1. 集合
2. 定义

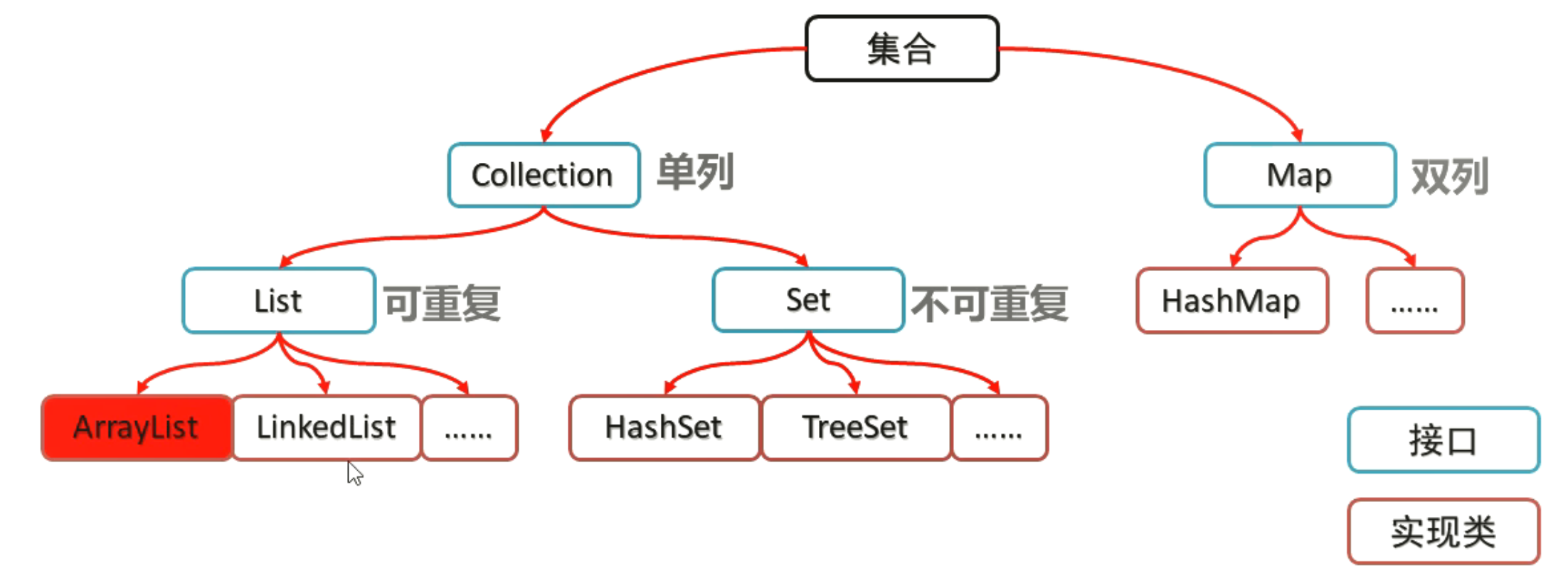
集合是一种存储空间可变的存储模型，用于存取数据，优点是容量可变。

1. 集合类的体系结构

集合分为单列集合和双列集合。

单列集合为Collection接口，又分为List和Set接口，List接口又分别用ArrayList类，LinkdedList类等等类实现接口。

双列集合为Map接口，有HashMap类等等类实现Map接口。



1. Collection集合
2. 定义

上面说过，Collection集合是一个单列集合的顶层接口。它不提供具体的实现，而是用于传递集合，让更加具体的子接口如Set和List接口来实现。

1. Collection集合的常用方法

方法名 说明

boolean add(E e) 添加元素

boolean remove(Object o) 从集合中移除指定的元素

void clear() 清空集合中的元素

boolean contains(Object o) 判断集合中是否存在指定的元素

boolean isEmpty() 判断集合是否为空

int size() 集合的长度，也就是集合中元素的个数

1. 创建Collection对象

用多态的方式，使用Collection的子接口的具体实现类来创建Collection对象。

如：

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

1. Collection的遍历
2. 迭代器

集合可以使用迭代器进行遍历，如何得到迭代器呢？

通过iterator iterator()方法可以得到该集合的迭代器。

1. 遍历示例

public class IteratorDemo {

  public static void main(String[] args) {

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

    //添加元素

    c.add("hello");

    c.add("world");

    //Iterator<E> iterator()：返回此集合中元素的迭代器

    Iterator<String> it = c.iterator();

    while (it.hasNext()) {

      String s = it.next();

      System.out.println(s);

   }

 }

}

1. List集合
2. 定义

List接口是Collection接口的子接口，可以使用Collection接口的功能，也有自己特有的功能。List集合称为有序集合，允许有重复的元素。用户可以精确控制列表中每个元素的插入位置。用户可以通过整数索引访问元素，并搜索列表中的元素。

1. List集合的特有方法

方法名 描述

void add(int index,E element) 在此集合中的指定位置插入指定的元素

E remove(int index) 删除指定索引处的元素，返回被删除的元素

E set(int index,E element) 修改指定索引处的元素，返回被修改的元素

E get(int index) 返回指定索引处的元素

1. ListIterator
2. 定义

List集合特有的迭代器，可以双向移动迭代器。

1. 获取方法

通过List集合的listIterator()方法得到。

1. 特有方法

（1）E previous()

返回列表中的上一个元素

（2）Boolean hasPrevious()

如果列表迭代器在相反方向遍历时有更多元素，返回true

1. List集合的实现类
2. ArrayList集合

底层实现是数组，数组的结构特点是查询块，增删慢。

1. LinkedList集合

底层实现是链表，链表的特点是查询慢，增删快。

1. LinkedList集合的特有方法

由于LinkedList集合是链表实现，因此有链表的相关方法实现。

方法名 说明

public void addFirst(E e) 在该列表开头插入指定的元素

public void addLast(E e) 将指定的元素追加到此列表的末尾

public E getFirst() 返回此列表中的第一个元素

public E getLast() 返回此列表中的最后一个元素

public E removeFirst() 从此列表中删除并返回第一个元素

public E removeLast() 从此列表中删除并返回最后一个元素

1. Set集合
2. 定义

Collection接口的子接口，元素存取无序，不能存储重复元素，没有索引，只能用迭代器或者增强for循环遍历。

1. HashSet
2. 哈希值

（1）定义

JDK根据对象的地址或者字符串或者数字算出来的int类型的数值

（2）获取方法

Object类中的public int hashCode()：返回对象的哈希码值

（3）特点

同一对象的哈希值是相同的。

一般不同对象的哈希值是不同的，但通过重写hashCode()方法，可以让不同对象的哈希值相同。

1. HashSet集合的特点

（1）底层的数据结构为哈希表

（2）存取顺序不相同

（3）不包含重复元素

1. 哈希表

底层为数组+链表实现，存储链表的数组。将哈希值通过某种计算得出的数值结果相同的连成一个链表，存储在数组中。

1. HashSet集合保证元素唯一性原理
2. 根据哈希值计算存储位置

根据对象的哈希值计算存储位置，若该位置没有元素则直接存入

若该位置有元素存在，则进入下一步。

1. 比较哈希值

比较已存在的元素的哈希值与当前元素的哈希值，若哈希值不同，则将当前元素存储，若相同则进入下一步。

1. equals（）方法比较两个元素的内容

如果内容不同，则将当前元素存储，若相同，则不存储当前元素。

1. 注意事项

HashSet要实现存储元素唯一性，则要重写存储元素对象类的equals()方法和hashCode()方法，通过idea的自动重写即可实现。

1. LinkedHashSet集合

特点：

哈希表和链表实现的Set集合，链表保证元素的有序，则元素的存取顺序相同；哈希表保证元素唯一。

1. TreeSet集合
2. 定义

TreeSet实现Comparable接口，该接口对于实现它的类的对象强加一个自然排序。同时TreeSet接口也实现了Comparator接口，该比较器可以对对象进行排序。

使用自然排序还是比较器排序，由构造方法决定。

1. 特点

（1）元素按照一定规则排序，排序方式为自然排序或者规定的比较器

（2）不包含重复元素

（3）没有索引，不能用普通for循环遍历

1. Comparable自然排序的使用

Comparable自然排序接口对于实现改接口的每个类对象都施加自然排序。因此，在使用Comparable接口的每个类都要implements该接口。

该类实现Comparable接口时，要重写compareTo()方法，该方法中如果返回0，则说明是重复元素，不添加；如果是正数，按照升序存储；如果是负数，按照降序存储。

1. compareTo（）方法重写实例理解

（1）如果重写compareTo（）方法如下所示，如何理解呢？

@override

Public int compareTo(Student s){}

Int num = this.age – s.age;

Return num;

}

理解：This理解为较大的元素，s理解为较小的元素；若返回this.age – s.age ，则按照升序存储，若返回 s.age – this.age ，则按照降序存储；若返回0，则不添加。

（2）重写compareTo()方法如下所示，如何理解？

@Override

Public int compareTo(Student s){

Int num = this.age – s.age;

Int num2 = num==? This.name.compareTo(s.name) : num;

Return num2;

}

理解：按照age升序排列，若age相同，则按照姓名的字母顺序排列。注意主要条件是age，次要条件是name。

1. 比较器Comparator的使用

比较器Comparator需要在构造TreeSet时传入，Comparator也是一个接口，我们可以选择用实现类实现接口，也可以用匿名内部类的方式构造匿名对象。

使用匿名内部类时，构建的是匿名类对象，如下所示：  
TreeSet<Student> ts = new TreeSet<Student>(new Comparator<Student>(){

@Override

Public int compare(Student s1, Student s2){

Int num = s1.getAge() – s2.getAge();

Int num2 = num == 0 ? s1.getName().compareTo(s2.getName()) : num;

Return num2;

}

});

Comparator接口重写的是compare（）方法，这里的s1代表着之前的this，这里的s2代表着之前的s。操作都是相同的，因此不过多介绍了。

1. Map集合
2. 概述

Map接口是双列集合接口，有键值和对应的值。

interface Map<K,V>  K：键的类型；V：值的类型

1. 特点
2. 键值对的映射关系
3. 键值不能重复，但是值可以重复
4. 一个键值对应一个值
5. 元素的存取是无序的
6. Map集合的使用

注意Map是接口，那么要使用Map集合就要通过Map接口的实现类实现Map接口。

如：

Map<String, String> map = new HashMap<String, String>();

Map接口用HashMap实现类创建对象。

1. Map接口的常用方法

方法名 说明

V put(K key,V value) 添加元素

V remove(Object key) 根据键删除键值对元素

void clear() 移除所有的键值对元素

boolean containsKey(Object key) 判断集合是否包含指定的键

boolean containsValue(Object value) 判断集合是否包含指定的值

boolean isEmpty() 判断集合是否为空

int size() 集合的长度，也就是集合中键值对的个数

V get(Object key) 根据键获取值

Set keySet() 获取所有键的集合

Collection values() 获取所有值的集合

Set<Map.Entry<K,V>> entrySet() 获取所有键值对对象的集合

1. Map集合的遍历
2. 根据键值遍历
3. 获取所有键值，用keySet()方法
4. 遍历键值的集合，用增强for循环实现
5. 根据键值找对应的值
6. 根据键值对遍历
7. 获取所有键值对集合， Set<Map.Entry<k,v>> entrySet()
8. 遍历键值对，获取键值和对应的值
9. Collections集合工具类

（一）定义

Collections类是一个工具类，作用于集合。工具类中的成员方法都是private，因此需要用类名直接调用。

如：

Collections.sort()

（二）常用方法

方法名 说明

public static void sort(List list) 将指定的列表按升序排序

public static void reverse(List<?> list) 反转指定列表中元素的顺序

public static void shuffle(List<?> list) 使用默认的随机源随机排列指定的列表

1. 增强for循环
2. 定义

学过了Iterator迭代器后，我们就需要会使用增强for循环。其内部原理是一个Iterator迭代器，所有实现了Iterable接口的类的对象都能够使用增强for循环。

1. 格式

For(元素数据类型 变量名 ： 数组或者Collection集合){}

如：

Int[] arr = {1,2,3};

For(int I : arr)

{

System.out.println(i);

}