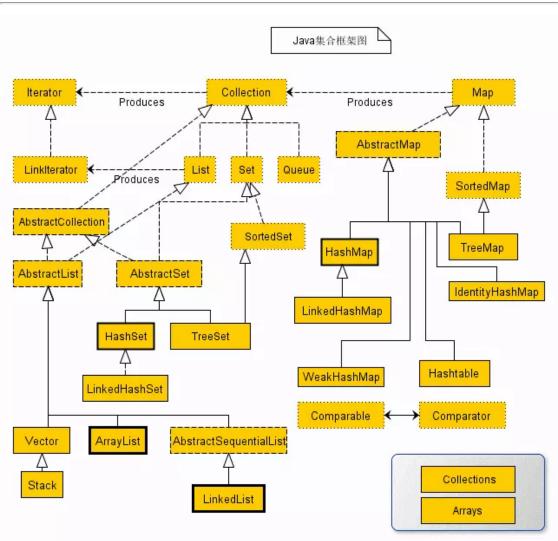
java集合框架概述

笔记本: java基础

创建时间: 2019/5/3 23:41 **更新时间:** 2019/5/4 0:01

作者: ANNER

URL: https://www.jianshu.com/p/63e76826e852



Java集合框架主要包括两种类型的容器,一种是集合(Collection),存储一个元素集合,另一种是图(Map),存储键/值对映射。Collection接口又有3种子类型,List、Set和Queue,再下面是一些抽象类,最后是具体实现类,常用的有ArrayList、LinkedList、HashSet、LinkedHashMap、LinkedHashMap等等。

Collection接口

Collection接口是处理对象集合的根接口,其中定义了很多对元素进行操作的方法, AbstractCollection是提供Collection部分实现的抽象类。下图展示了Collection接口中的 全部方法。

```
✓ O Collection < E >

    A size(): int

       A isEmpty(): boolean
       A contains(Object) : boolean
       A toArray() : Object[]
       A toArray(T[]) <T>:T[]
       A add(E): boolean
       A remove(Object) : boolean
       A containsAll(Collection<?>): boolean
       A addAll(Collection <? extends E>): boolean
       A removeAll(Collection<?>): boolean
       Premovelf(Predicate <? super E>): boolean
       A retainAll(Collection <?>): boolean
       A clear(): void
       A equals(Object) : boolean
       A hashCode(): int
       Spliterator(): Spliterator < E >
       D stream(): Stream < E >
       D parallelStream(): Stream<E>
```

其中,有几个比较常用的方法,比如方法add()添加一个元素到集合中,addAll()将指定集合中的所有元素添加到集合中,contains()方法检测集合中是否包含指定的元素,toArray()方法返回一个表示集合的数组。Collection接口有三个子接口,下面详细介绍。

List

List接口扩展自Collection,它可以定义一个允许重复的有序集合,从List接口中的方法来看,List接口主要是增加了面向位置的操作,允许在指定位置上操作元素,同时增加了一个能够双向遍历线性表的新列表迭代器ListIterator。AbstractList类提供了List接口的部分实现,AbstractSequentialList扩展自AbstractList,主要是提供对链表的支持。下面介绍List接口的两个重要的具体实现类,也是我们可能最常用的类,ArrayList和LinkedList

- ArrayList:通过阅读ArrayList的源码,我们可以很清楚地看到里面的逻辑,它是用数组存储元素的,这个数组可以动态创建,如果元素个数超过了数组的容量,那么就创建一个更大的新数组,并将当前数组中的所有元素都复制到新数组中
- LinkedList 是在一个链表中存储元素。所以,LinkedList的元素添加和删除其实就对应着链表节点的添加和移除。

Set接口扩展自Collection,它与List的不同之处在于,规定Set的实例不包含重复的元素。 AbstractSet是一个实现Set接口的抽象类,Set接口有三个具体实现类,分别是散列集 HashSet、链式散列集LinkedHashSet和树形集TreeSet。

散列集HashSet是一个用于实现Set接口的具体类,可以使用它的无参构造方法来创建空的散列集,也可以由一个现有的集合创建散列集。在散列集中,有两个名词需要关注,初始容量和客座率。实际上HashSet就是基于后面介绍的HashMap而实现的,客座率是确定在增加规则集之前,该规则集的饱满程度,当元素个数超过了容量与客座率的乘积时,容量就会自动翻倍。

查看散列集HashSet的源码实现可以看到它内部是使用一个HashMap来存放元素的,因为HashSet的元素就是其内部HashMap的键集合,所以所以HashSet可以做到元素不重复

- LinkedHashSet是继承自HashSet的,支持对规则集内的元素排序。HashSet中的元素是没有被排序的,而LinkedHashSet中的元素可以按照它们插入规则集的顺序提取
- TreeSet扩展自AbstractSet,并实现了NavigableSet, AbstractSet扩展自 AbstractCollection,树形集是一个有序的Set,其底层是一颗树,这样就能从Set里面提取一个有序序列了

Queue

队列是一种先进先出的数据结构,元素在队列末尾添加,在队列头部删除。Queue接口扩展自Collection,并提供插入、提取、检验等操作。

```
Queue < E >

A add(E): boolean

A offer(E): boolean

A remove(): E

A poll(): E

A element(): E

A peek(): E
```

接口Deque,是一个扩展自Queue的双端队列,它支持在两端插入和删除元素,因为 **LinkedList类实现了Deque接口,所以通常我们可以使用LinkedList来创建一个队列**。 PriorityQueue类实现了一个优先队列,优先队列中元素被赋予优先级,拥有高优先级的先被删除。

Map接口

Map, 图, 是一种存储键值对映射的容器类, 在Map中键可以是任意类型的对象, 但不能有重复的键, 每个键都对应一个值, 真正存储在图中的是键值构成的条目。下面是接口

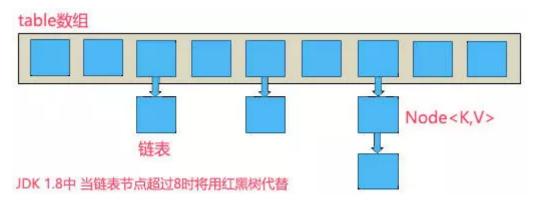
Map的类结构。

```
Map<K, V>
       A size(): int
       A isEmpty(): boolean
       A containsKey(Object) : boolean
       A containsValue(Object) : boolean
       A get(Object) : V
       A put(K, V) : V
       A remove(Object) : V
       A putAll(Map <? extends K, ? extends V >): void
       A clear(): void
       A keySet() : Set<K>
       A values() : Collection < V >
       A entrySet() : Set < Entry < K, V >>
    > OS Entry < K, V >
       A equals(Object) : boolean
       A hashCode(): int
       getOrDefault(Object, V) : V
       forEach(BiConsumer<? super K, ? super V>): void
       PreplaceAll(BiFunction<? super K, ? super V, ? extends V>): void
       D putlfAbsent(K, V): V
       o D remove(Object, Object) : boolean
       Preplace(K, V, V): boolean
       replace(K, V) : V
       ComputeIfAbsent(K, Function <? super K, ? extends V>): V
       ComputeIfPresent(K, BiFunction<? super K, ? super V, ? extends V>): V
       Compute(K, BiFunction <? super K, ? super V, ? extends V>) : V
       • merge(K, V, BiFunction <? super V, ? super V, ? extends V >) : V
```

从上面这张图中我们可以看到接口Map提供了很多查询、更新和获取存储的键值对的方法,更新包括方法clear()、put()、putAll()、remove()等等,查询方法包括 containsKey、containsValue等等。Map接口常用的有三个具体实现类,分别是 HashMap、LinkedHashMap、TreeMap

HashMap

HashMap是基于哈希表的Map接口的非同步实现,继承自AbstractMap,AbstractMap是部分实现Map接口的抽象类。在平时的开发中,HashMap的使用还是比较多的。我们知道ArrayList主要是用数组来存储元素的,LinkedList是用链表来存储的,那么HashMap的实现原理是什么呢? 先看下面这张图:



在之前的版本中,HashMap采用数组+链表实现,即使用链表处理冲突,同一hash值的链表都存储在一个链表里。但是当链表中的元素较多,即hash值相等的元素较多时,通过key值依次查找的效率较低。而JDK1.8中,HashMap采用数组+链表+红黑树实现,当链表长度超过阈值(8)时,将链表转换为红黑树,这样大大减少了查找时间

LinkedHashMap

LinkedHashMap继承自HashMap,它主要是用链表实现来扩展HashMap类,HashMap中条目是没有顺序的,但是在LinkedHashMap中元素既可以按照它们插入图的顺序排序,也可以按它们最后一次被访问的顺序排序

TreeMap

TreeMap基于红黑树数据结构的实现,键值可以使用Comparable或Comparator接口来排序。TreeMap继承自AbstractMap,同时实现了接口NavigableMap,而接口NavigableMap则继承自SortedMap。SortedMap是Map的子接口,使用它可以确保图中的条目是排好序的。

其它集合类

面主要对Java集合框架作了详细的介绍,包括Collection和Map两个接口及它们的抽象类和常用的具体实现类,下面主要介绍一下其它几个特殊的集合类,Vector、Stack、HashTable、ConcurrentHashMap以及CopyOnWriteArrayList

Vector

Java设计者们在对之前的容器类进行重新设计时保留了一些数据结构,其中就有Vector。 用法上,**Vector与ArrayList基本一致,不同之处在于Vector使用了关键字synchronized** 将访问和修改向量的方法都变成同步的了,所以对于不需要同步的应用程序来说,类 ArrayList比类Vector更高效

stack

Stack, 栈类,是Java2之前引入的,继承自类Vector。

hashtable

Hashtable和前面介绍的HashMap很类似,它也是一个散列表,存储的内容是键值对映射,不同之处在于,Hashtable是继承自Dictionary的,Hashtable中的函数都是同步的,这意味着它也是线程安全的,另外,Hashtable中key和value都不可以为null。

总结

- Java集合框架主要包括Collection和Map两种类型。其中Collection又有3种子类型, 分别是List、Set、Queue。Map中存储的主要是键值对映射。
- 规则集Set中存储的是不重复的元素,线性表中存储可以包括重复的元素,Queue队列描述的是先进先出的数据结构,可以用LinkedList来实现队列。
- 效率上,规则集比线性表更高效。
- ArrayList主要是用数组来存储元素,LinkedList主要是用链表来存储元素,HashMap的底层实现主要是借助数组+链表+红黑树来实现。
- Vector、Hashtable等集合类效率比较低但都是线程安全的。java.util.concurrent下包含了大量线程安全的集合类,效率上有较大提升。