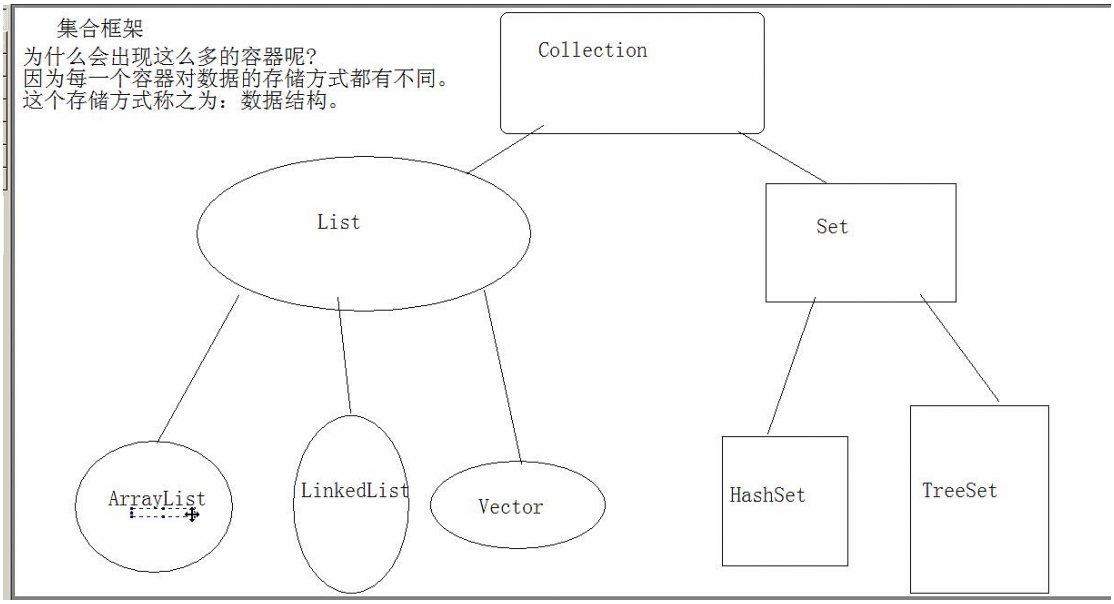
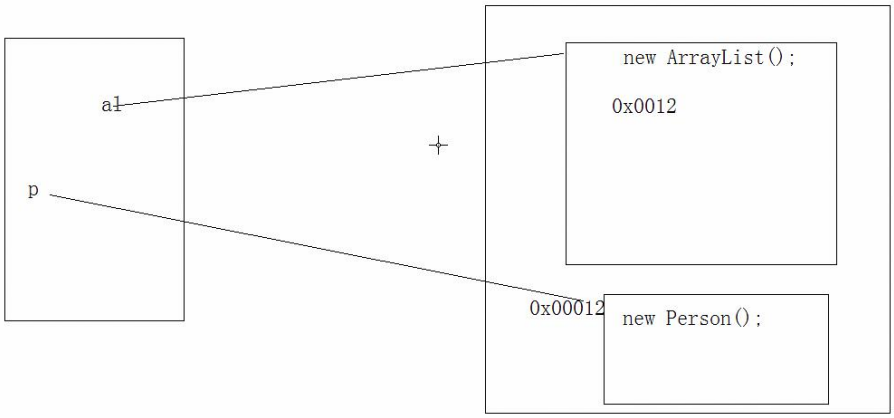
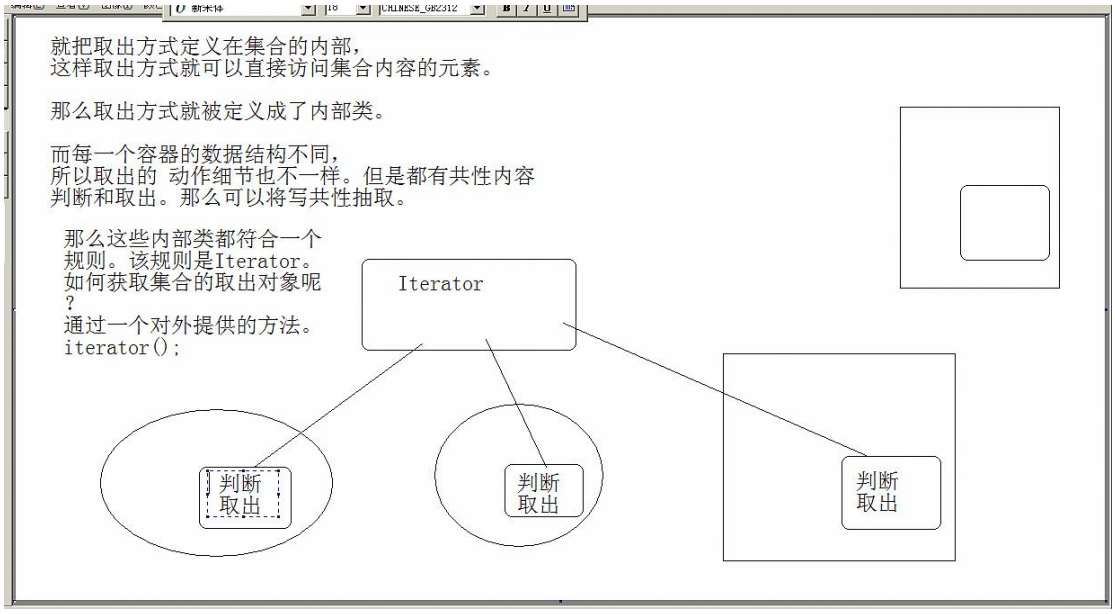
集合框架

为什么会出现这么多的容器呢?
因为每一个容器对数据的存储方式都有不同。
这个存储方式称之为：数据结构。

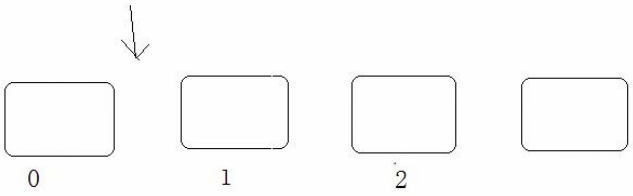


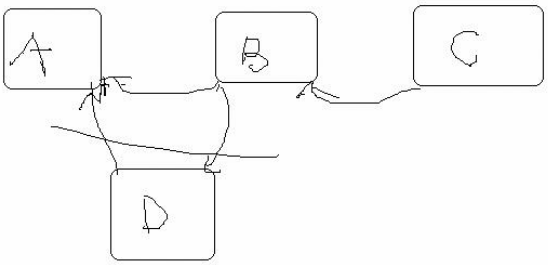

```
Person p = new Person();

ArrayList al = new ArrayList();
al.add(p);
```






```
import java.util.*;
/*
Collection 定义了集合框架的共性功能。
1，添加
    add(e);
    addAll(collection);
2，删除
    remove(e);
    removeAll(collection);
    clear();
3，判断。
    contains(e);
    isEmpty();
4，获取
    iterator();
    size();
5，获取交集。
    retainAll();
6，集合变数组。
    toArray();
1，add 方法的参数类型是 Object。以便于接收任意类型对象。
2，集合中存储的都是对象的引用(地址)
什么是迭代器呢？
其实就是集合的取出元素的方式。
如同抓娃娃游戏机中的夹子。
迭代器是取出方式，会直接访问集合中的元素。
所以将迭代器通过内部类的形式来进行描述。
通过容器的 iterator()方法获取该内部类的对象。
*/
```





List

/*

Collection

|--List:元素是有序的，元素可以重复。因为该集合体系有索引。

|--ArrayList:底层的数据结构使用的是数组结构。特点：查询速度很快。但是增删稍慢。线程不同步。

|--LinkedList:底层使用的链表数据结构。特点：增删速度很快，查询稍慢。线程不同步。

|--Vector:底层是数组数据结构。线程同步。被 ArrayList 替代了。因为效率低。

|--Set：元素是无序，元素不可以重复。、

List：

特有方法。凡是可以操作角标的方法都是该体系特有的方法。

增

add(index,element);

addAll(index,Collection);

删

remove(index);

改

set(index,element);

查

get(index):

subList(from,to);

listIterator();

int indexOf(obj):获取指定元素的位置。
ListIterator listIterator();

List 集合特有的迭代器。ListIterator 是 Iterator 的子接口。

在迭代时，不可以通过集合对象的方法操作集合中的元素。
因为会发生 ConcurrentModificationException 异常。

所以，在迭代器时，只能用迭代器的放过操作元素，可是 Iterator 方法是有限的，
只能对元素进行判断，取出，删除的操作，
如果想要其他的操作如添加，修改等，就需要使用其子接口，ListIterator。

该接口只能通过 List 集合的 listIterator 方法获取。

*/

/*
LinkedList:特有方法：
addFirst();
addLast();

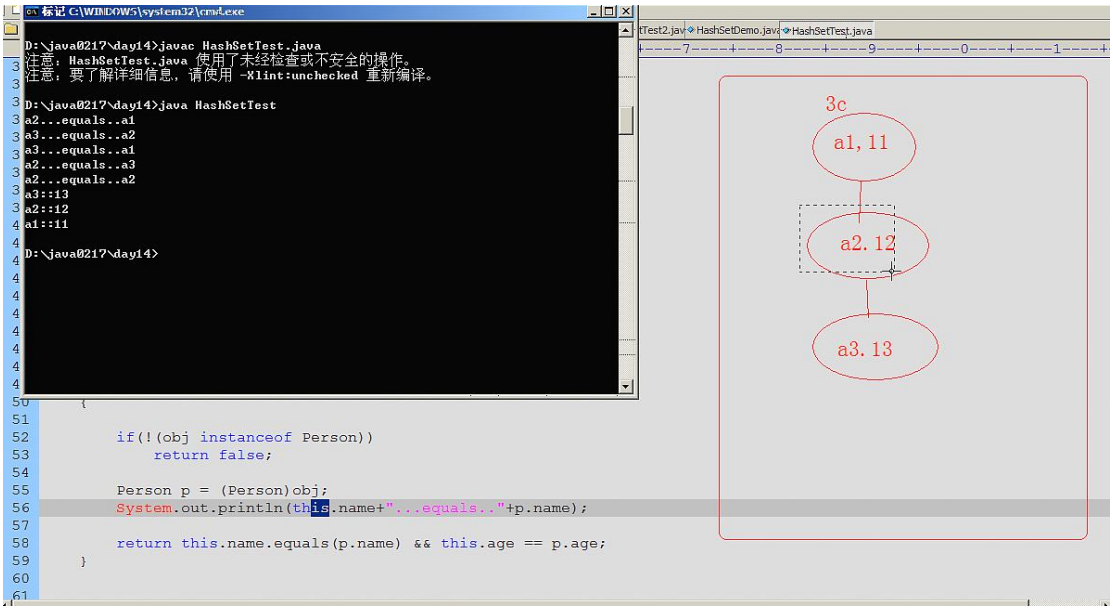
getFirst();
getLast();
获取元素，但不删除元素。如果集合中没有元素，会出现 NoSuchElementException

removeFirst();
removeLast();
获取元素，但是元素被删除。如果集合中没有元素，会出现 NoSuchElementException

在 JDK1.6 出现了替代方法。
offerFirst();
offerLast();

peekFirst();
peekLast();
获取元素，但不删除元素。如果集合中没有元素，会返回 null。

pollFirst();
pollLast();
获取元素，但是元素被删除。如果集合中没有元素，会返回 null。
*/



/*

|--Set：元素是无序(存入和取出的顺序不一定一致)，元素不可以重复。、

|--HashSet:底层数据结构是哈希表。是线程不安全的。不同步。

HashSet 是如何保证元素唯一性的呢？

是通过元素的两个方法，hashCode 和 equals 来完成。

如果元素的 HashCode 值相同，才会判断 equals 是否为 true。

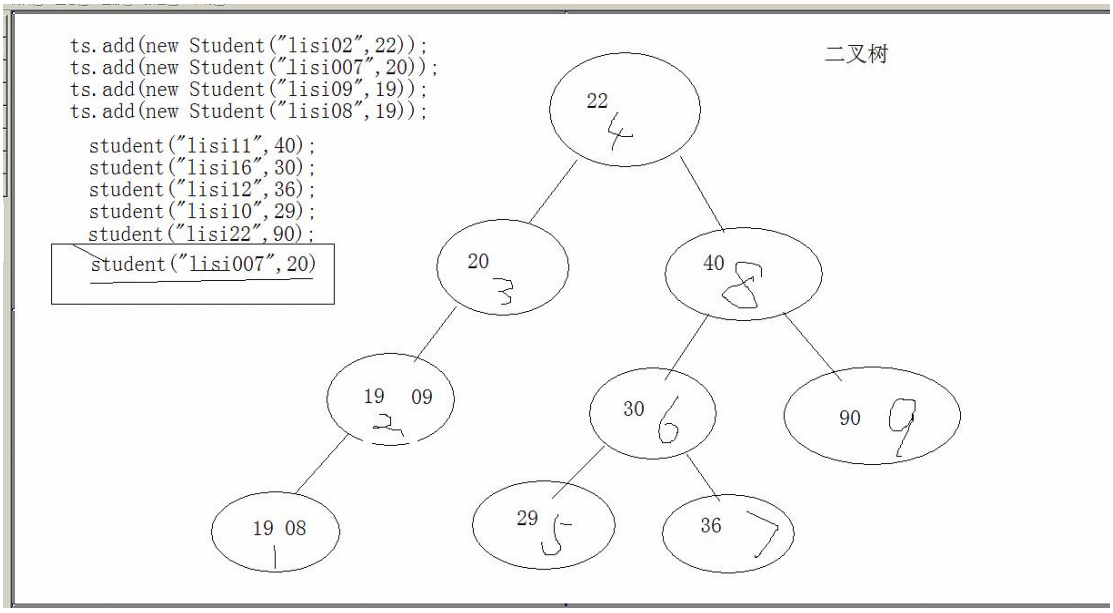
如果元素的 hashcode 值不同，不会调用 equals。

注意,对于判断元素是否存在，以及删除等操作，依赖的方法是元素的 hashcode 和 equals 方法。

|--TreeSet：

Set 集合的功能和 Collection 是一致的。

*/



--TreeSet：可以对 Set 集合中的元素进行排序。

底层数据结构是二叉树。

保证元素唯一性的依据：

compareTo 方法 return 0.

TreeSet 排序的第一种方式：让元素自身具备比较性。

元素需要实现 Comparable 接口，覆盖 compareTo 方法。

也种方式也成为元素的自然顺序，或者叫做默认顺序。

TreeSet 的第二种排序方式。

当元素自身不具备比较性时，或者具备的比较性不是所需要的。

这时就需要让集合自身具备比较性。

在集合初始化时，就有了比较方式。

需求：

往 TreeSet 集合中存储自定义对象学生。

想按照学生的年龄进行排序。

记住，排序时，当主要条件相同时，一定判断一下次要条件。

```
/*
当元素自身不具备比较性，或者具备的比较性不是所需要的。
这时需要让容器自身具备比较性。
定义了比较器，将比较器对象作为参数传递给 TreeSet 集合的构造函数。

当两种排序都存在时，以比较器为主。

定义一个类，实现 Comparator 接口，覆盖 compare 方法。
*/
```

/*
泛型：JDK1.5 版本以后出现新特性。用于解决安全问题，是一个类型安全机制。

好处
1.将运行时期出现问题 ClassCastException，转移到了编译时期。，
　　方便于程序员解决问题。让运行时问题减少，安全。，

2，避免了强制转换麻烦。

泛型格式：通过<>来定义要操作的引用数据类型。

在使用 java 提供的对象时，什么时候写泛型呢？

通常在集合框架中很常见，
只要见到<>就要定义泛型。

其实<> 就是用来接收类型的。

当使用集合时，将集合中要存储的数据类型作为参数传递到<>中即可。
*/

泛型类
*
什么时候定义泛型类？
当类中要操作的引用数据类型不确定的时候，
早期定义 Object 来完成扩展。
现在定义泛型来完成扩展。
*/

泛型方法
*/

//泛型类定义的泛型，在整个类中有效。如果被方法使用，
//那么泛型类的对象明确要操作的具体类型后，所有要操作的类型就已经固定了。
//
//为了让不同方法可以操作不同类型，而且类型还不确定。
//那么可以将泛型定义在方法上。

/*
特殊之处：
静态方法不可以访问类上定义的泛型。
如果静态方法操作的应用数据类型不确定，可以将泛型定义在方法上。

*/

泛型接口
GenericDemo5 文件

泛型限定
/*
? 通配符。也可以理解为占位符。
泛型的限定；
? extends E: 可以接收 E 类型或者 E 的子类型。上限。
? super E: 可以接收 E 类型或者 E 的父类型。下限
*/

/*
Map 集合：该集合存储键值对。一对一对往里存。而且要保证键的唯一性。

1，添加。

put(K key, V value)

putAll(Map<? extends K,? extends V> m)

2，删除。

clear()

remove(Object key)

3，判断。

containsValue(Object value)

containsKey(Object key)

isEmpty()

4，获取。

get(Object key)

size()

values()

entrySet()

keySet()

Map

|--Hashtable:底层是哈希表数据结构，不可以存入 null 键 null 值。该集合是线程同步的。jdk1.0.效率低。

|--HashMap：底层是哈希表数据结构，允许使用 null 值和 null 键，该集合是不同步的。将 hashtable 替代，jdk1.2.效率高。

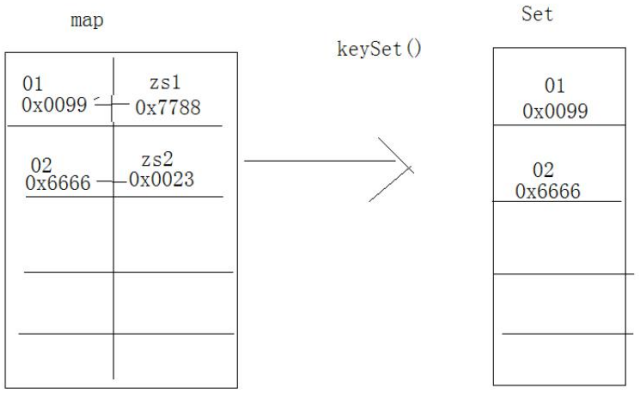
|--TreeMap：底层是二叉树数据结构。线程不同步。可以用于给 map 集合中的键进行排序。

和 Set 很像。

其实大家，Set 底层就是使用了 Map 集合。

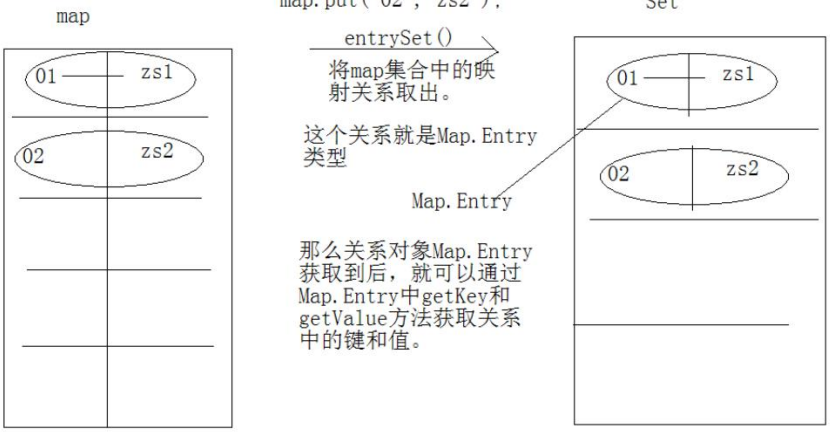
*/

```
Map map = new HashMap();
map.put("01","zs1");
map.put("02","zs2");
```



在通过Set集合的迭代器取出Set集合中键。
通过map的get（key）方法获取键对应的值。

```
Map map = new HashMap();
map.put("01","zs1");
map.put("02","zs2");
```



```
/*
map 集合的两种取出方式：
1，Set<k> keySet：将 map 中所有的键存入到 Set 集合。因为 set 具备迭代器。
    所有可以迭代方式取出所有的键，在根据 get 方法。获取每一个键对应的值。
    Map 集合的取出原理：将 map 集合转成 set 集合。在通过迭代器取出。

2，Set<Map.Entry<k,v>> entrySet：将 map 集合中的映射关系存入到了 set 集合中，
                    而这个关系的数据类型就是：Map.Entry

                    Entry 其实就是 Map 中的一个 static 内部接口。
                    为什么要定义在内部呢？
                    因为只有有了 Map 集合，有了键值对，才会有键值的映射关系。
                    关系属于 Map 集合中的一个内部事物。
                    而且该事物在直接访问 Map 集合中的元素。
*/

/*
Map.Entry  其实 Entry 也是一个接口，它是 Map 接口中的一个内部接口。

interface Map
{
    public static interface Entry
    {
        public abstract Object getKey();
        public abstract Object getValue();

    }
}

class HashMap implements Map
{
    class Hahs implements Map.Entry
    {
        public    Object getKey(){}
        public    Object getValue(){}
    }

}
*/
```

/*
集合框架的工具类。
Collections:集合框架的工具类。里面定义的都是静态方法。

Collections 和 Collection 有什么区别？
Collection 是集合框架中的一个顶层接口，它里面定义了单列集合的共性方法。
它有两个常用的子接口，
List：对元素都有定义索引。有序的。可以重复元素。
Set：不可以重复元素。无序。

Collections 是集合框架中的一个工具类。该类中的方法都是静态的
提供的方法中有可以对 list 集合进行排序，二分查找等方法。
通常常用的集合都是线程不安全的。因为要提高效率。
如果多线程操作这些集合时，可以通过该工具类中的同步方法，将线程不安全的集合，转换成安全的。
*/

/*
Arrays:用于操作数组的工具类。
里面都是静态方法。

asList:将数组变成 list 集合

//把数组变成 list 集合有什么好处？
/*可以使用集合的思想和方法来操作数组中的元素。

注意：将数组变成集合，不可以使用集合的增删方法。
因为数组的长度是固定。
contains。Get indexOf() subList();
如果你增删。那么会反生 UnsupportedOperationException,
*/

/*
如果数组中的元素都是对象。那么变成集合时，数组中的元素就直接转成集合中的元素。
如果数组中的元素都是基本数据类型，那么会将该数组作为集合中的元素存在。
*/

/*
集合变数组。
Collection 接口中的 toArray 方法。
*/
/*
1,指定类型的数组到底要定义多长呢？
当指定类型的数组长度小于了集合的 size，那么该方法内部会创建一个新的数组。长度为集合的 size。
当指定类型的数组长度大于了集合的 size，就不会新创建了数组。而是使用传递进来的数组。
所以创建一个刚刚好的数组最优。

2,为什么要将集合变数组？
为了限定对元素的操作。不需要进行增删了。
*/

*
高级 for 循环

格式：
for(数据类型 变量名 ： 被遍历的集合(Collection)或者数组)
{

}
对集合进行遍历。
只能获取集合元素。但是不能对集合进行操作。

迭代器除了遍历，还可以进行 remove 集合中元素的动作。
如果是用 ListIterator，还可以在遍历过程中对集合进行增删改查的动作。

传统 for 和高级 for 有什么区别呢？

高级 for 有一个局限性。必须有被遍历的目标。

建议在遍历数组的时候，还是希望是用传统 for。因为传统 for 可以定义脚标。
*/

/*
JDK1.5 版本出现的新特性。

方法的可变参数。
在使用时注意：可变参数一定要定义在参数列表最后面。
*/

/*
可变参数。
其实就是上一种数组参数的简写形式。
不用每一次都手动的建立数组对象。
只要将要操作的元素作为参数传递即可。
隐式将这些参数封装成了数组。
*/