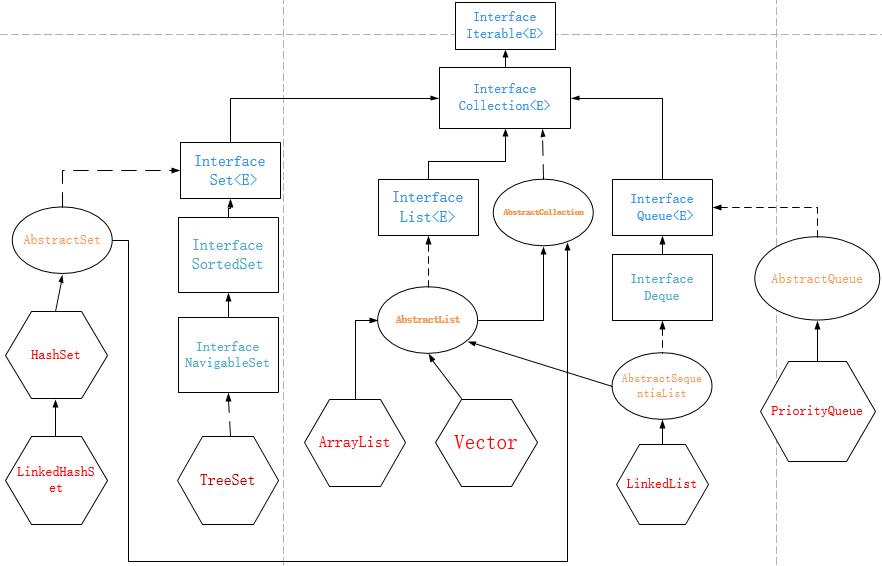
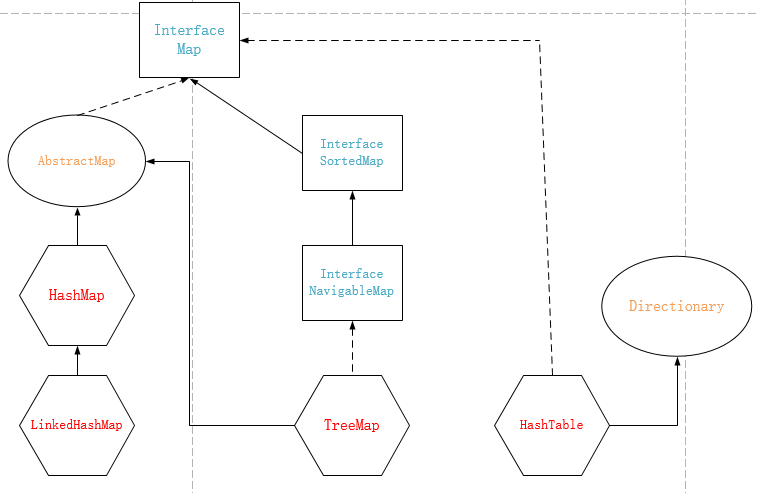
1.**collectin框架总览**



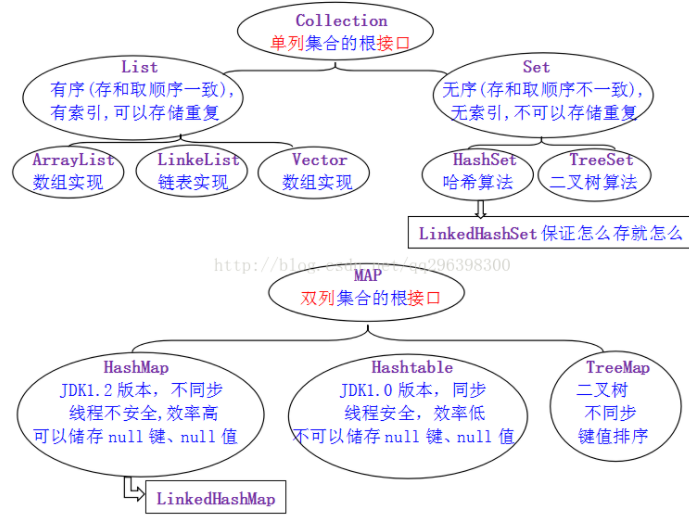
上图中，接口我用**蓝色字体矩形框**标示，抽象类用**橙色椭圆形框**标示，最终的实体类用**红色六边形标示**。另外，虚线标示实现（implements），实线标示继承（extends）。

Collection继承了Iterable接口，并且有Set，List，Queue，AbstractCollection子接口（一般不去重点解读）。

2.**Map框架**



Map接口有SortedMap子接口，AbstractMap抽象类，以及HashTable实现类



3.**List簇解读**

**3.1关于ArrayList和Vector的区别（**

①ArrayList和Vector底层都是数组实现的，初始容量为10；**在ArrayList的底层，是通过DEFAULT\_CAPACITY的常量来指定的，而Vector的底层，是直接在空构造器中，通过写死了一个this(10)来指定的。**

②**Vector大部分底层方法都是加了synchronized关键词**，所以Vector是线程同步的，而ArrayList不是；（**这里可以插入讲一下synchronized的底层实现，或者synchronized与lock的区别和联系**）

③**Vector比ArrayList多了一个构造方法，public Vector(int initialCapacity, int capacityIncrement)，初始化容量和增长容量，用于扩容。**ArrayList扩容是按照**old+old>>1**实现的，表示**增长一半容量**，但是位运算相对于普通运算较快。Vector默认是**增长一倍容量**，或者如果**增长容量大于0**，按**增长容量扩容**。

**3.2关于ArrayList和LinkedList的区别与联系**

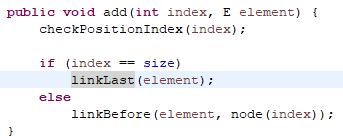
①ArrayList是实现了基于动态数组的数据结构，LinkedList是基于链表的数据结构，LinkedList底层是采用双向链表实现的，有个头尾指针；

②两者都不是同步的，如果多线程同时访问的时候，必须**保持外部同步**，用**Collections.synchronizedList(**[**Collection**](https://blog.csdn.net/qq296398300/article/details/53082686)**<T> c)来“包装”该列表。**

③对于随机访问get和set，ArrayList优于LinkedList，**因为数组本身就可以根据索引值，来随机访问元素，而双向链表只能通过遍历实现，不过LinkedList对其稍微做了优化，index<size>>1，从前半部分开始找，否则从后半部分开始找。**

④从源码来看，ArrayList想要在指定位置插入或删除元素时，主要**耗时的是System.arraycopy动作**，**会移动index后面的所有元素**，LinkedList主要耗时是**在找到index**，然后直接插入或删除，两者不一定谁快谁慢，一般说LinkedList快一些

结论：在插入或删除时，如果数据量小，LinkedList会比较快，但是在20w条以上的数据之后，LinkedList的速度就比ArrayList慢了。

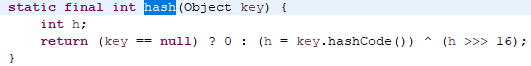


原因，因为在LinkedList在插入的时候，判断index是否等于size，等于的话插入较快，如果不等，就要先查找index对应的节点，采用前半部分和后半部分查找，这个效率就相对比较慢了。

**4.HashMap的底层实现原理**

**4.1总体介绍**

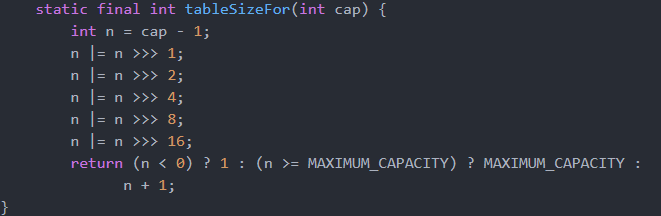
HashMap实现了Map接口，允许存储一个key为null的entry，但是允许存储多个value为null的数据（为什么可以，存入null的键过程是什么）；



**因为HashMap获取hash值的方法是hash（key），当key为null时，返回零；因此key为null的entry要么不存在table数组中，要么就在table[0]的位置。如果当前table[0]的值为null，直接将key，value设为table[0],否则就在链表或者红黑树中找，找到就更新值，没找到就插入进去。**

Hashtable与HashMap很相似，但是Hashtable方法基本是synchronized修饰的，但是Hashtable不允许存储null值和null键。**因为hashtable获取hash值是通过key.hashCode函数，如果key为空，直接空指针，另外如果value为空，会进行判空，也抛出空指针异常**

**4.2相关源码（不详细介绍）**



根据初始容量，通过上面的函数寻找大于cap且最接近的2^n的值，例如7，返回8，当table的容量为0的时候，第一次扩容是按照threshold来的，后面就是一直\*2；因此table的长度总是2^n；为什么要让table长度为2^n，因为如果需要均匀的分布元素的位置，需要对hash%length；这样只要hash函数够平均即可，但是由于hash%length效率低，采用hash%length-1，这个前提是length一定要是2^n，才能达到减少碰撞的效果。

**首先将他们的数据结构，初始化，增删改查，扩容，然后总结区别，简述一个流程，从初始化到删除**

