**集合工具类的使用特性和使用场景**

Collection（集合）：

1. 可以动态保存多个对象，使用比较方便；
2. 提供了一系列的操作对象的方法：add、remove、set、get等。

collection接口实现类的特点：

1. collection实现子类可以存放多个元素，每个元素可以是Object；
2. 有些collection的实现类可以存放重复元素，有些不可以；
3. 有些collection的实现类是有序的（list），有些是无序的（Set）；
4. collection接口没有直接的实现子类，是通过它的子接口Set和List来实现的。

collection接口遍历元素的方式：

1. 使用Iterator（迭代器）
2. Iterator对象称为迭代器，主要用于遍历collection集合中的元素；
3. 所有实现了collection接口的集合类都有一个iterator()方法，用以返回一个实现了iterator接口的对象，即可以返回一个迭代器；
4. Iterator仅用于遍历集合，其本身不存放对象。

Iterator的执行原理：

Iterator iterator = coll.iterator();//得到一个集合的迭代器

//hasNext;判断是否还有下一个元素

While(iterator.hasNext()){

//next()作用：1.下移 2.将下移后的集合位置上的元素返回

System.out.println(iterator.next());

}

1. 增强for循环

for(元素类型 元素名：集合名或数组名){

访问元素

}

List接口（是Collection接口的子接口）

1. List集合类中元素有序（即添加顺序和取出顺序一致）且可重复；
2. List集合中的每个元素都有其对应的顺序索引，即支持索引；
3. List容器宏的元素都对应一个整数型的序号记载其在容器中的位置，可以根据序号存取容器中的元素；
4. 常用的List接口实现类有：ArrayList、LinkedList和Vector

List接口的常用方法：

1. add(int index, Object ele);//在index位置插入ele元素；
2. Boolean addAll(int index, Collection eles);//从index位置开始将eles中的所有元素添加进来；
3. Object get(int index);//获取指定Index位置的元素；
4. Int indexOf(Object obj);//返回obj在集合中首次出现的位置；
5. Int lastIndexOf(Object obj);//返回obj在当前集合中末次出现的位置
6. Object remove(int index);//溢出指定index位置的元素，并返回此元素
7. Object set(int index, Object ele);//设置指定index位置的元素为ele，相当于替换；
8. List subList(int fromIndex, int toIndex);//返回从fromIndex到toIndex位置的子集合

List的三种遍历方式

1. 使用Iterator迭代器循环遍历；
2. 使用增强for循环遍历；
3. 使用普通for循环遍历。

ArrayList结构：

1. permits all elements，including null，ArrayList可以加入null并且多个加入；
2. ArrayList是由数组来实现数据存储的；
3. ArrayList基本等同于Vector，除了ArrayList是线程不安全（执行效率高）；在多线程情况下，ArrayList可能不太适合；
4. ArrayList中维护了一个Object类型的数组elementData；
5. 当创建ArrayList对象时，如果使用的是无参构造器，则初始elementData容量为0，第1次添加则扩容elementData为10，如需要再次扩建，则扩容elementData为1.5倍；
6. 如果使用的是指定大小的构造器，则初始elementData容量为指定大小，如果需要扩容，则直接扩容elementData为1.5倍。

Vector结构：

1. Vector底层也是一个对象数组，protected Object[] elementData；
2. Vector是线程同步的，即线程安全，Vector类的操作方法带有synchronized；
3. 在开发中，需要线程同步安全时，考虑使用Vector。

LinkedList结构：

1. LinkedList底层实现了双向链表和双端队列的特点；
2. 可以添加任意元素（元素可以重复），包括null；
3. 线程不安全，没有实现同步。
4. LinkedList底层维护了一个双向链表；
5. LinkedList中维护了两个属性first和last分别指向首节点和尾节点；
6. 每个结点（Node对象），里面又维护了prev、next、item三个属性，其中通过prev指向前一个，通过next指向后一个节点，最终实现双向链表；
7. LinkedList的元素的添加和删除不是通过数组完成的，相对来说效率较高；

LinkedList和ArrayList的比较：

1. 如果改查比较多，选择ArrayList；
2. 如果增删比较多，选择LinkedList；
3. 一般来说，在程序中，绝大部分都是查询，因此大部分情况下会选择ArrayList。

Set接口

1. 无序（添加和取出的顺序不一致），没有索引；
2. 不允许重复元素，所以最多包含一个null；
3. 和List接口一样，Set接口也是Collection的子接口，因此常用方法和Collection接口一样；
4. 以Set接口的实现类HashSet；
5. Set接口的实现类的对象(Set接口对象)，不能存放重复的元素，可以添加一个null；
6. Set接口对象存放数据是无序（即添加的顺序和取出的顺序不一致）。

HashSet实现类

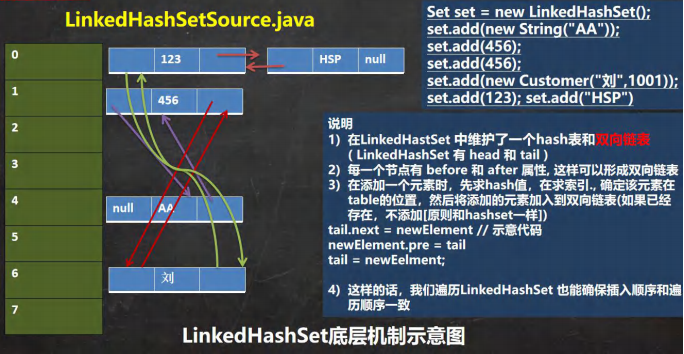
1. HashSet实现了Set接口；
2. HashSet实际上是HashMap；
3. 利用存放null值，但是只能有一个null；
4. HashSet不保证元素是有序的，取决于hash后，再确定索引的结果；
5. 不能有重复元素/对象。

HashSet添加元素原理：

1. HashSet底层是HashMap；
2. 添加一个元素时，先得到hash值-转成-索引值；
3. 找到存储数据表table，看这个索引位置是否已经存放元素；
4. 如果没有则直接加入；
5. 如果有，调用equals比较，如果相同就放弃添加，如果不相同则添加到最后；

LinkedHashSet接口

1. LinkedHashSet是HashSet的子类；
2. LinkedHashSet底层是一个LinkedHashMap，其底层维护了一个数组+双向链表；
3. LinkedHashSet根据元素的hashCode值来决定元素的存储位置，同时使用链表来维护元素的次序，使得元素看起来是以插入顺序保存的；
4. LinkedHashSet不允许添加重复元素。

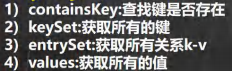


Map接口实现类：

1. Map与Collection并列存在，用于保存具有映射关系的数据Key-Value；
2. Map中的Key和value可以是任何引用类型的数据，会封装到HashMap$Node对象中；
3. Map中的key不允许重复；
4. Map中的value可以重复；
5. Map中的Key可以为null、value也可以为null，注意key为null时只能有一个，value为null可以有多个；
6. 常用String类作为Map的key；
7. key和value之间存在单向一对一关系，即通过指定的key总能找到对应的value；
8. Map存放数据的key-value中，一对key-value是放在一个HashMap$Node中的。

Map接口的常用方法：

1. put()：根据k-v添加元素；
2. remove()：根据键删除映射关系；
3. get()：根据键获取值；
4. size()：获取元素个数；
5. isEmpty()：判断个数是否为0；
6. clear()：清除k-v；
7. containsKey()：查找键是否存在。



HashMap扩容机制：

1. HashMap底层维护了Node类型的数组table，默认为null；
2. 当创建对象时，将加载因子（loadfactor）初始化为0.75；
3. 当添加key-value时，通过key的哈希值得到在table的索引。然后判断该索引处是否有元素，如果没有元素直接添加或该索引处有元素，则继续判断该元素的key和准备加入的key是否相等，如果相等则直接替换value；如果不相等则需要判断是树结构还是链表结构，做出相应的处理；如果添加时发现容量不够则需要扩容；
4. 第1次添加则需要扩容table容量为16，临界值（threshold）为12（16\*0.75）；
5. 以后再扩容则需要扩容table容量为原来是2倍（32），临界值为原来的2倍即24，以此类推。

HashTable结构：

1. 存放的元素是键值对：即k-v；
2. HashTable的键和值都不能为null，否则会抛出NullPointerException；
3. HashTable使用方法基本上和HashMap一样；
4. HashTable是线程安全的（synchronized），HashMap是线程不安全的。

Properties实现类：

1. Properties类继承自Hashtable类并且实现了Map接口，也是使用一种键值对的形式来保存数据；
2. 其使用特点和HashTable类似；
3. Properties还可以用于从xxx.properties文件中加载数据到Properties类对象，并进行读取和修改。

Collection工具类：

排序操作：

1. reverse(List)：反转List中元素的顺序；
2. shuffle(List)：对List集合元素进行随机排序；
3. sort(List)：根据元素的自然顺序指定List集合元素按照升序排列；
4. sort(List, Comparator)：根据指定的Comparator产生的顺序对List集合元素进行排序；
5. swap(List, int, int)：将指定List集合中的i处元素和j处元素进行交换。
6. Object max(Collection)：根据元素的自然顺序，返回给定集合中的最大元素
7. Object max(Collection, Comparator)：根据Comparator指定的顺序，返回给定集合中的最大元素；
8. Object min(Collection)；
9. Object min(Collection, Comparator)；
10. Int frequency(Collection, Object)：返回指定集合中指定元素的出现次数；
11. Void copy(List dest, List src)：将src中的内容复制到dest中；
12. Boolean replaceAll(List list, Object oldVal, Object newVal)：使用新值替换List对象的所有旧值。