# 集合树图：



**List**

**特点:**

①有序可以插入元素到指定位置.可以通过下标来访问list里面的元素(类似数组).

②因为有序所以可以通过for循环遍历,迭代器

**LinkedList:**

**特点:**

①实现了List接口,元素可以是null值, 因为有get(), remove(),insert()在链表头尾. 链表可以做成stack, queue.

②没有同步方法,使用多线程时,要手动构造一个同步的List.

③由于基于链表实现,地址随意,在任意点插入,删除元素的时间复杂度都是常数.但是在查询某个元素值时,花费时间复杂度高.

**ArrayList:**

**特点:**

①实现了List接口,元素可以是null值,链表可以做成stack, queue.

②没有同步方法,使用多线程时,要手动构造一个同步的List.

③由于基于数组实现,地址是连续的,在任意点插入,删除元素的时间复杂度都是O(n).但是在查询某个元素值时,花费时间复杂度低(线性)

**总结:访问次数多用ArrayList,增删次数多用LinkedList.**

**Set**

**特点:**

①Set不可以存储相同的元素(equals()方法). 元素在set中的位置其实是由HashCode决定(一般位置也是定死的)

②因为set无序,所以只可以用迭代器.

**TreeSet**

**特点:**

①基于二差树（红黑树的数据结构）实现的,Treeset的数据全部都是排序好的, 不允许存放null值.

**HashSet**

**特点:**

①基于哈希表实现,无续,可以放入null,因为set不允许有重复元素,所以只可以有一个null.

②要实现Hashcode方法.

总结:HashSet基于Hash算法实现,性能都优于TreeSet,如果要查找就使用HashSet,排序就需要TreeSet.

**Map**

**特点:**

①存储键值对; 主要有HashMap, Hashtable, LinkedHashMap 和TreeMap

**HashMap**

**特点:**

①最常用的Map,根据HashCode存储数据,根据键可以获取值.访问速度快. 键可以为空,但必须唯一. (只可以有一个键为null)

②不支持同步,如果需要就要实现同步方法.

**Hashtable**

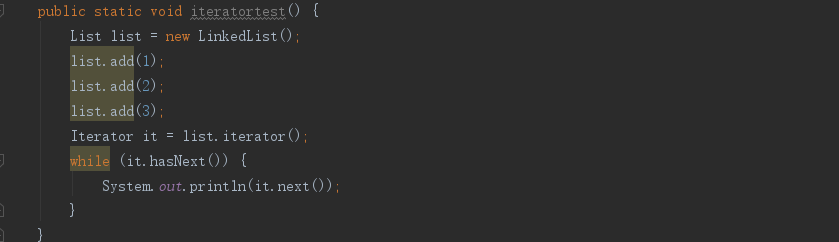
**特点:**

①键或值都不能为空;

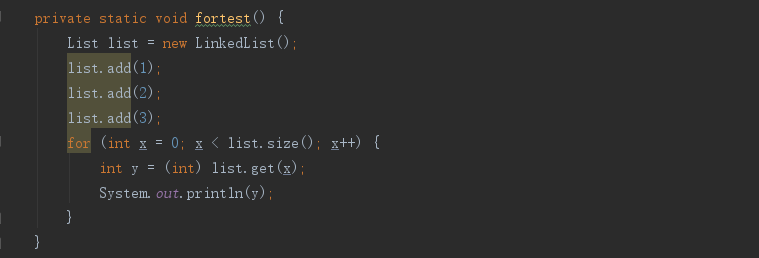
②它支持线程的同步，只允许一个线程写Hashtable,也导致了Hashtable在写入时会比较慢。

**集合遍历方法:**

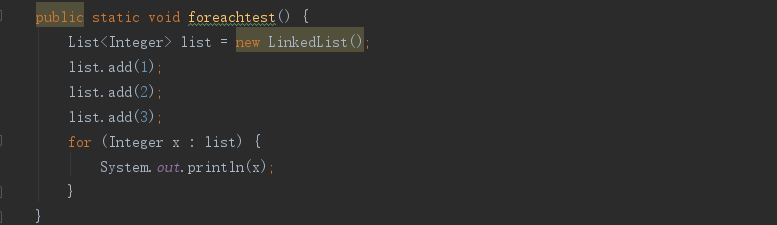
**①通过迭代器实现**



**②通过for循环**

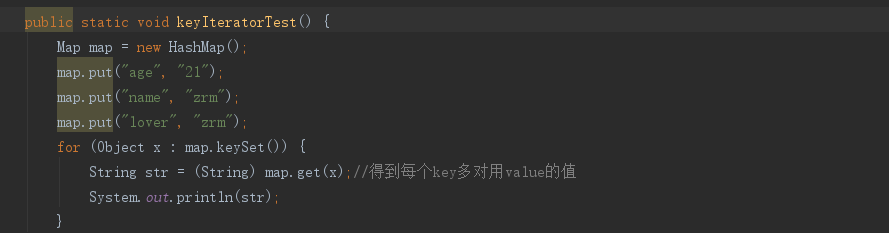


**③通过for each(也是迭代器)循环**

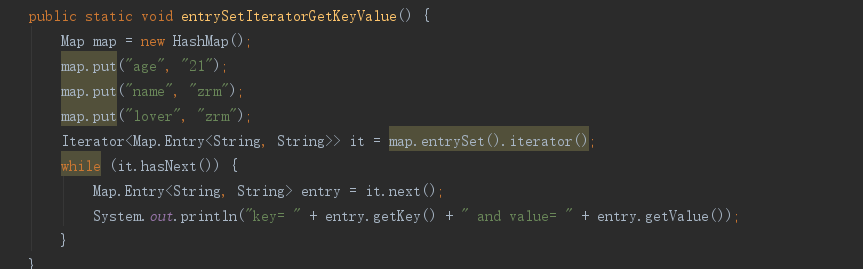


**Map遍历**

**①key Iterator循环**



**②用Map.entrySet使用iterator遍历key和value**



**③用Map.entrySet遍历key和value**