JDK重要模块代码分析---HashMap源码分析

## 第一部分：代码逻辑分析（JDK1.8）

import java.util.HashMap;

HashMap(散列表、哈希表)。我们知道，java.util中的clloection集合类中， 最为常用的两种是List和Map类，而HashMap就是属于Map集合的阵营。HashMap最常见的实现方法是拉链法——即一系列链表为数组元素组成的数组。HashMap由链表+数组组成，他的底层结构是一个数组，而数组的元素是一个单向链表。每个数组储存的元素代表的是每一个链表的头结点。

HashMap是Map类的实现类之一，继承自父类AbstractMap，已知的直接子类有LinkedHashMap和PrinterStateReasons。Hash是基于哈希表的Map接口的实现类。

HashMap的一个实例有两个影响其性能的参数： 初始容量和负载因子。容量是哈希表中的桶数，初始容量只是创建哈希表时的容量。负载因子是在容量自动增加之前允许哈希表得到满足的度量。当在散列表中的条目的数量超过了负载因数和电流容量的乘积，哈希表被重新散列（即内部数据结构被重建），使得哈希表具有桶的大约两倍。

作为一般规则，默认负载因子（0.75）提供了时间和空间成本之间的良好折中。更高的值会降低空间开销，但会增加查找成本（反映在HashMap类的大部分操作中，包括get和put ）。在设置其初始容量时，应考虑地图中预期的条目数及其负载因子，以便最小化重新组播操作的数量。如果初始容量大于最大条目数除以负载因子，则不会发生重新排列操作。

HashMap的主干是一个Entry数组。Entry是HashMap的基本组成单元，每一个Entry包含一个key-value键值对。

static class Node<K,V> implements Map.Entry<K,V> {  
 final int hash;  
 final K key;  
 V value;  
 Node<K,V> next;

…;

}

## 第二部分 代码的案例

###### 案例一：equals

Obey the general contract when overriding equals():

* Reflexive

x.equals(x) must be true

Code example（HashMap类的第306行）：

public final boolean equals(Object o) {  
 if (o == this) //Reflexive  
 return true;   
 if (o instanceof Map.Entry) {  
 Map.Entry<?,?> e = (Map.Entry<?,?>)o;  
 if (Objects.*equals*(key, e.getKey()) &&  
 Objects.*equals*(value, e.getValue()))  
 return true;  
 }  
 return false;  
}

###### 案例二：Always override toString

默认的toString方法：

调用Object的toString方法：return class name + “@” + unsigned hexadecimal representation of the hash code of the object

Code example（HashMap类的第294行）：

static class Node<K,V> implements Map.Entry<K,V> {  
 …；  
 public final String toString() { return key + "=" + value; }  
 …；  
}

###### 案例三：Always override hashCode when you override equals

Contract:

1) hashCode() must return same integer on multiple calls, as long as equals() unchanged

2) If x.equals(y), then x, y have same hashcode

3) It is not required that unequal objects have different hashcodes.

Code example（HashMap类的第296行和306行）:

public final int hashCode() {  
 return Objects.*hashCode*(key) ^ Objects.*hashCode*(value);  
}

…;

public final boolean equals(Object o) {  
 if (o == this)  
 return true;  
 if (o instanceof Map.Entry) {  
 Map.Entry<?,?> e = (Map.Entry<?,?>)o;  
 if (Objects.*equals*(key, e.getKey()) &&  
 Objects.*equals*(value, e.getValue()))  
 return true;  
 }  
 return false;  
}

###### 案例四：Prefer Collection to Stream as a return type

The Collection interface is a subtype of Iterable and has a stream method, so it provides for both iteration and stream access. Therefore, Collection or an appropriate subtype is generally the best return type for a public, sequence- returning method.

Code example（HashMap类的第1316行）：

*/\*\*  
 \* Returns a shallow copy of this <tt>HashMap</tt> instance: the keys and  
 \* values themselves are not cloned.  
 \*  
 \** ***@return*** *a shallow copy of this map  
 \*/*@SuppressWarnings("unchecked")  
@Override  
public Object clone() {  
 HashMap<K,V> result;  
 try {  
 result = (HashMap<K,V>)super.clone();  
 } catch (CloneNotSupportedException e) {  
 // this shouldn't happen, since we are Cloneable  
 throw new InternalError(e);  
 }  
 result.reinitialize();  
 result.putMapEntries(this, false);  
 return result;  
}

###### 案例五：Define immutable objects

方法：将对象定义为静态final变量，不允许改变

如HashMap中的两个默认的初始容量值16和负载因子值0.75

Code example（HashMap类第236行和248行）：

*/\*\*  
 \* The default initial capacity - MUST be a power of two.  
 \*/*static final int *DEFAULT\_INITIAL\_CAPACITY* = 1 << 4; // aka 16

*/\*\*  
 \* The load factor used when none specified in constructor.  
 \*/*static final float *DEFAULT\_LOAD\_FACTOR* = 0.75f;

案例六：Write doc comments for all exposed API elements

Precede every exported class, interface, constructor,

method, and field description with a doc comment

Doc comment should describe CONTRACT between method and its client

No two members or constructors should have the same summary description

Every method should have

@param tag for each parameter

@return tag (unless return type is void)

@throws tag for each exception (both checked and unchecked)

Code example（HashMap类第438行）：

*/\*\*  
 \* Constructs an empty <tt>HashMap</tt> with the specified initial  
 \* capacity and load factor.  
 \*  
 \** ***@param*** *initialCapacity the initial capacity  
 \** ***@param*** *loadFactor the load factor  
 \** ***@throws*** *IllegalArgumentException if the initial capacity is negative  
 \* or the load factor is nonpositive  
 \*/*public HashMap(int initialCapacity, float loadFactor) {  
 if (initialCapacity < 0)  
 throw new IllegalArgumentException("Illegal initial capacity: " +  
 initialCapacity);  
 if (initialCapacity > *MAXIMUM\_CAPACITY*)  
 initialCapacity = *MAXIMUM\_CAPACITY*;  
 if (loadFactor <= 0 || Float.*isNaN*(loadFactor))  
 throw new IllegalArgumentException("Illegal load factor: " +  
 loadFactor);  
 this.loadFactor = loadFactor;  
 this.threshold = *tableSizeFor*(initialCapacity);  
}

###### 案例七：Minimize the scope of local variables

The most powerful technique for minimizing the scope of a local variable is to declare it where it is first used.

Code example：（HashMap类第1296行）

@Override  
public void replaceAll(BiFunction<? super K, ? super V, ? extends V> function) {  
 Node<K,V>[] tab;  
 if (function == null)  
 throw new NullPointerException();  
 if (size > 0 && (tab = table) != null) {  
 int mc = modCount;  
 for (int i = 0; i < tab.length; ++i) {  
 for (Node<K,V> e = tab[i]; e != null; e = e.next) {  
 e.value = function.apply(e.key, e.value);  
 }  
 }  
 if (modCount != mc)  
 throw new ConcurrentModificationException();  
 }  
}

###### 案例八：Eliminate unchecked warnings

If you can’t eliminate a warning, but you can prove that the code that provoked the warning is typesafe, then (and only then) suppress the warning with an @SuppressWarnings("unchecked") annotation.

Code example（HashMap类第1322行）：

*/\*\*  
 \* Returns a shallow copy of this <tt>HashMap</tt> instance: the keys and  
 \* values themselves are not cloned.  
 \*  
 \** ***@return*** *a shallow copy of this map  
 \*/*@SuppressWarnings("unchecked")  
@Override  
public Object clone() {  
 HashMap<K,V> result;  
 try {  
 result = (HashMap<K,V>)super.clone();  
 } catch (CloneNotSupportedException e) {  
 // this shouldn't happen, since we are Cloneable  
 throw new InternalError(e);  
 }  
 result.reinitialize();  
 result.putMapEntries(this, false);  
 return result;  
}

###### 案例九：Design and document for inheritance or else prohibit it

为继承编写文档

Code example（HashMap类的父类AbstractMap第129行）：

*/\*\*  
 \* {****@inheritDoc****}  
 \*  
 \** ***@implSpec*** *\* This implementation iterates over <tt>entrySet()</tt> searching  
 \* for an entry with the specified key. If such an entry is found,  
 \* the entry's value is returned. If the iteration terminates without  
 \* finding such an entry, <tt>null</tt> is returned. Note that this  
 \* implementation requires linear time in the size of the map; many  
 \* implementations will override this method.  
 \*  
 \** ***@throws*** *ClassCastException {****@inheritDoc****}  
 \** ***@throws*** *NullPointerException {****@inheritDoc****}  
 \*/*public V get(Object key) {  
 Iterator<Entry<K,V>> i = entrySet().iterator();  
 if (key==null) {  
 while (i.hasNext()) {  
 Entry<K,V> e = i.next();  
 if (e.getKey()==null)  
 return e.getValue();  
 }  
 } else {  
 while (i.hasNext()) {  
 Entry<K,V> e = i.next();  
 if (key.equals(e.getKey()))  
 return e.getValue();  
 }  
 }  
 return null;  
}

###### 案例十：Check parameters for validity

Most methods and constructors have some restrictions on what values may be passed into their parameters. For example, it is not uncommon that index values must be non-negative and object references must be non-null. You should clearly document all such restrictions and enforce them with checks at the beginning of the method body. This is a special case of the general principle that you should attempt to detect errors as soon as possible after they occur. Failing to do so makes it less likely that an error will be detected and makes it harder to determine the source of an error once it has been detected.If an invalid parameter value is passed to a method and the method checks its

Code example（HashMap类第438-477行）：

*/\*\*  
 \* Constructs an empty <tt>HashMap</tt> with the specified initial  
 \* capacity and load factor.  
 \*  
 \** ***@param*** *initialCapacity the initial capacity  
 \** ***@param*** *loadFactor the load factor  
 \** ***@throws*** *IllegalArgumentException if the initial capacity is negative  
 \* or the load factor is nonpositive  
 \*/*public HashMap(int initialCapacity, float loadFactor) {  
 if (initialCapacity < 0)  
 throw new IllegalArgumentException("Illegal initial capacity: " +  
 initialCapacity);  
 if (initialCapacity > *MAXIMUM\_CAPACITY*)  
 initialCapacity = *MAXIMUM\_CAPACITY*;  
 if (loadFactor <= 0 || Float.*isNaN*(loadFactor))  
 throw new IllegalArgumentException("Illegal load factor: " +  
 loadFactor);  
 this.loadFactor = loadFactor;  
 this.threshold = *tableSizeFor*(initialCapacity);  
}  
  
*/\*\*  
 \* Constructs an empty <tt>HashMap</tt> with the specified initial  
 \* capacity and the default load factor (0.75).  
 \*  
 \** ***@param*** *initialCapacity the initial capacity.  
 \** ***@throws*** *IllegalArgumentException if the initial capacity is negative.  
 \*/*public HashMap(int initialCapacity) {  
 this(initialCapacity, *DEFAULT\_LOAD\_FACTOR*);  
}  
  
*/\*\*  
 \* Constructs an empty <tt>HashMap</tt> with the default initial capacity  
 \* (16) and the default load factor (0.75).  
 \*/*public HashMap() {  
 this.loadFactor = *DEFAULT\_LOAD\_FACTOR*; // all other fields defaulted  
}

代码中检查了initialCapacity、loadFactor的有效性