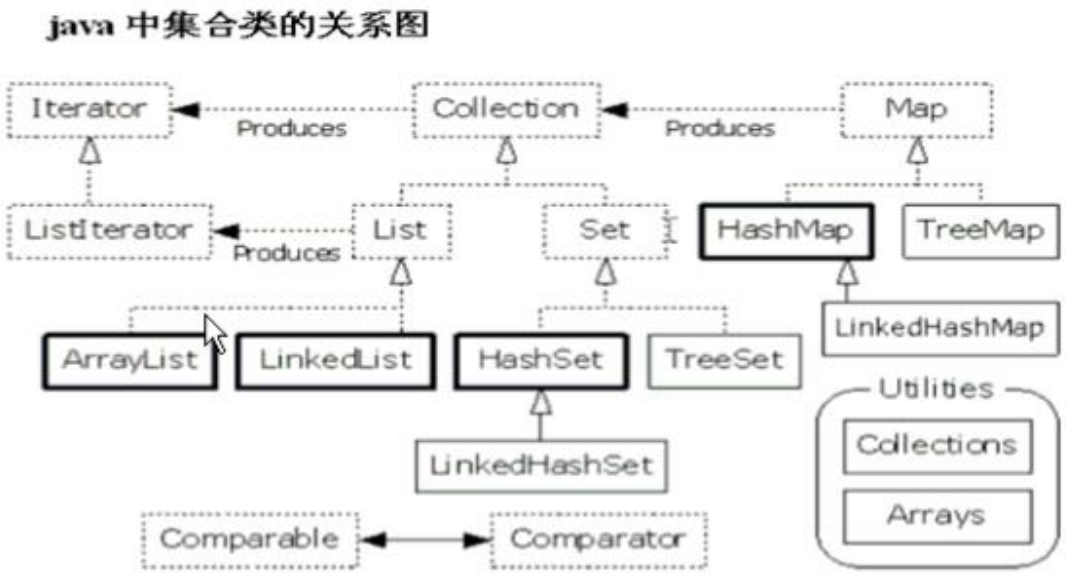
JAVA 集合类

应用场景：

1. 需要在任意时间创建任意对象

定义：



List

ArrayList

长于随机访问元素，但插入和删除元素比较慢

LinkedList

长于插入和删除元素，但访问元素的速度比较慢

添加可以把它当成栈、队列和双端队列的方法

Iterator:

将遍历操作与数据的底层结构分离，可以遍历任何实现 Interator 接口的元素

ListIterator

只能针对 List 集合

可以双向移动

比 Iterator 的功能更加强大

Set

不包含重复元素，比如得到一篇文章中的单词

HashSet

查询速度快，但不保证顺序

底层是散列表

TreeSet

保证元素的顺序

底层是红黑树

LinkedHashSet

保证元素的插入顺序

使用链表维护元素的插入数据

Map

将对象映射到另一个对象，比如统计文章中单词出现的次数

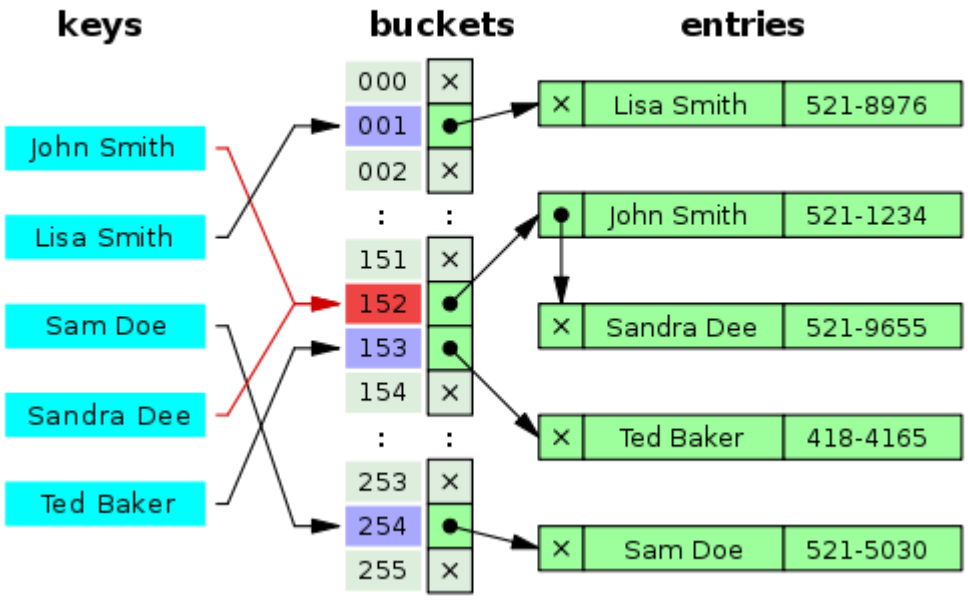
适配器模式：

你使用了别人的类，但你不能修改，你依赖于别人类的一些功能，但又需要修改一下才能满足需求，这个时候可以考虑适配器模式

比如 A 类继承了 ArrayList(实现了 iterator 接口),A 类需要它的遍历可以是反向遍历，但同时保留前向遍历。这个时候 A 类可以使用内部类重新实现以 iterator 接口。A 类就是适配器。

HashMap

工作机制：



调用 hasCode 方法，找到 key 在 buckets 位置（Lisa Smith 在 001），调用 equals 方法找到（key）在单链表的实际位置，从而找到值。

hasCode:

当往容器中填充自己定义的类的引用时，需要重新定义 hasCode 方法，因为默认是根据引用地址来来查找 key 在 buckets 的位置，这样会造成就是这对象的值是相同，但是由于引用地址不一样导致找不到相同的对象

equals

重写 equals 的原因和上面类似。