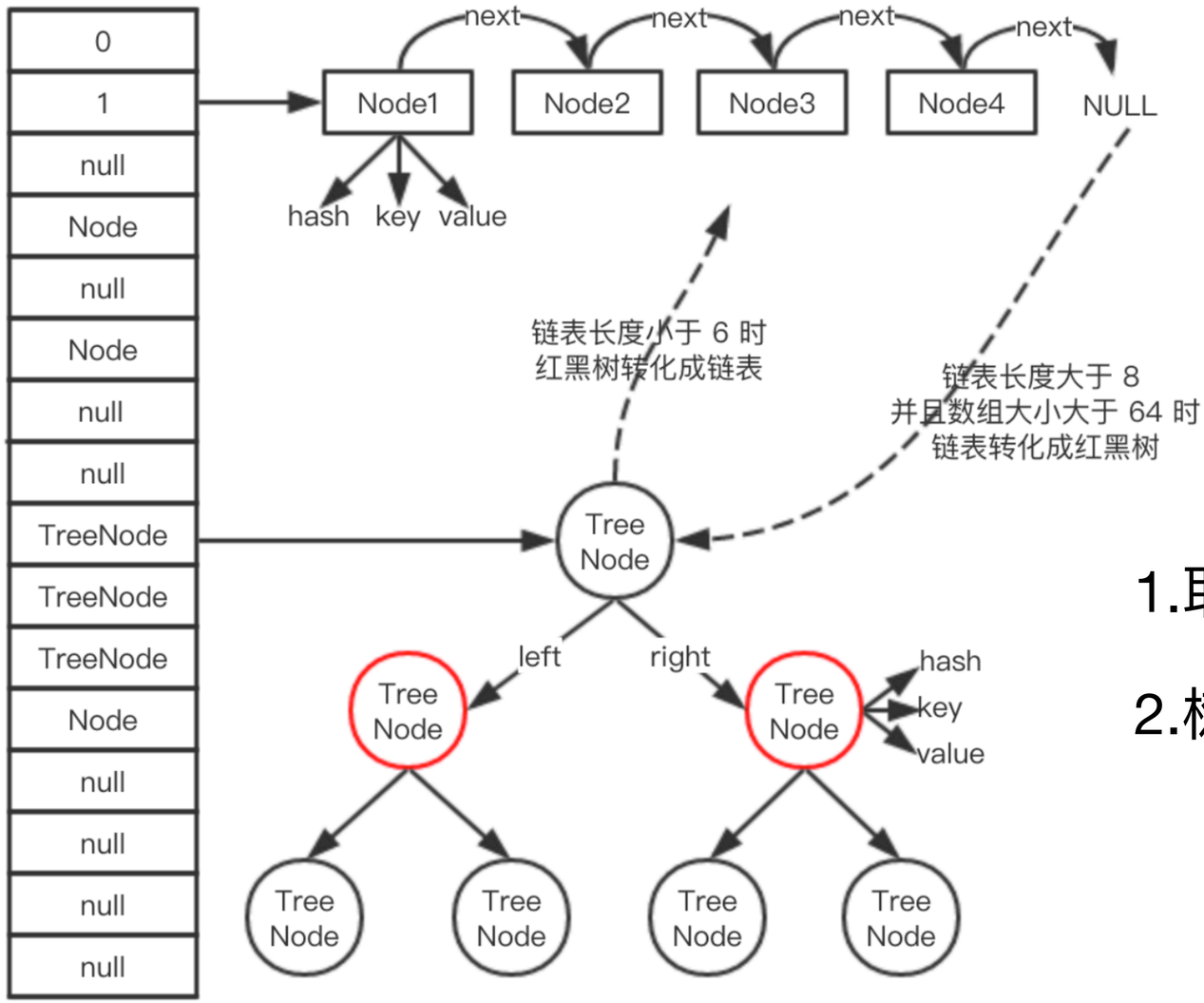
1.继承至 Vector，但初始容量为空

2.peek/pop 数组为空会抛异常

LinkedList


prev
prev
prev
prev
prev
Null
first
Node1
Node2
Node3
last
Null
next
next
next
next
next

HashMap



1.取 index = (n-1)&hash

2.树化因子为啥是 8?

Hashtable

1.初始容量 11，线程安全

2.put 时 key/value 不可为空，不然 NPE

3.取 index = (hash&0x7FFFFFFF)%tab.length

4.扩容是 x2+1

LinkedHashMap

1.继承至 HashMap，维护插入顺序，也可实现 LRU 策略

2.get 方法的实现

HashSet

1.在其构造方法中 new HashMap 来实现

2.add/contains

CopyOnWriteArrayList

1.线程安全的 ArrayList，适合读多写少的场景

2.实现：synchronized + 数组拷贝 + volatile

ConcurrentHashMap

1.线程安全的 HashMap，新增转移节点保证扩容安全

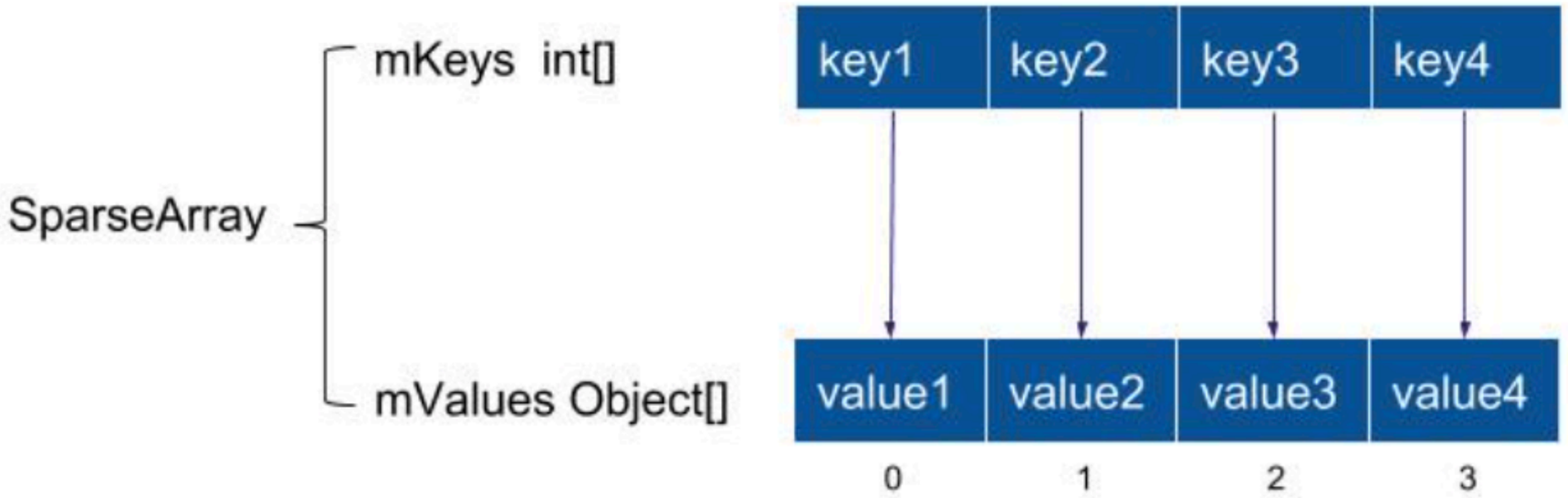
2.put 时通过 CAS + 自旋 + synchronized 保证线程安全

LinkedBlockingQueue

1.实现了 BlockingQueue 接口，在 Queue 接口上增加了阻塞的概念

2.链表存储 + ReentrantLock

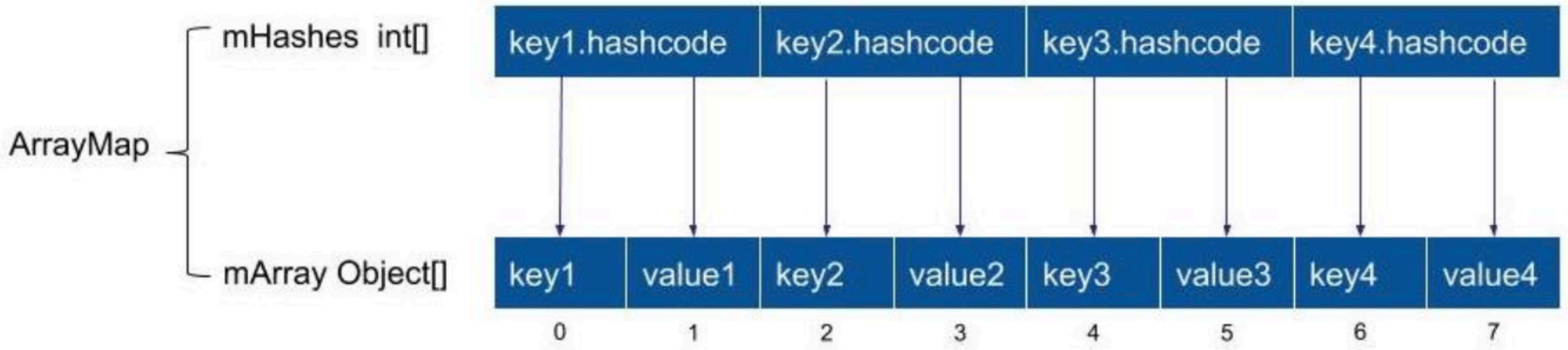
SparseArray



1.初始容量为 10，采用二分查找找到 index

2.延迟删除

ArrayMap



1.缓存了10 个长度为 4或8 的 ArrayMap 对象

2.扩容时选择靠近4 或 8 的容量，否则扩容 1.5 倍

3.存储数据不足 1/3，缩容 50%

推荐一个工具