## 回顾

1. 集合顶层接口:collection接口-->list接口/set接口
2. List接口的实现类:ArrayList LinkedList Vector

掌握ArrayList 很多方法 遍历方式(3种)

ArrayList :有序的数组结构,动态数组,长度可以改变

ArrayList<指定泛型> list=new ArrayList<指定泛型>();

常用方法:

求大小:size()

添加:add([E](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \o "ArrayList 中的类型参数) element) **[add](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "add(int, E))**(int index, [E](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \o "ArrayList 中的类型参数) element)

修改:**[set](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "set(int, E))**(int index, [E](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \o "ArrayList 中的类型参数) element)   
删除:remove(int index) **[remove](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "remove(java.lang.Object))**([Object](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/lang/Object.html" \o "java.lang 中的类) o) /**[clear](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "clear())**()

查询:get(int index)/**[indexOf](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "indexOf(java.lang.Object))**([Object](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/lang/Object.html" \o "java.lang 中的类) o)/**[lastIndexOf](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "lastIndexOf(java.lang.Object))**([Object](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/lang/Object.html" \o "java.lang 中的类) o)/

**[contains](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "contains(java.lang.Object))**([Object](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/lang/Object.html" \o "java.lang 中的类) o)

判断:**[isEmpty](mk:@MSITStore:E:\\标准化教案\\帮助文档\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/util/../../java/util/ArrayList.html" \l "isEmpty())**() (size等于0)

遍历方式(3种):for(int i=0;i<list.size();i++) for(Emp emp:list)

Iterator iterator= list.iterator();

while(iterator.hasNext()) {Emp emp=iterator.next();}

|  |  |
| --- | --- |
| List接口(有序可重复) | 常见方法(add get remove size等等) |
| ArrayList类(可变长度数组) | 检索操作效率高,但是插入删除效率不高 |
| LinkedList类 可变链表结构 | 链表的实现,增加了很多方法 addFist()...插入删除效率高 |
| Vector类可变 | 和ArrayList 线程同步 |

## 作业讲解

# 泛型再说明

一、泛型的概念

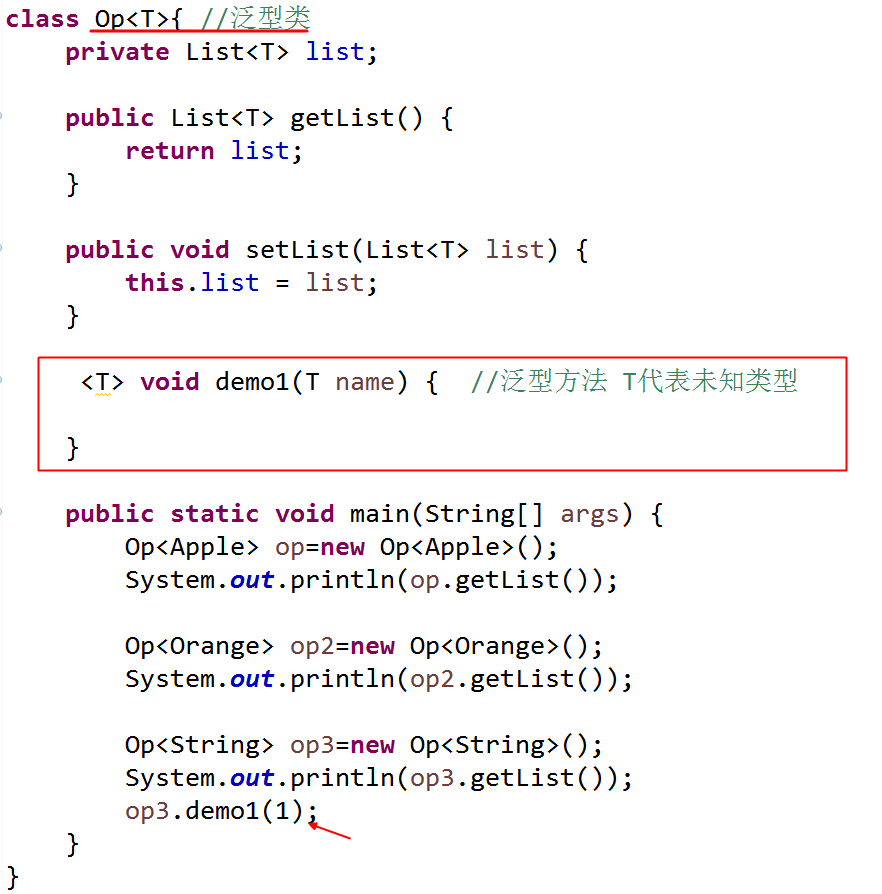
泛型的定义：泛型是JDK 1.5以后的的一项新特性，它的本质是参数化类型（Parameterized Type）的应用，也就是说所操作的数据类型被指定为一个参数，在用到的时候在指定具体的类型。这种参数类型可以用在类、接口和方法的创建中，分别称为泛型类、泛型接口和泛型方法。在Java[编程](http://www.cppentry.com/" \t "https://www.cppentry.com/_blank)思想中解释道：泛型的主要目的之一就是指定容器要持有什么类型的对象，而且由我们的编译器来保证类型的正确性。

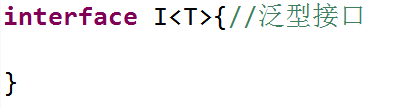
二、泛型的语法以及使用

关于泛型的使用有泛型类，泛型接口，以及泛型方法。

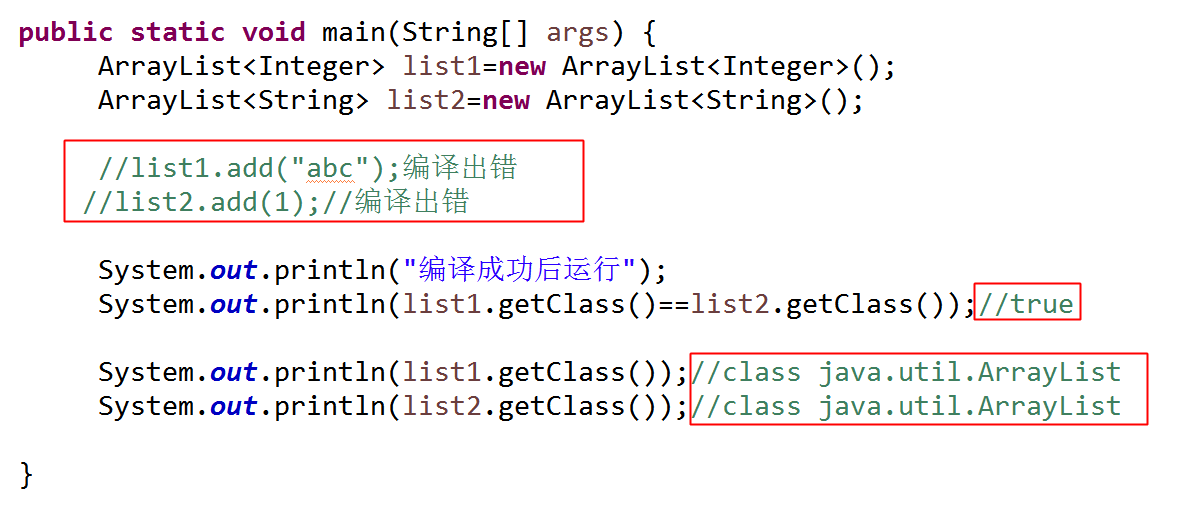
1.关于泛型类的定义以及使用介绍

关于泛型类（Generic Type）的语法定义其实也非常的简单，主要是在类名之后加上一个“<>”，然后在这个“<>”里面加上一个类型参数





1. 泛型擦除:泛型只存在于编译期间,编译完成后,自动擦除,运行期是没有泛型
2. 在写代码时,定义泛型,就无法把一个String类型的实例加到ArrayList<Integer>中,因为Integer和String在编译的时候是完全不同的类型,但是运行结果却是true,这就是java泛型擦除造成的.
3. 因为不管是ArrayList<Integer>还是ArrayList<String>,在编译完成后都会被编译器擦除成ArrayList.
4. Java泛型擦除 是java泛型中一个重要特性,其目的是避免过多的创建类而造成的运行时的过度消耗.



1. 自定义泛型

泛型常见字母

T Type 表示类型

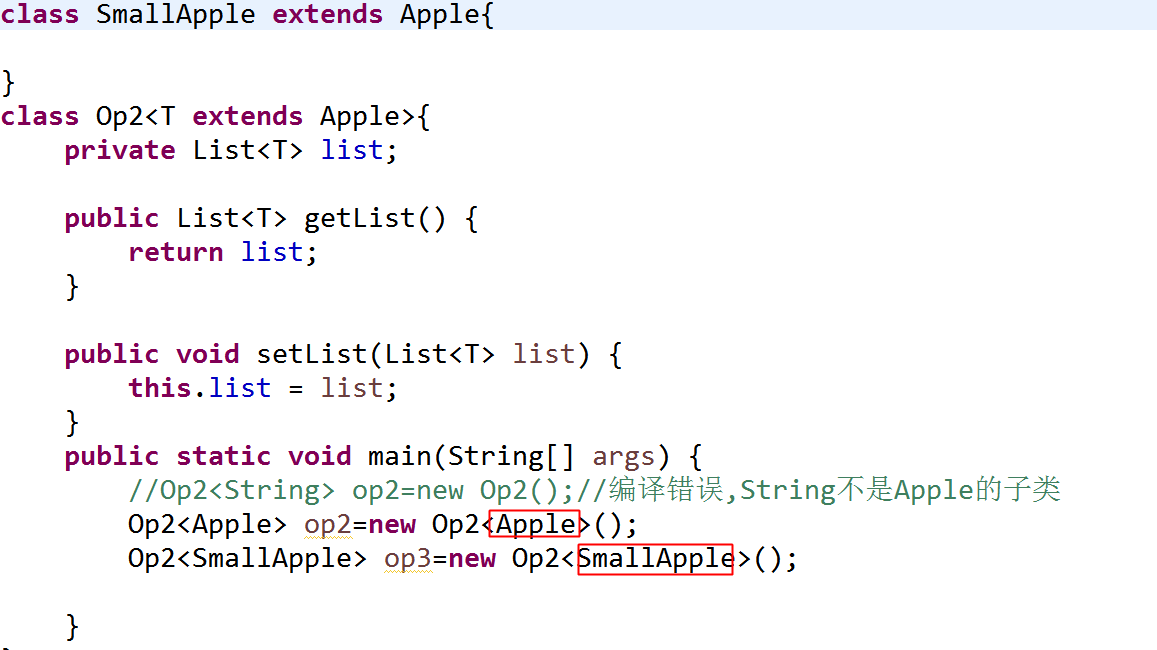
K V 分别代表键值中的Key、Value

E 代表Element

？表示不确定的类型

<? extends T>

类型上界. ?表示的是此处传入的类型必须是 T类型本身或者其子类.如果这里T为Number的话,那么我们只能传入Number本身或者子类,比如Double、Float、Integer、Long 等基本数据类型的包装类.

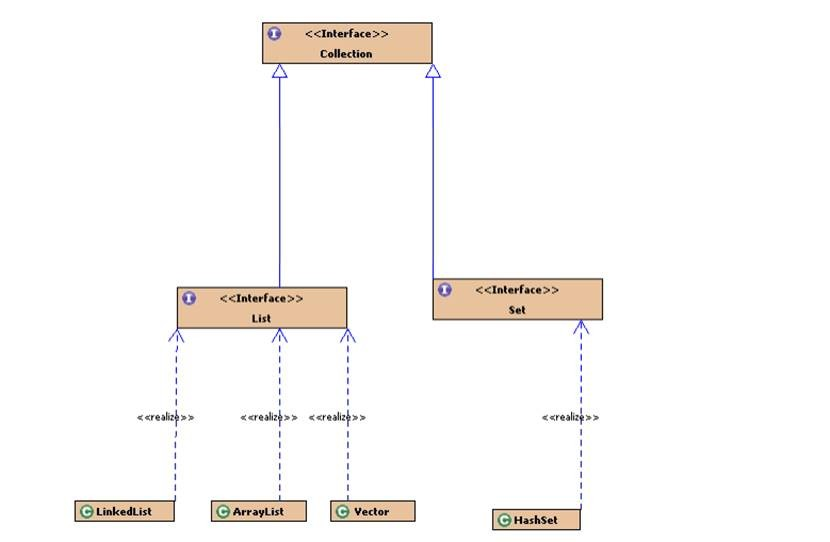


<? super T>

类型下界 ?表示的是此处传入的类型必须是 T类型本身或者其超(父)类.如果这里T为Integer的话,那么我们只能传入Integer本身或者超类,比如Number类.

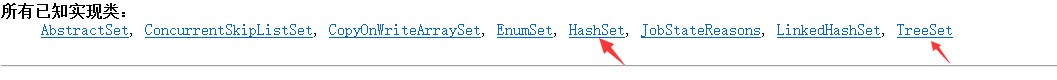
# set接口的介绍

Collection层次结构图:

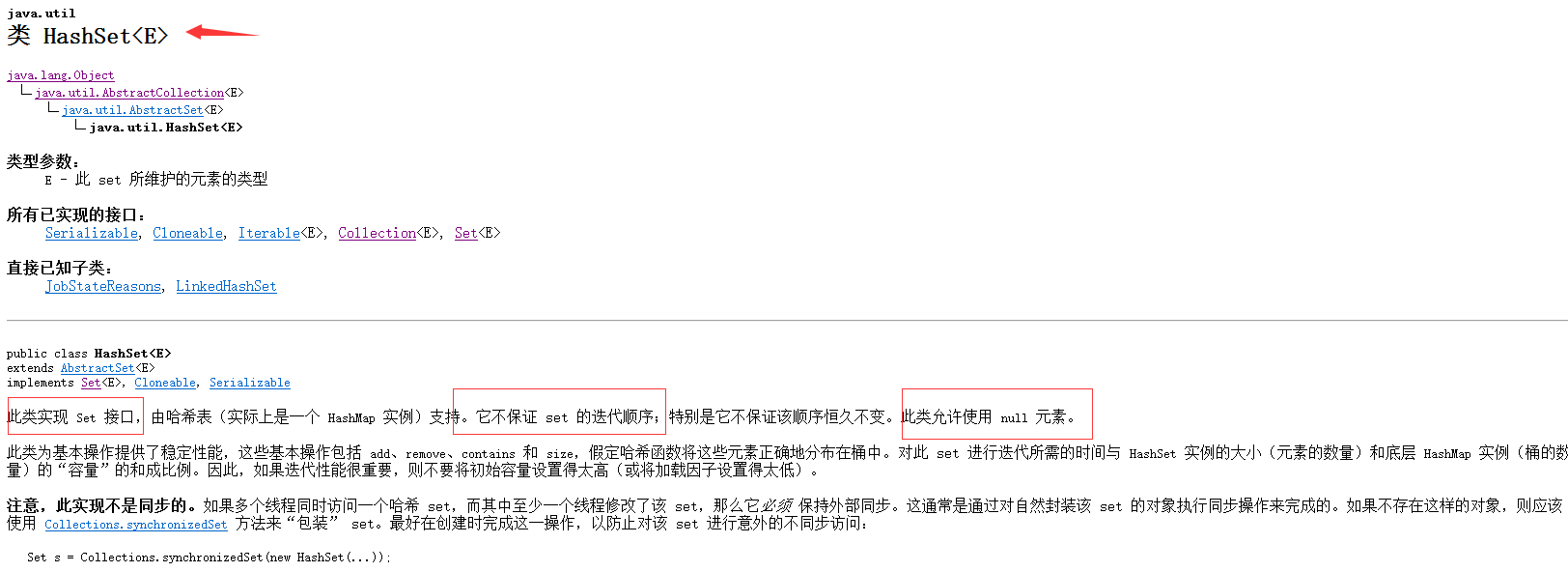


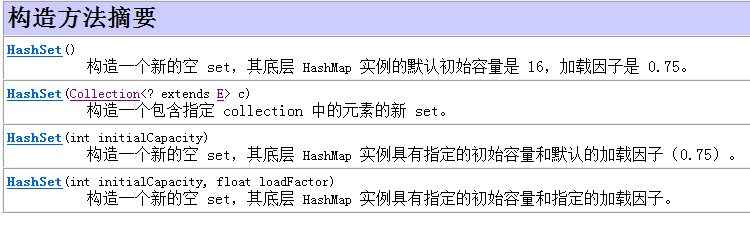
层次结构要熟记;各自的特点;





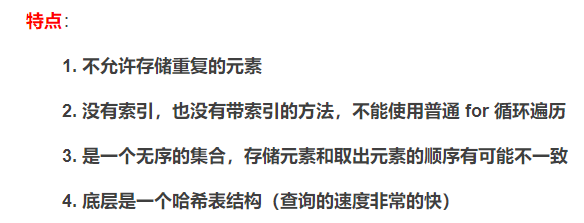
## HashSet











## HashSet底层数据结构是哈希表如何保证元素唯一性？

是通过元素的两个方法，hashCode和equals方法

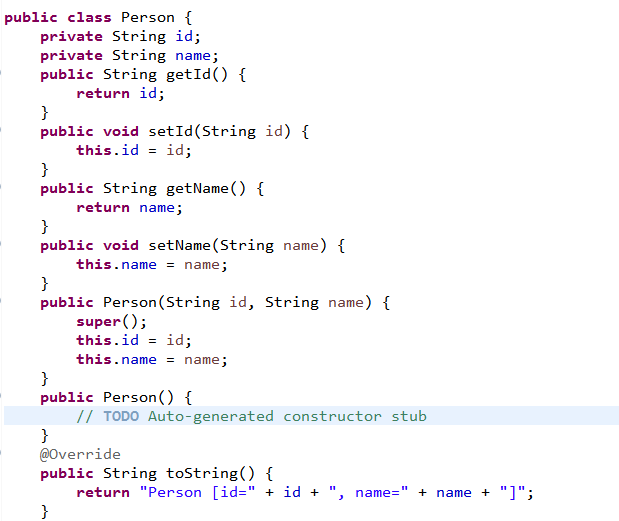
如果哈希值不同（hashcode），就直接添加到集合中

如果哈希值相同（hashcode），再比较equals方法。

## 重写equals()方法和hashcode()例子

Java对象如果要比较是否相等，则需要重写equals方法，同时重写hashCode方法。

将两个对象加入HashSet；

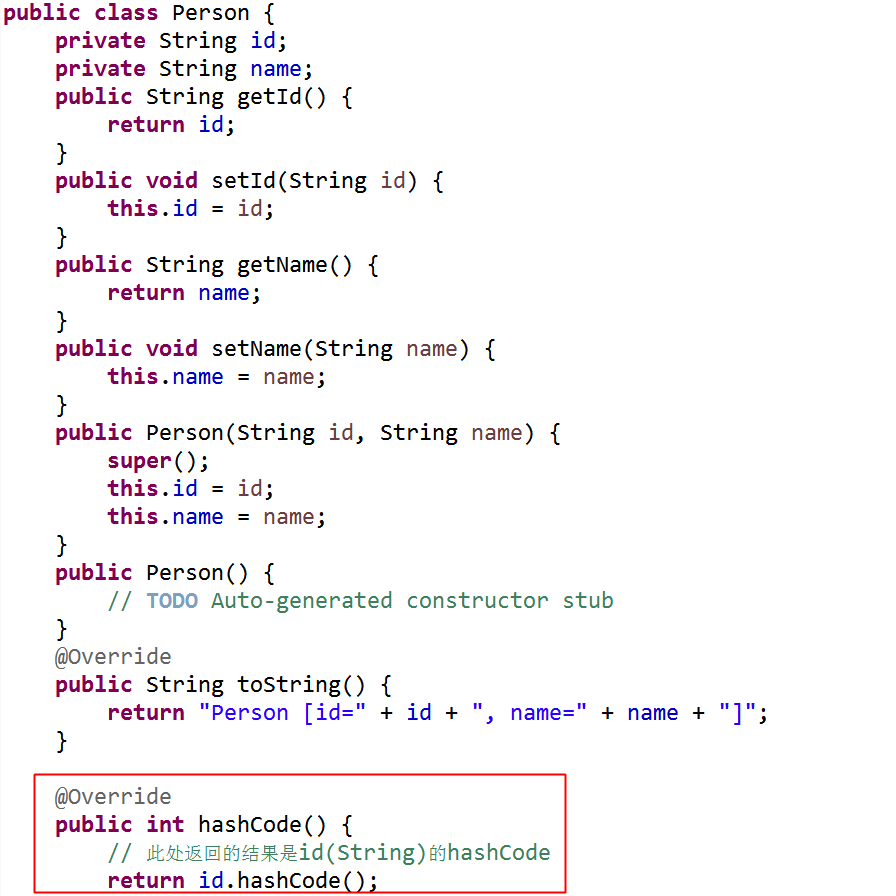


测试set的size结果是多少？

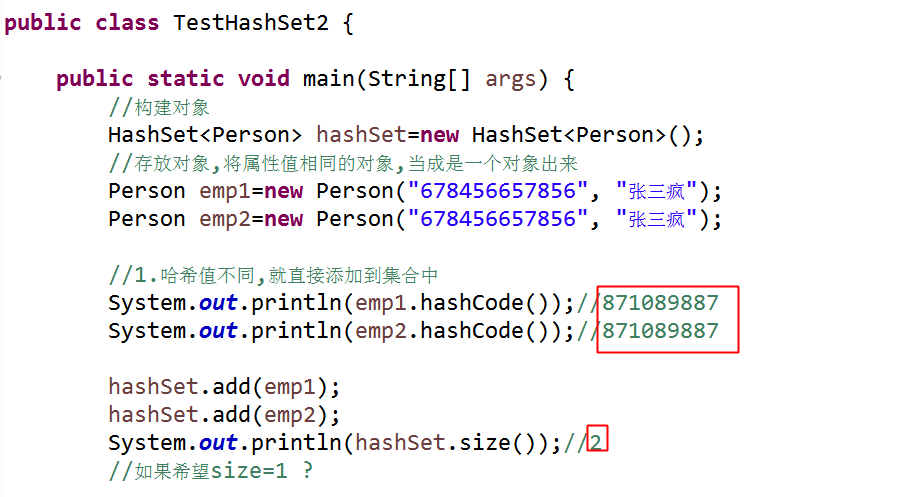


如果希望size为1应该怎么办？

根据内容，我们知道，我们先尝试重写Person的hashcode方法:

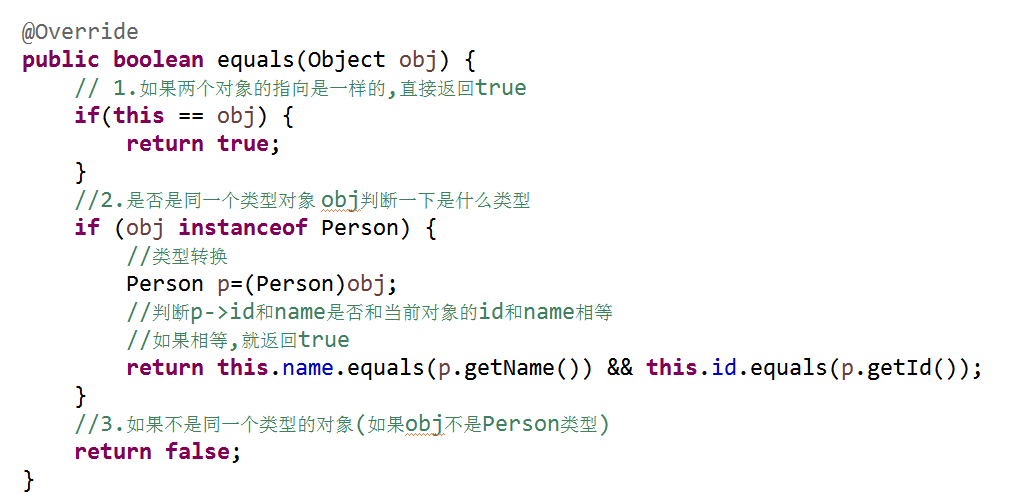


尝试去访问代码:



所以我们要进入第二步:

重写 Person类的equals方法:



测试类:



## 关于HashCode和equals方法再说明：

关于hashcode，我们一定要知道一个口诀：

1、hashcode相等，两个对象不一定相等，需要通过equals方法进一步判断；

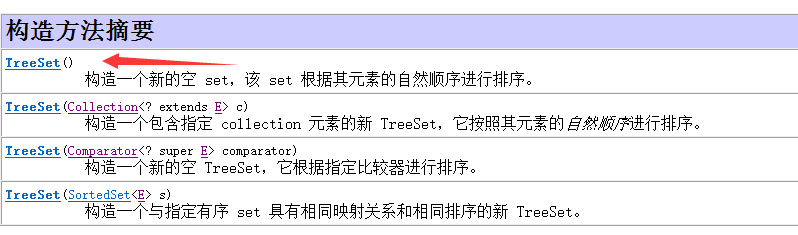
2、hashcode不相等，两个对象一定不相等；

3、equals方法为true，则hashcode肯定一样；

4、equals方法为false，则hashcode不一定不一样，也有可能是一样；

# TreeSet：不重复、自然排序

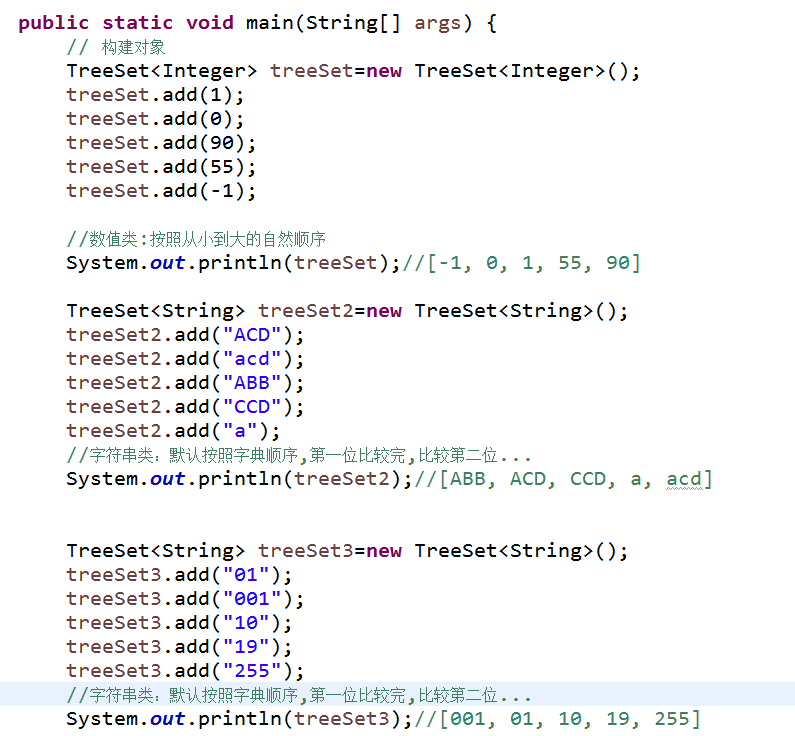




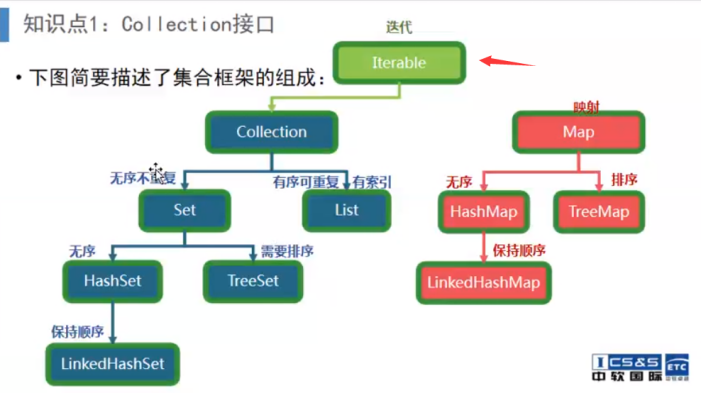
①元素之间有一定自然顺序

②数值类：默认从小到大

③字符串类：默认按照字典顺序



# Map接口



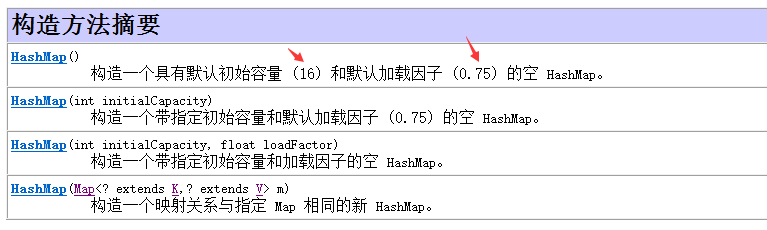


键集,值集, 或键-值映射关系集的形式查看某个映射的内容

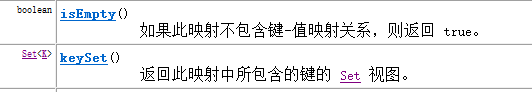
键不能重复,值可以

## HashMap

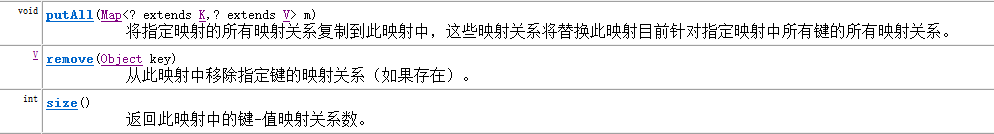














### HashMap特点：

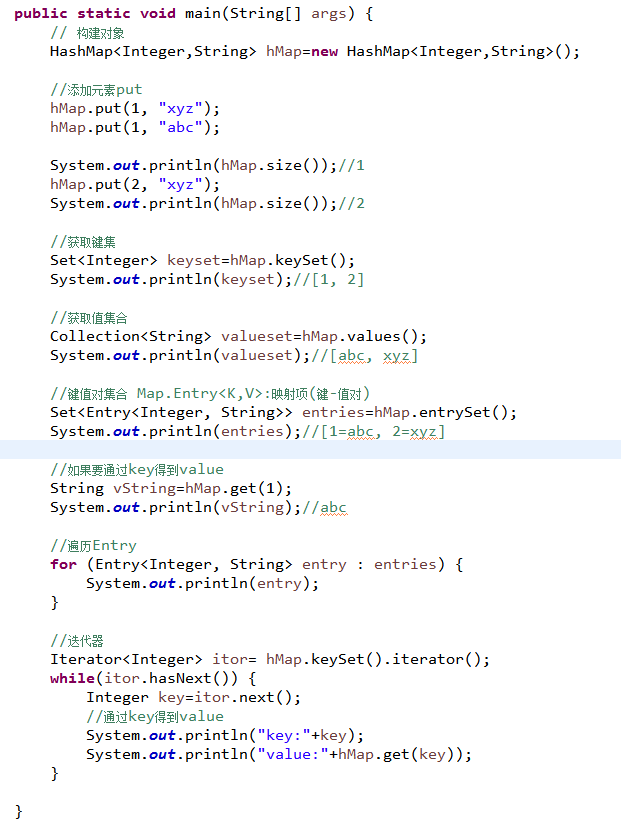
①HashMap<Key,Value> 无序映射集，以键值对存储 key键---value值

②Key键不能重复，可以null值，如果key重复是后面的value覆盖之前的value。

③不同步，多线程不安全

③遍历Map集合：可以使用键集keySet() 或者 键值集entrySet()

### 常见方法：



### 课堂练习:

EmpManage(增加,查询)改成HashMap<Integer,Emp>

## TreeMap(了解)



特点：

①有序映射集，加入关于key的逻辑顺序。Key不能为null

②允许通过实现自定义比较器实现新的比较规则。

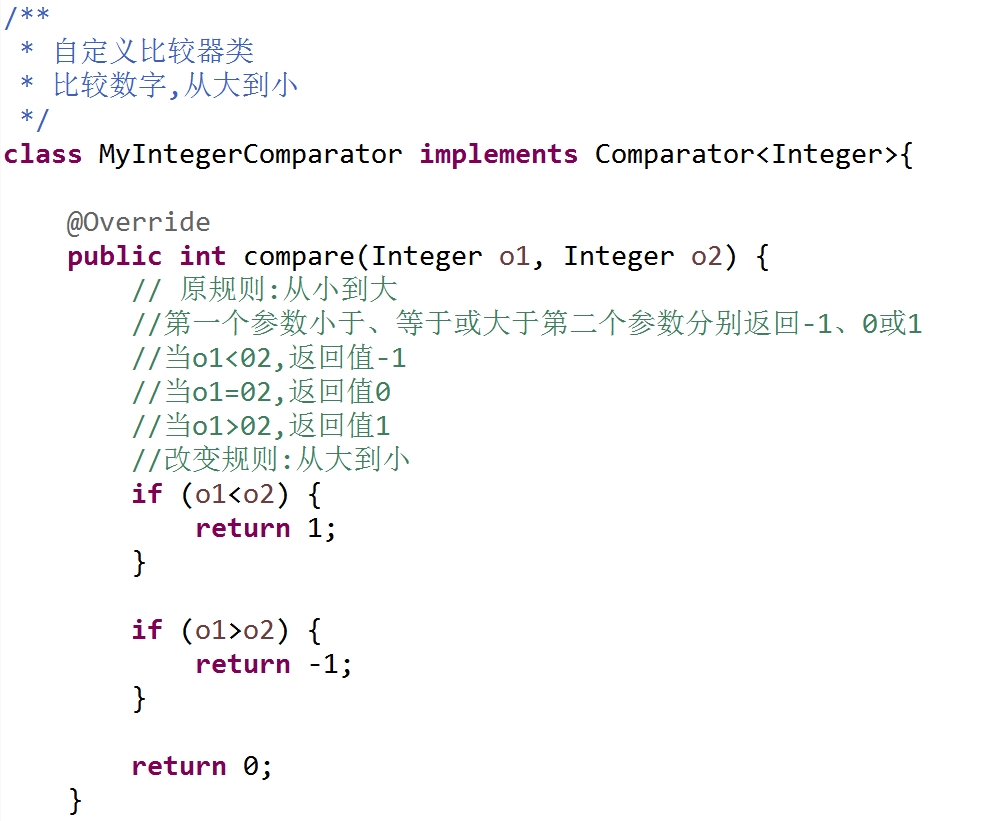
③不同步，多线程不安全。

注意：TreeMap允许通过实现自定义的比较器实现新的比较规则。自定义一个类实现Comparator接口。例如数值从大到小，字符串按照长度从小到大。

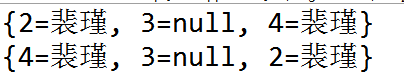
使用构造方法：







效果图:



<? extends T> 表示 任何继承自类型 T 的类型，表示的是此处?我们传入的类型必须是T类型本身或其子类.

<? super T> 表示 任何是类型 T 的超类的类型,表示的是此处?我们传入的类型必须是T类型本身或其超类.

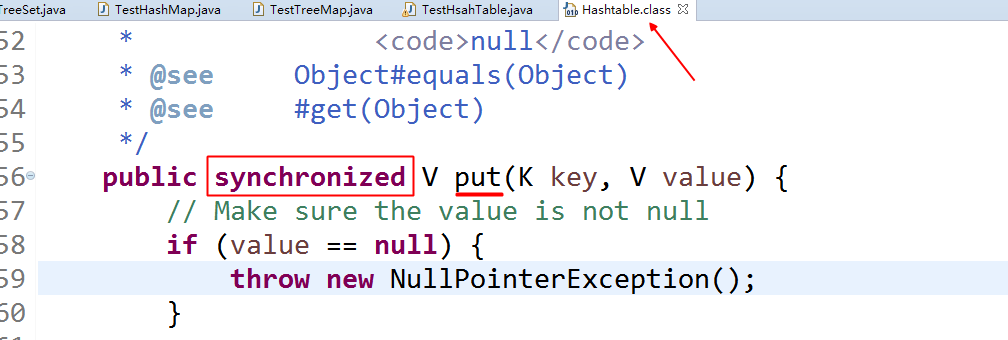
## HashTable(了解)



特点：①不允许将null设置为key; 编译通过，但是运行后报空指针异常。

②线程安全。同步。

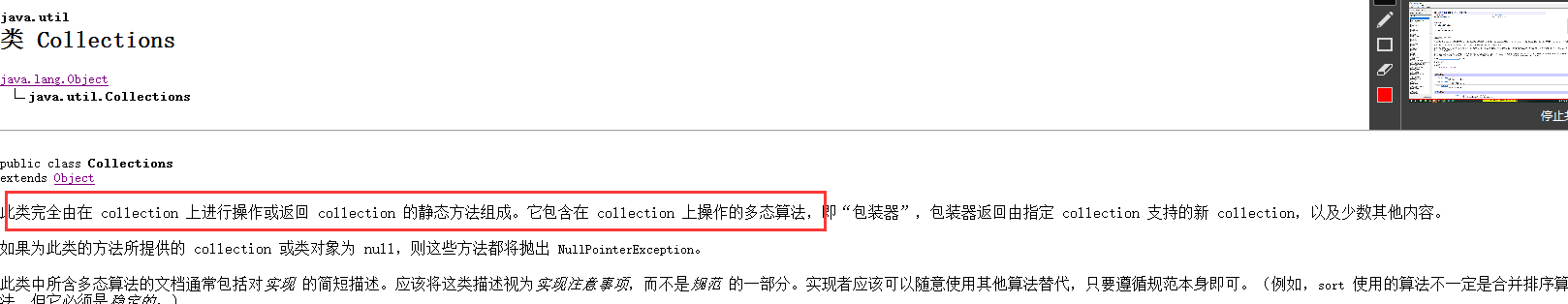




## [面试题]HashMap和TreeMap和HashTable的区别

各自的特点

# Collections：

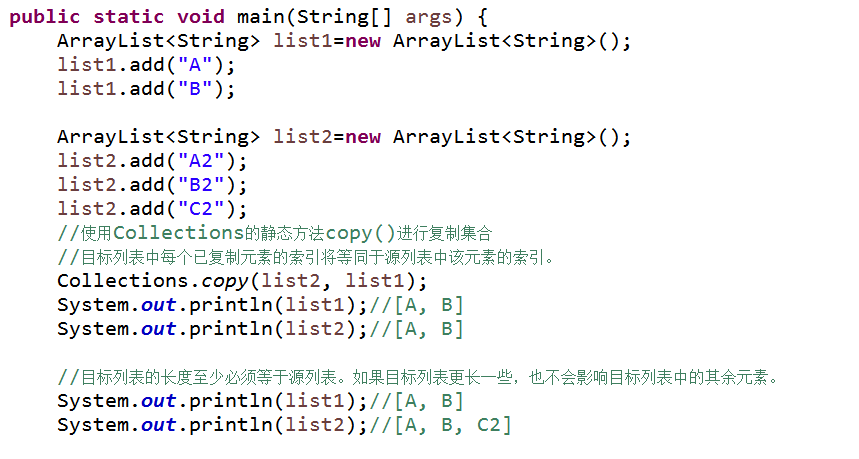


## 【面试题】Collection和Collections有什么区别?

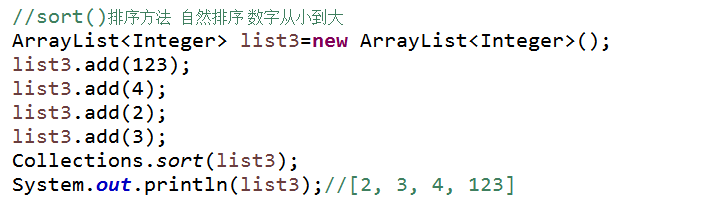
Collection: 是一个接口,表示集合可变长度;

Collections是一个类,这个类提供了很多的静态方法直接完成特定的集合访问;



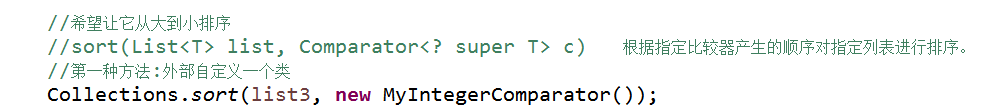


Sort()



## 自定义比较器(难)

使用:

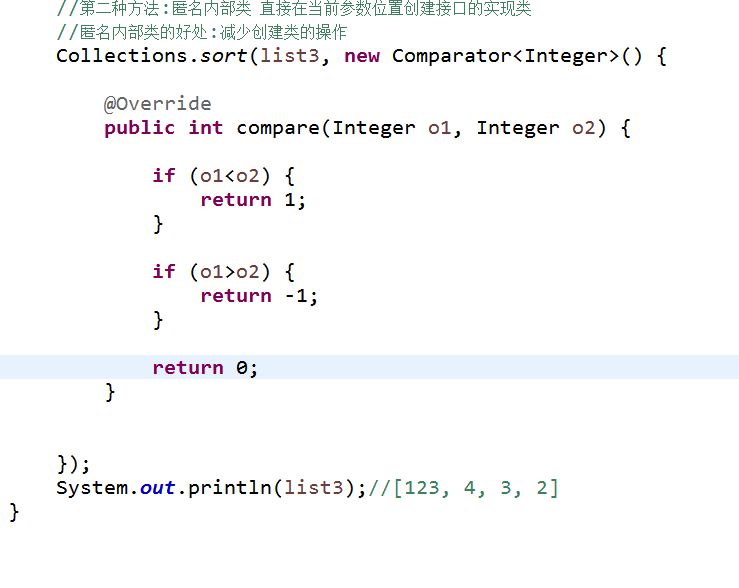




## 匿名内部类

直接在当前参数位置创建接口的实现类，该类没有名称。可以减少类的创建。





# 总结:

1. 集合框架collection接口下,可以使用Iterator来实现遍历

|  |  |
| --- | --- |
| Set接口 | 不重复,无序 |
| Hashset |

1.Collection系列:List(ArrayList,LinkedList Vector) List特征?Set(Hashset/treeset) set特征

2.Hashet如何保证数据的唯一性:如果哈希值不同（hashcode），就直接添加到集合中

如果哈希值相同（hashcode），再比较equals方法。

3.泛型:是jdk1.5之后的参数化类型 泛型类/泛型接口/泛型方法 作用:在编译期间做类型的检测;泛型的擦除:编译完成后,泛型自动擦除,自定义泛型

4.Map:系列:HashMap: key value 一些方法