# Hashmap底层实现原理

Hashmap的数据结构数组，链表，红黑树（jdk1.8）组成，数组如果不扩容的话默认是由16位长度的数组，如果发生冲突的话会产生单线链表来解决冲突。



Hashmap特点

1. 快速存储 ：比如当我们对hashmap进行get和put的时候速度非常快
2. 快速查找（时间复杂度o(1)）当我们通过key去get一个value的时候时间复杂度非常的低，效率非常高
3. 可伸缩：1数组扩容，边长。2，单线列表如果长度超过8的话会变成红黑树

这里就是介绍了hashmap的数据结构

接着如果说到了hashmap的话就不得不说下hashmap的核心hash算法，

Hash算法

在聊哈算法之前我们要知道在Java中所有对象都有hashcode（使用key的），如果使用object对象get hashcode的话会得到要给int类型的指，我们在hashmap中主要是用他的key去计算它的值的。

Hash值的计算

Hash值=（hashcode）^(hashcode >>> 16)

Hashcode予hashcode自己向右位移16位的异或运算。这样可以确保算出来的值足够随机。因为进行hash计算的时候足够分散，以便于计算数组下标的时候算的值足够分散。前面说过hashmap的底层是由数组组成，数组默认大小是16，那么数组下标是怎么计算出来的呢，那就是

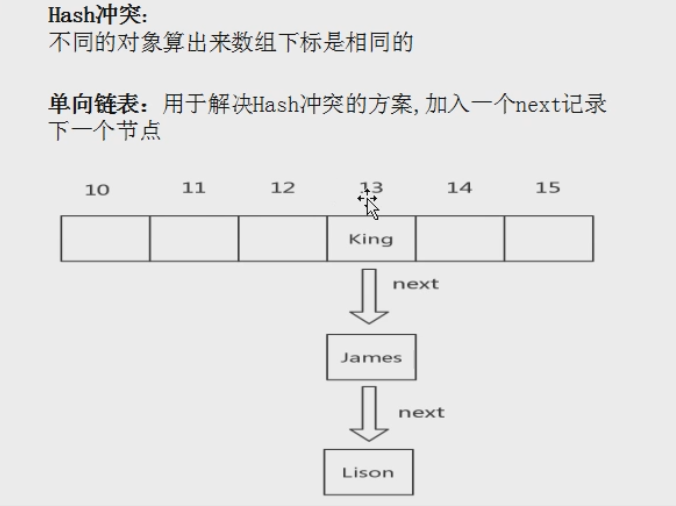
数组下标：hash&(16-1) = hash%16

对哈市计算得到的hash进行16的求余，得到一个16的位数，比如说是1到15之间的一个数，hashmap会与hash值和15进行予运算。这样可以效率会更高。计算机中会容易识别这种向右位移，向坐位移

Hash冲突

不同的对象算出来的数组下标是相同的这样就会产生hash冲突。

Hash冲突会产生单线链表。



当单线链表达到一定长度后效率会非常低，那么在jdk1.8以后的话，加入了红黑树，也就是说单线列表达到一定长度后就会变成一个红黑树

Hashmap底层原理扩容

扩容

数组长度变成2倍 0.75

触发条件

数组存储比例达到75% -- 0.75

一下是需要理解的：

Hashmap的扩容并不是为单线链表准备的，单线链表只是为了解决hash冲突准备的。也就是说当数组达到一定长度，比如说hashmap默认数组长度是16，那么达到出发条件，数组存储比例达到了75% ，也就是16\*0.75=12的时候就会发生扩容

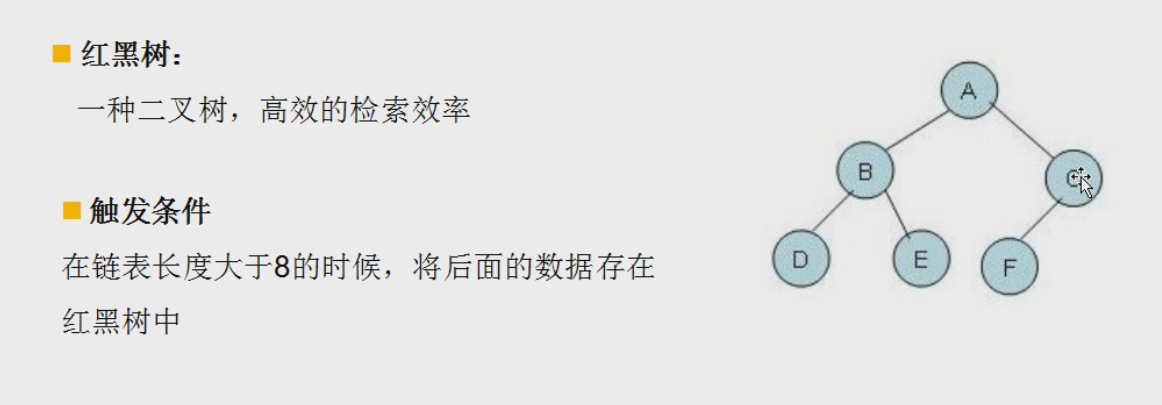
红黑树

一种二叉树，高效的检索效率

前面说了，当链表达到一定长度后，链表就会变成红黑树

触发条件

在链表长度大于8的时候，将后面的数据存在二叉树中



红黑树就是上图所示

A下面有两个节点BC，B和C下面又有DEF

总结

Hashmap的数据结构

数组，单线链表，红黑树（1.8）

Hashmap的特点

1. 快速存储
2. 快速查找
3. 可伸缩

Hash算法

为什么要用hash算法。在我们Java中任何对象都有hashcode，hash算法就是通过hashcode与自己进行向右位移16的异或运算。这样做是为了计算出来的hash值足够随机，足够分散，还有产生的数组下标足够随机

还有就是hash冲突

面试的时候问为什么会有hash冲突

比如说我们同样要插入一个数组，大家计算出来都是13号。那么这个时候就会产生hash冲突，为了解决hash冲突，我们的数组就会变成单线链表，然后当我们的单线链表达到一定长度的时候就产生红黑树。