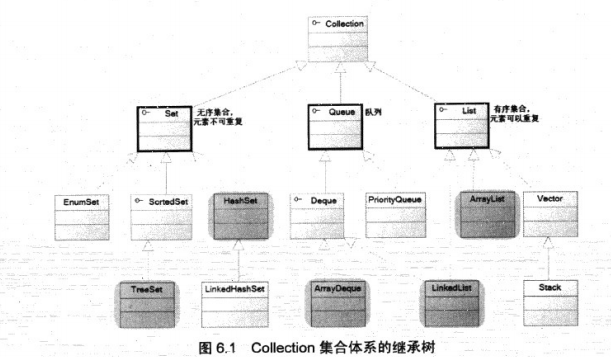
**Java集合**

java集合大致可以分为set、list、map三个体系，set代表无序不可重复的集合，list代表有序可重复的集合，map代表具有映射关系的集合。java5之后新增了queue集合，代表队列的集合。数组里能存放基本类型和对象，而集合只能存放对象。

java的所有集合类都派生自collection和map两个接口。

# 1、Collection接口

## 1.1、Collection接口体系继承树



## 1.2、Collection常用方法

boolean add(Object o): 向集合中新增一个元素

void clear(): 清除集合中的所有元素

boolean contains(Object o): 集合中是否包含某个元素

boolean isEmpty(): 集合是否为空

Iterator iterator(): 返回集合的Iterator对象

boolean remove(Object o): 在集合中删除指定对象

int size(): 返回集合大小

Object[] toArray(): 把集合转换成一个数组

## 1.3、访问集合中的元素

### 1.3.1、使用Iterator接口迭代集合

Iterator主要用于遍历集合中的元素，Iterator对象也被叫做迭代器。Iterator只用于迭代集合并不存放数据，所以如果要得到Iterator对象就先需要一个集合对象。

常用方法：

boolean hasNext(): 如果被迭代的集合元素还没有被遍历就返回true

Object next(): 返回集合里的下一个元素

void remove(): 删除集合中遍历的上一个元素

### 1.3.2、使用foreach遍历集合

示例代码：

public class Test6 {

public static void main(String[] args){

Collection<String> books = new ArrayList<String>();

books.add("book1");

books.add("book2");

books.add("book3");

for(String tempBookName : books){

System.out.println("tempBookName = " + tempBookName);

}

}

}

# 2、Set接口

Set集合和Collection集合基本上完成一样，Set判断两个对象相同不是使用==，而是使用equals方法，只要两个元素的equals方法返回值相同，那么set就认为是两个相同的元素。

## 2.1、HashSet

HashSet类是Set接口的实现类，该类按照hash算法来存储元素，因此具有很好的存取和查找性能。

特点：

（1）不能保证元素的顺序

（2）线程不安全

（3）集合元素可以是null

HashSet集合判断两个元素是否相等的依据是两个对象通过equals方法是否相等，还有两个对象的hashCode方法返回值是否相等。

如果重写对象的equals方法，那么也应该重写hashCode方法。

如果HashSet集合中有两个以上的元素的hashCode相等，那么就会影响性能。

## 2.2、LinkedHashSet

LinkedHashSet是HashSet的子类，使用链表的方式维护元素的顺序，所以性能比HashSet略差一点，该类中也不允许存在重复的元素。

## 2.3、TreeSet

TreeSet是SortedSet接口的实现类，TreeSet确保集合元素处于排序状态。

常用方法：

Comparator comparator(): 如果集合采用自定义的排序，那么返回自定义的Comparator对象，否则返回null.

Object first(): 返回集合中的第一个元素

Object last(): 返回集合中的最后一个元素

## 2.4、各Set实现类的性能分析

HashSet比TreeSet性能好，特别是添加、查询等操作，因为TreeSet需要额外的红黑树算法来维护元素的次序。

对于普通的插入、删除操作，LinkedHashSet要比HashSet慢一点，因为LinkedHashSet需要链表维护元素的次序。对于遍历操作，LinkedHashSet性能会更好一点。

EnumSet是所有set集合中性能最好的，但是元素只能存放枚举类的枚举值。

以上几个集合都是线程不安全的。

# 3、List集合

List集合代表一个有序可重复的集合。

List通过equals方法判断两个对象是否相同。

## 3.1、ListIterator接口

ListIterator接口继承了Iterator接口，具有Iterator的所有方法，ListIterator还增加了向前迭代的功能。

新增方法如下：

boolean hasPrevious(): 返回关联的集合是否还有上一个元素

Object previous(): 返回上一个元素

void add(Object o): 在指定位置插入一个元素

## 3.2、ArrayList和Vector

ArrayList和Vector类都是基于数组实现的类，初始化的数组长度是10。

ArrayList和Vector基本上是一样的，ArrayList是线程不安全的，Vector是线程安全的，Vector比较古老，存在很多问题，所以不建议用。

## 3.3、LinkedList

LinkedList是基于链表的list接口的实现类，所以对于插入和删除操作速度非常快。LinkedList类还实现了Deque接口，可以作为栈来用。

LinkedList与ArrayDeque、ArrayList的实现机制不同，ArrayDeque和ArrayList底层是通过数组的方式来存放元素的，因此随机访问集合元素性能比较好，而LinkedList内部已链表的形式存放集合元素，因此随机访问性能较差，但插入、删除等操作性能较好，遍历时最好用迭代器迭代。

# 4、Queue接口

Queue是一个队列集合，是一个先进先出的容器。

常用方法：

void add(o): 插入一个元素

Object element(): 获取队列的头部元素，但是不删除

boolean offer(Object o): 向队列尾部插入元素，比add方法更好

Object poll(): 获取队列的头部元素，并删除，如果队列空就返回null

Object remove(): 获取队列的头部元素，并删除

## 4.1、Deque接口

Deque接口是Queue接口的子接口，底层数组长度是16，是一个双端队列。

常用方法：

void addFirst(Object o): 将指定元素插入队列的开头

void addLast(Object o): 将指定元素插入队列尾部

Iterator descendingIterator(): 返回双端队列的迭代器

Object getFirst(): 获取双端队列的第一个元素

Object getLast(): 获取双端队列的最后一个元素

boolean offerFirst(Object o): 将指定元素插入双端队列的开头

boolean offerLast(Object o): 将指定元素插入双端队列的尾部

Object poolFirst(): 获取双端队列的一个元素并从队列中删除

Object poolLast(): 获取双端队列的最后一个元素并从队列中删除

Object pop(): 栈方法，取出栈顶的元素，相当于removeFirst()

void push(Object o): 栈方法，向栈顶中放入元素，相当于addFirst方法

### 4.1.1、ArrayDeque

ArrayDeque是Queue接口的实现类，底层数组初始化大小为16，底层采用一个动态的，可重分配的Object[]数组来存储集合元素。当集合元素超出了数组大小后，系统会重新分配一个Object[]数组来存放集合元素。

ArrayDeque的性能比Stack更好，所以程序中如果需要用到栈的数据结构，可以用ArrayDeque或者LinkedList，而不是Stack。

# 5、Map接口

Map用于保存具有映射关系的数据，Map中的key不允许重复，key和value之间的关系总是一一对应的，map中有keyset方法用于返回所有key的set集合。Map提供了一个Entry的内部类用于存放key-value键值对。

常用方法：

void clear(): 删除map中的所有元素

boolean containsKey(Object o)：查询map中是否有指定的key

boolean containsValue(Object o): 查询map中是否有指定的一个或多个value

Set entrySet(): 返回map中key-value组成的set集合

Object get(Object key): 获取指定key的value

boolean isEmpty(): map是否为空

Set ketSet(): 返回map中所有key组成的set集合

object put(object key,object value): 向map中插入数据

bolean remove(object key): 从map中删除指定key的元素

Collections values(): 返回该map里所有value组成的collection集合

## 5.1、HashMap和Hashtable

HashMap和Hashtable都是Map的实现类，Hashtable从jdk 1.0开始就有了，是线程安全的,Hashtable中不允许有null值的存在，HashMap中可以有null。

判断两个key是否相等，也要看key对象的equals方法和hashcode方法是否相等。

当使用自定义的类作为key值时，如果重写了该类的equals和hashCode方法，则应该保证两个方法的判断标准是一致的。

### 5.1.1、LinkedHashMap

LinkedHashMap是HashMap的子类，使用双向链表来维护key-value对的次序，该链表维护迭代的次序。LinkedHashMap需要维护元素的插入顺序，所以性能低于HashMap。