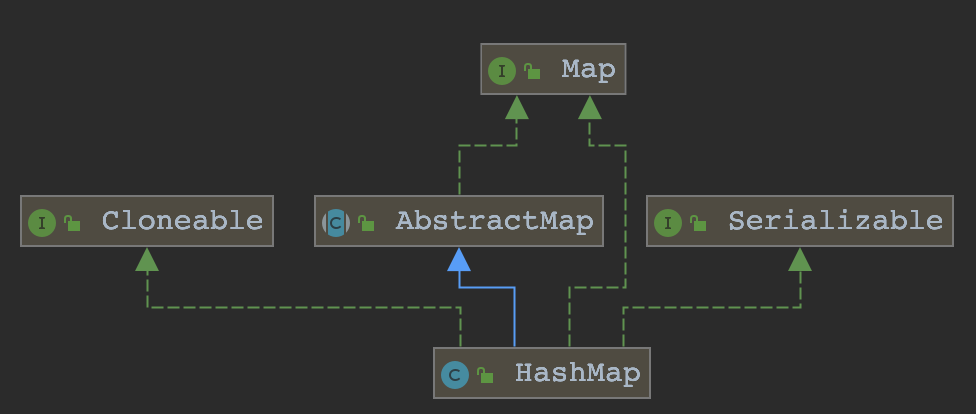
**HashMap的继承体系结构**：



从HashMap的继承体系结构中我们可以发现一个有意思的现象：HashMap继承了类AbstractMap，而AbstractMap是一个实现了Map接口的抽象类，但是HashMap又再一次实现了Map接口，相当于**HashMap实现了两次Map接口**。

**HashMap内部的节点结构**：

HashMap内部存在着两种类型的节点，这两个类都是HashMap的静态内部类。

（1）Node

HashMap内部的大多数节点都是以Node形式存在的，Node类实现了Map.Entry接口，包含4个字段。

final int hash：节点的哈希值。

final K key：键。

V value：值。

Node<K, V> next：指向下一个Node节点的指针。

这里有一个问题，**为什么Node节点要有指向下一个Node节点的指针呢**？

（2）TreeNode

TreeNode类继承自Map.Entry类，而Map.Entry类又继承自Map.Node类，因此**TreeNode是Node的子类**。TreeNode是**红黑树**的节点。

**HashMap是如何存储数据的**：

HashMap内部有一个字段：transient Node<K,V>[] table。从这个字段我们可以看出，**HashMap内部维护的是一个Node数组，将相应的键值对的Node节点存储在这个table数组里**。

**HashMap中的两个重要概念**：

**初始容量**：新建一个HashMap时，table数组的大小。默认的初始容量是16（必须是2的幂）。

**负载率**：评判rehash操作是否需要进行的指标，当HashMap中的Node节点数量大于负载率和当前table数组大小的乘积时，rehash操作就需要发生了。默认的负载率是0.75。

**哈希冲突的处理**：

MIN\_TREEIFY\_CAPACITY：只有table数组的长度大于这个数时，才可能将链表转换为红黑树存储。