# 数组和集合

Java中的集合类包括数组和集合。

数组，可以存基本数据类型也可以存对象，但是数组的长度是固定的，在需要存储的数据量不确定的时候不能使用数组；

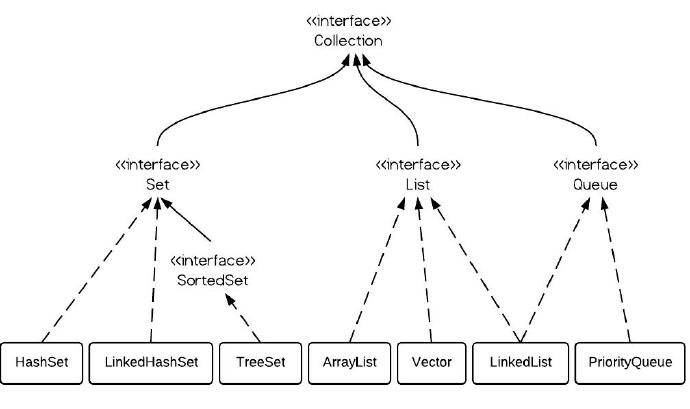
集合，只能存储对象不能存储基本数据类型，长度可变。

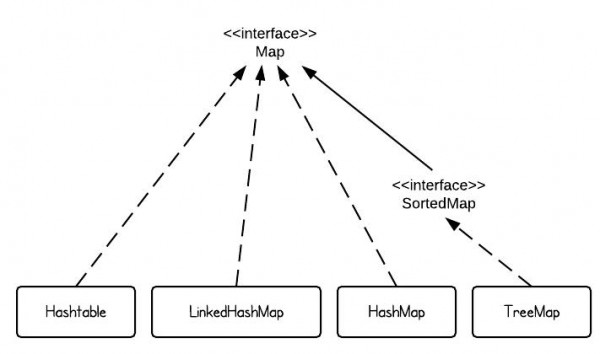
Java.util.Collection是一个接口，它提供了对集合对象进行基本操作的通用接口方法。Collection接口的意义是为各种具体的集合提供了最大化的统一操作方式。

java.util.Collections 是一个包装类。它包含有各种有关集合操作的静态多态方法。此类不能实例化，就像一个工具类，服务于Java的Collection框架。

# 接口

Collection和Map接口结构图如下：





Collection接口是集合类的根接口，Java中没有提供这个接口的直接的实现类。但是却让其被继承产生了两个接口，就是Set和List。Set中不能包含重复的元素。List是一个有序的集合，可以包含重复的元素，提供了按索引访问的方式。

Map是Java.util包中的另一个接口，它和Collection接口没有关系，是相互独立的，但是都属于集合类的一部分。Map包含了key-value对。Map不能包含重复的key，但是可以包含相同的value。

Iterator，所有的集合类，都实现了Iterator接口，这是一个用于遍历集合中元素的接口，主要包含以下三种方法：

1.hasNext()是否还有下一个元素。

2.next()返回下一个元素。

3.remove()删除当前元素。

# 几种重要的接口和类简介

1、List（有序、可重复）

List里存放的对象是有序的，同时也是可以重复的，List关注的是索引，拥有一系列和索引相关的方法，查询速度快。因为往list集合里插入或删除数据时，会伴随着后面数据的移动，所以插入删除数据速度慢。

2、Set（无序、不能重复）

Set里存放的对象是无序，不能重复的，集合中的对象不按特定的方式排序，只是简单地把对象加入集合中。

3、Map（键值对、键唯一、值不唯一）

Map集合中存储的是键值对，键不能重复，值可以重复。根据键得到值，对map集合遍历时先得到键的set集合，对set集合进行遍历，得到相应的值。

# 遍历

在类集中提供了以下四种的常见输出方式：

1）Iterator：迭代输出，是使用最多的输出方式。

2）ListIterator：是Iterator的子接口，专门用于输出List中的内容。

3）foreach输出：JDK1.5之后提供的新功能，可以输出数组或集合。

4）for循环

代码示例如下：

for的形式：for（int i=0;i<arr.size();i++）{...}

foreach的形式： for（int　i：arr）{...}

iterator的形式：

Iterator it = arr.iterator();

while(it.hasNext()){ object o =it.next(); ...}

# Map集合

实现类：HashMap、Hashtable、LinkedHashMap和TreeMap

HashMap

HashMap是最常用的Map，它根据键的HashCode值存储数据，根据键可以直接获取它的值，具有很快的访问速度，遍历时，取得数据的顺序是完全随机的。因为键对象不可以重复，所以HashMap最多只允许一条记录的键为Null，允许多条记录的值为Null，是非同步的

Hashtable

Hashtable与HashMap类似，是HashMap的线程安全版，它支持线程的同步，即任一时刻只有一个线程能写Hashtable，因此也导致了Hashtale在写入时会比较慢，它继承自Dictionary类，不同的是它不允许记录的键或者值为null，同时效率较低。

ConcurrentHashMap

线程安全，并且锁分离。ConcurrentHashMap内部使用段(Segment)来表示这些不同的部分，每个段其实就是一个小的hash table，它们有自己的锁。只要多个修改操作发生在不同的段上，它们就可以并发进行。

LinkedHashMap

LinkedHashMap保存了记录的插入顺序，在用Iteraor遍历LinkedHashMap时，先得到的记录肯定是先插入的，在遍历的时候会比HashMap慢，有HashMap的全部特性。

TreeMap

TreeMap实现SortMap接口，能够把它保存的记录根据键排序，默认是按键值的升序排序（自然顺序），也可以指定排序的比较器，当用Iterator遍历TreeMap时，得到的记录是排过序的。不允许key值为空，非同步的；

# map的遍历

第一种：KeySet()

将Map中所有的键存入到set集合中。因为set具备迭代器。所有可以迭代方式取出所有的键，再根据get方法。获取每一个键对应的值。 keySet():迭代后只能通过get()取key 。

取到的结果会乱序，是因为取得数据行主键的时候，使用了HashMap.keySet()方法，而这个方法返回的Set结果，里面的数据是乱序排放的。

典型用法如下：

Map map = new HashMap();

map.put("key1","lisi1");

map.put("key2","lisi2");

map.put("key3","lisi3");

map.put("key4","lisi4");

//先获取map集合的所有键的set集合，keyset（）

Iterator it = map.keySet().iterator();

//获取迭代器

while(it.hasNext()){

Object key = it.next();

System.out.println(map.get(key));

}

第二种：entrySet（）

Set<Map.Entry<K,V>> entrySet() //返回此映射中包含的映射关系的 Set 视图。（一个关系就是一个键-值对），就是把(key-value)作为一个整体一对一对地存放到Set集合当中的。Map.Entry表示映射关系。entrySet()：迭代后可以e.getKey()，e.getValue()两种方法来取key和value。返回的是Entry接口。

典型用法如下：

Map map = new HashMap();

map.put("key1","lisi1");

map.put("key2","lisi2");

map.put("key3","lisi3");

map.put("key4","lisi4");

//将map集合中的映射关系取出，存入到set集合

Iterator it = map.entrySet().iterator();

while(it.hasNext()){

Entry e =(Entry) it.next();

System.out.println("键"+e.getKey () + "的值为" + e.getValue());

}

推荐使用第二种方式，即entrySet()方法，效率较高。

对于keySet其实是遍历了2次，一次是转为iterator，一次就是从HashMap中取出key所对于的value。而entryset只是遍历了第一次，它把key和value都放到了entry中，所以快了。两种遍历的遍历时间相差还是很明显的。

# 主要实现类区别小结

Vector和ArrayList

1，vector是线程同步的，所以它也是线程安全的，而arraylist是线程异步的，是不安全的。如果不考虑到线程的安全因素，一般用arraylist效率比较高。

2，如果集合中的元素的数目大于目前集合数组的长度时，vector增长率为目前数组长度的100%，而arraylist增长率为目前数组长度的50%。如果在集合中使用数据量比较大的数据，用vector有一定的优势。

3，如果查找一个指定位置的数据，vector和arraylist使用的时间是相同的，如果频繁的访问数据，这个时候使用vector和arraylist都可以。而如果移动一个指定位置会导致后面的元素都发生移动，这个时候就应该考虑到使用linklist,因为它移动一个指定位置的数据时其它元素不移动。

ArrayList 和Vector是采用数组方式存储数据，此数组元素数大于实际存储的数据以便增加和插入元素，都允许直接序号索引元素，但是插入数据要涉及到数组元素移动等内存操作，所以索引数据快，插入数据慢，Vector由于使用了synchronized方法（线程安全）所以性能上比ArrayList要差，LinkedList使用双向链表实现存储，按序号索引数据需要进行向前或向后遍历，但是插入数据时只需要记录本项的前后项即可，所以插入数度较快。

arraylist和linkedlist

1.ArrayList是实现了基于动态数组的数据结构，LinkedList基于链表的数据结构。

2.对于随机访问get和set，ArrayList觉得优于LinkedList，因为LinkedList要移动指针。

3.对于新增和删除操作add和remove，LinedList比较占优势，因为ArrayList要移动数据。 这一点要看实际情况的。若只对单条数据插入或删除，ArrayList的速度反而优于LinkedList。但若是批量随机的插入删除数据，LinkedList的速度大大优于ArrayList. 因为ArrayList每插入一条数据，要移动插入点及之后的所有数据。

HashMap与TreeMap

1、 HashMap通过hashcode对其内容进行快速查找，而TreeMap中所有的元素都保持着某种固定的顺序，如果你需要得到一个有序的结果你就应该使用TreeMap（HashMap中元素的排列顺序是不固定的）。

2、在Map 中插入、删除和定位元素，HashMap是最好的选择。但如果您要按自然顺序或自定义顺序遍历键，那么TreeMap会更好。使用HashMap要求添加的键类明确定义了hashCode()和 equals()的实现。

两个map中的元素一样，但顺序不一样，导致hashCode()不一样。

同样做测试：

在HashMap中，同样的值的map,顺序不同，equals时，false;

而在treeMap中，同样的值的map,顺序不同,equals时，true，说明，treeMap在equals()时是整理了顺序了的。

HashTable与HashMap

1、同步性:Hashtable是线程安全的，也就是说是同步的，而HashMap是线程序不安全的，不是同步的。

2、HashMap允许存在一个为null的key，多个为null的value 。

3、hashtable的key和value都不允许为null。

