<http://my.oschina.net/u/2322635/blog/393691>

关于EnumMap的理解，源码：

**public** V put(K key, V value) {

typeCheck(key);

**int** index = ((Enum)key).ordinal();

Object oldValue = vals[index];

vals[index] = maskNull(value);

**if** (oldValue == **null**)

size++;

**return** unmaskNull(oldValue)

}

**public** V get(Object key) {

**return** (isValidKey(key) ?

unmaskNull(vals[((Enum)key).ordinal()]) : **null**);

}

**private** **boolean** isValidKey(Object key) {

**if** (key == **null**)

**return** **false**;

// Cheaper than instanceof Enum followed by getDeclaringClass

Class keyClass = key.getClass();

**return** keyClass == keyType || keyClass.getSuperclass() == keyType;

}

一、Java HashMap的工作原理  
面试的时候经常会遇见诸如：“java中的HashMap是怎么工作的”，“HashMap的get和put内部的工作原理”这样的问题。  
Put :  
让我们看下put方法的实现：  
public V put(K key, V value) {  
  if (key == null)  
   return putForNullKey(value);  
  int hash = hash(key.hashCode());  
  int i = indexFor(hash, table.length);  
  for (Entry<k , V> e = table[i]; e != null; e = e.next) {  
   Object k;  
   if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {  
    V oldValue = e.value;  
    e.value = value;  
    e.recordAccess(this);  
    return oldValue;  
   }  
  }  
  modCount++;  
  addEntry(hash, key, value, i);  
  return null;  
 }  
现在我们一步一步来看下上面的代码。  
    1).对key做null检查。如果key是null，会被存储到table[0]，因为null的hash值总是0。  
    2).key的hashcode()方法会被调用(hashCode方法的定义用到了native关键字，表示它是由C或C++采用较为底层的方式来实现的，你可以认为它返回了该对象的内存地址；而缺省equals  
  
则认为，只有当两者引用同一个对象时，才认为它们是相等的。)，然后计算hash值。hash值用来找到存储Entry对象的数组的索引。有时候hash函数可能写的很不好，所以JDK的设计者添加  
  
了另一个叫做hash()的方法，它接收刚才计算的hash值作为参数。如果你想了解更多关于hash()函数的东西，可以参考：hashmap中的hash和indexFor方法  
    3).indexFor(hash,table.length)用来计算在table数组中存储Entry对象的精确的索引。  
    4)如果两个key有相同的hash值(也叫冲突)，他们会以链表的形式来存储。所以，这里我们就迭代链表。用当前Entry的value来替换之前的value。  
  
Get:  
现在我们来看下get方法的实现：  
public V get(Object key) {  
  if (key == null)  
   return getForNullKey();  
  int hash = hash(key.hashCode());  
  for (Entry<k , V> e = table[indexFor(hash, table.length)]; e != null; e = e.next) {  
   Object k;  
   if (e.hash == hash && ((k = e.key) == key || key.equals(k)))  
    return e.value;  
  }  
  return null;  
 }  
  
当你传递一个key从hashmap获取value的时候：  
    1).对key进行null检查。如果key是null，table[0]这个位置的元素将被返回。  
    2).key的hashcode()方法被调用，然后计算hash值。  
    3).indexFor(hash,table.length)用来计算要获取的Entry对象在table数组中的精确的位置，使用刚才计算的hash值。  
    4).在获取了table数组的索引之后，会迭代链表，调用equals()方法检查key的相等性，如果equals()方法返回true，get方法返回Entry对象的value，否则，返回null。  
  
要牢记以下关键点：  
  
    1)HashMap有一个叫做Entry的内部类，它用来存储key-value对。  
    2)上面的Entry对象是存储在一个叫做table的Entry数组中。  
    3)table的索引在逻辑上叫做“桶”(bucket)，它存储了链表的第一个元素。  
    4)key的hashcode()方法用来找到Entry对象所在的桶。  
    5)如果两个key有相同的hash值，他们会被放在table数组的同一个桶里面。  
    6)key的equals()方法用来确保key的唯一性。  
    7)value对象的equals()和hashcode()方法根本一点用也没有。  
  
  
二、Hashtable、LinkedHashMap、TreeMap、SortedMap、WeakHashMap、IdentityHashMap、ConcurrentHashMap的区别:  
(1) HashMap与HashTable的区别：  
a.Hashtable中的对象是线程安全的。而HashMap则是异步的，因此HashMap中的对象并不是线程安全的。因为同步的要求会影响执行的效率，所以如果你不需要线程安全的集合那么使用  
  
HashMap是一个很好的选择，这样可以避免由于同步带来的不必要的性能开销，从而提高效率。  
b.值：HashMap可以让你将空值作为一个表的条目的key或value，但是Hashtable是不能放入空值的。HashMap最多只有一个key值为null，但可以有无数多个value值为null。  
注意：  
1、用作key的对象必须实现hashCode和equals方法。  
2、不能保证其中的键值对的顺序  
3、尽量不要使用可变对象作为它们的key值。  
(2) LinkedHashMap：  
     它的父类是HashMap，使用双向链表来维护键值对的次序，迭代顺序与键值对的插入顺序保持一致。LinkedHashMap需要维护元素的插入顺序，插入性能略低于HashMap，但在迭代访问元  
  
素时有很好的性能，因为它是以链表来维护内部顺序。  
(3) TreeMap和SortedMap:  
Map接口派生了一个SortMap子接口，SortMap的实现类为TreeMap。TreeMap也是基于红黑树对所有的key进行排序，有两种排序方式：自然排序和定制排序。HashMap通常比TreeMap快一点(树  
  
和哈希表的数据结构使然)，建议多使用HashMap，在需要排序的Map时候才用TreeMap。   
(4) WeakHashMap：  
 WeakHashMap与HashMap的用法基本相同，区别在于：后者的key保留对象的强引用，即只要HashMap对象不被销毁，其对象所有key所引用的对象不会被垃圾回收;WeakHashMap适合短时间内就  
  
过期的缓存时最好使用weakHashMap，它包含了一个自动调用的方法expungeStaleEntries，这样就会在值被引用后直接执行这个隐含的方法，将不用的键清除掉。  
(5) IdentityHashMap类：  
IdentityHashMap与HashMap基本相似，只是当两个key严格相等时，即key1==key2时，它才认为两个key是相等的 。IdentityHashMap也允许使用null，但不保证键值对之间的顺序。  
(6) EnumMap类：  
1、EnumMap中所有key都必须是单个枚举类的枚举值，创建EnumMap时必须显示或隐式指定它对应的枚举类。  
2、EnumMap根据key的自然顺序，即枚举值在枚举类中定义的顺序，来维护键值对的次序。  
3、EnumMap不允许使用null作为key值，但value可以。  
(7) ConcurrentHashMap：  
1.ConcurrentHashMap对整个桶数组进行了分段，而HashMap则没有  
2.ConcurrentHashMap在每一个分段上都用锁进行保护，从而让锁的粒度更精细一些，并发性能更好，而HashMap没有锁机制，不是线程安全的。  
  
  
三、红黑树的理解？  
红黑树是一种自平衡二叉查找树，红黑树是一种很有意思的平衡检索树；每次插入的时候都要进行计算，保证二叉树的平衡；如果有2的N次方数据量级，查询的时候只需要查询N次即可。  
我们对任何有效的红黑树加以如下增补要求:  
        1.节点是红色或黑色。  
        2.根是黑色。  
        3.所有叶子（外部节点）都是黑色。  
        4.每个红色节点的两个子节点都是黑色。(从每个叶子到根的所有路径上不能有两个连续的红色节点)  
        5.从每个叶子到根的所有路径都包含相同数目的黑色节点。   
 这些约束强制了红黑树的关键属性: 从根到叶子的最长的可能路径不多于最短的可能路径的两倍长。结果是这个树大致上是平衡的。