集合

笔记本: Java

创建时间: 2021/8/12 23:18 **更新时间:** 2021/8/14 4:57

作者: Lcxuan

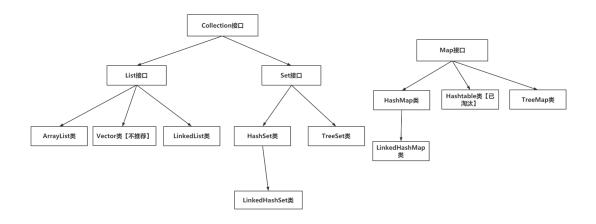
标签: ArrayList, Collection, HashMap, HashSet, LinkedList, List, TreeMap, TreeSet

URL: about:blank

集合:对多个数据进行存储操作,可以对不同类型的数据进

行存储

应用场景: 当需要将相同结构的个体整合到一起时



Collection接口

常用方法:

• 增加: add()、addAll()

• 删除: clear()【清空所有】、remove()【删除指定的元

素】

• 修改:

• 查看: iterator()、size()【获取集合的元素】

判断: contains()【判断集合是否包含指定元素】、 equals()【判断两个集合是否相等】、isEmpty()【判断 集合是否为空】

遍历方式:

• 使用增强for循环:

```
//1、增强for循环
for (Object o: col) {
   System.out.println(o);
}
```

• 使用迭代器:

```
Iterator it = col.iterator();
//通过hasNext()判断是否有下一个元素,如果有则返回True,如果没有则返回False
while (it.hasNext()){
    //next():将元素获取到,并且指针下移
    System.out.println(it.next());
}
```

List接口

List接口中的实现类都是:不唯一且有序的

常用方法:

List接口中的常用方法在Collection接口中包含

- 增加: add(E e)【添加单个元素】、add(int index, E element)【在指定位置添加元素】
- 删除: remove(Object o)【删除指定的元素】、remove(int index)【删除指定下标的元素】
- 修改: set()【修改指定位置的元素】
- 查看: get()【查找指定位置的元素】
- 判断:

遍历方式:

• 使用普通for循环:

```
for (int i = 0; i < list.size(); i++) {
   System.out.println(list.get(i));
}</pre>
```

• 使用增强for循环:

```
for (Object o: list) {
   System.out.println(o);
}
```

• 使用迭代器:

```
Iterator iterator = list.iterator();
while (iterator.hasNext()){
   System.out.println(iterator.next());
}
```

ArrayList类

- 1、ArrayList类继承List接口
- 2、底层两个重要属性:

3、构造器给elementData属性初始化为空的数组

```
**Constructs an empty list with an initial capacity of ten.

*/

public ArrayList() {
    this.elementData = DEFAULTCAPACITY_EMPTY_ELEMENTDATA;
}

* Shared empty array instance used for default sized empty instances. We

* distinguish this from EMPTY_ELEMENTDATA to know how much to inflate when

* first element is added.

*/

private static final Object[] DEFAULTCAPACITY_EMPTY_ELEMENTDATA = {};
```

4、调用add()方法

```
/**

* Appends the specified element to the end of this list.

* Sparam e element to be appended to this list

* Speturn (@code true) (as specified by (@link Collection#add))

*/

* public boolean * pdd(E e) {

modCount++;

add(e, elementData, size);

return true;

}

* This helper method split out from add(E) to keep method

* bytecode size under 35 (the -XX:MaxInlineSize default value),

* which helps when add(E) is called in a CI-compiled loop.

*/

* private void add(E e, Object[] elementData, int s) {

if (s == elementData = grow();

elementData = grow();
```

ArrayList和Vector的区别:

- 1. ArrayList线程不安全,效率高
- 2. Vector线程安全,效率不高

LinkedList类 常用方法

LinkedList类中的常用方法在List接口中包含

- 增加: addFirst()【开头添加元素】、addLast()【结尾添加元素】、offer()【尾端添加元素】、offerFirst()【开头添加元素】、offerLast()【结尾添加元素】
- 删除: poll()【删除开头第一个元素,将删除元素返回】、pollFirst()【删除开头第一个元素,将删除元素返回】、pollLast()【删除尾部第一个元素,将删除元素返回】、removeFirst()【删除开头第一个元素,将删除元素返回】、removeLast()【删除尾部第一个元素,将删除元素返回】
- 修改:
- 查找: element()、getFirst()、getLast()、indexOf()、lastIndexOf()、peek()、peekFirst()、peekLast()
- 判断:

遍历方式

• 普通for循环

```
for (int i = 0; i < linkedList.size(); i++) {
   System.out.println(linkedList.get(i));</pre>
```

• 增强for循环

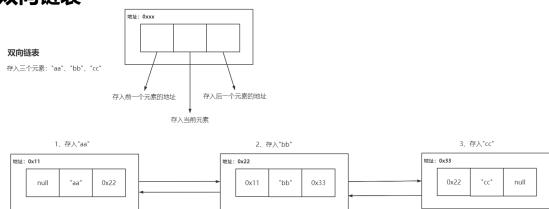
```
for (String value: linkedList ) {
   System.out.println(value);
}
```

• 迭代器

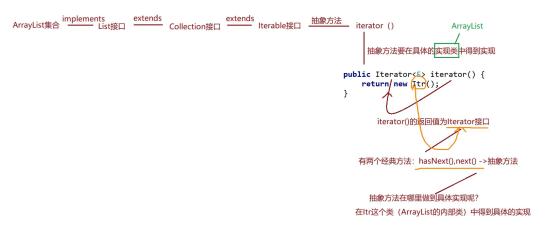
```
Iterator<String> iterator = linkedList.iterator();
while (iterator.hasNext()){
   System.out.println(iterator.next());
}
```

底层使用双向链表

双向链表



Iterator迭代器



ListIterator迭代器 常用方法

- hasNext(): 判断指针下面是否还存在值
- next(): 获取元素并且指针下移
- hasPrevious(): 判断指针上面是否还存在值
- Previous(): 获取元素并且指针上移

Set接口

Set接口中的实现类都是:唯一且无序的 Set接口中的常用方法在Collection接口中包含

遍历方式:

• 增强for循环

```
for (Object o:set) {
   System.out.println(o);
}
```

• 迭代器

```
Iterator<Object> iterator = set.iterator();
while (iterator.hasNext()){
   System.out.println(iterator.next());
}
```

HashSet类

创建集合:

```
HashSet<Integer> hashSet = new HashSet<>();
```

底层: 哈希表

注意: 如果是存入HashSet中的数据, 一定要重写 hashCode()和equals()方法

LinkedHashSet类

特点:唯一、有序,按照输入的顺序进行输出 继承HashSet

TreeSet类

特点: 唯一、无序或者有序(需要按照有序进行遍历)

底层: 二叉树

自定义的对象存入TreeSet中时,自定义的对象需要**继承**Comparable接口重写compareTo()方法 或者 使用外部的比较器,并且传入TreeSet的对象中

Map接口

特点: 无序、唯一

常用方法

• 增加: put()【添加数据】

• 删除: clear()【清空数据】、remove()【删除指定key的数据】

• 修改:

- 查看: entrySet()【返回此集合中的Set集合】、get() 【获取指定key的value】、keySet()【返回key的Set集 合】、size()【获取长度】、values()【返回value的 Collection集合】
- 判断: containsKey()【判断是否包含某个key】、containsValue()【判断是否包含某个value】、equals()【比较值】、isEmpty()【判断是否为空】

遍历方式:

• 使用keySet获取key

```
Set<String> keySet = map.keySet();
for (String key:keySet) {
   System.out.println(key);
}
```

• 使用values()获取value

```
Collection<Integer> values = map.values();
for (Integer integer:values) {
   System.out.println(integer);
}
```

使用entrySet

```
Set<Map.Entry<String, Integer>> entries = map.entrySet();
for (Map.Entry<String, Integer> entry : entries){
    System.out.println("获取的key:" + entry.getKey());
    System.out.println("获取的value: " + entry.getValue());
}
```

HashMap类

特点: 无序、唯一

底层: 哈希表

效率高,线程不安全,可以存入null,并且null也是唯一的

LinkedHashMap类

特点:有序、唯一

底层: 哈希表 + 链表

继承HashMap

TreeMap类

特点: 唯一、有序 (按照升序或降序)

底层:二叉树, key遵循二叉树原理, 放入key数据的对应类型内部一定要实现比较器 (Compare 【内部比较器、外部比

较器】)