**简介**

HashTable和HashMap一样也是一个散列表，它的存储的内容是键值对映射。

HashTable继承于Dictionary，实现了Map、Cloneable、java.io.Serializable接口。

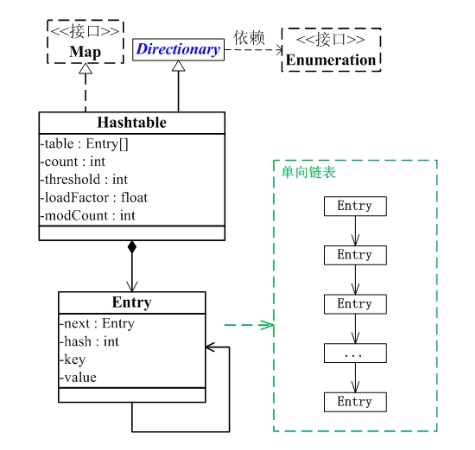
Hashtable 的函数都是同步的，这意味着它是线程安全的。HashMap线程不安全。

HashTable的key、value都不可以为null。HashMap的key、value可以为null。

Hashtable 的实例有两个参数影响其性能：初始容量 和 加载因子。

初始加载因子为0.75；初始容量默认为11，而hashMape为16。

**内部结构**



Hashtable继承于Dictionary类，HashMap则是继承了AbstractMap类，都实现了Map接口。Map是"key-value键值对"接口，Dictionary是声明了操作"键值对"函数接口的抽象类，AbstractMap实现了"键值对"的通用函数接口。

Hashtable是通过"拉链法"实现的哈希表。HashMap也是。

1. table是一个Entry[]数组类型，而Entry实际上就是一个单向链表。

2. count是Hashtable的大小，它是Hashtable保存的键值对的数量。HashMap的是size

3. threshold是HashMap的阈值，判断是否需要调整HashMap的容量。

4. loadFactor就是加载因子。

5. modCount是用来实现fail-fast机制的。

6. Entry实现了Map.Entry接口，实现了getValue(),getKey(),setValue(),equals(),hashcode()一些基本的读取操作。Entry结构跟HashMap相同

**构造函数**

1.HashTable()默认构造方法，默认容量为16，加载因子为0.75

2. HashTable(int capacity) 指定容量大小的构造函数

3. HashTable(int capacity, float loadFactor) 制定容量大小和加载因子的构造函数

4. HashTable(Map<? extends K, ? extends V> map) 包含子map的构造函数

除了名称都与HashMap相同。

**方法清单**

是否序列化 返回值 方法

synchronized void clear()

synchronized Object clone()

boolean contains(Object value)

synchronized boolean containsKey(Object key)

synchronized boolean containsValue(Object value)

synchronized Enumeration<V> elements()

synchronized Set<Entry<K, V>> entrySet()

synchronized boolean equals(Object object)

synchronized V get(Object key)

synchronized int hashCode()

synchronized boolean isEmpty()

synchronized Set<K> keySet()

synchronized Enumeration<K> keys()

synchronized V put(K key, V value)

synchronized void putAll(Map<? extends K, ? extends V> map)

synchronized V remove(Object key)

synchronized int size()

synchronized String toString()

synchronized Collection<V> values()

这里与HashMap相同的方法就不介绍了，介绍一下不同的方法

1.keys() //返回所有key的枚举对象

2.elements() // 返回所有value的枚举对象

Enumerator的作用是提供了“通过elements()遍历Hashtable的接口” 和 “通过entrySet()遍历Hashtable的接口”。因为，它同时实现了 “Enumerator接口”和“Iterator接口”。

**遍历**

除了和HashMap相同的三种遍历方法还有两种枚举遍历法

1. 根据hashtable.keys()获取到所有key的枚举对象，通过Enumeration遍历

Enumeration enu = table.keys();

while(enu.hasMoreElements()) {

System.out.print(enu.nextElement()); //输出key

System.out.print(table.get(enu.nextElement)); //输出value

}

2. 根据hashtable.elements()获取到所有value的枚举对象，通过Enumeration遍历

Enumeration enu = table.elements();

while(enu.hasMoreElements()) {

System.out.println(enu.nextElement());

}