# 集合

## 1.集合体系结构

**集合分为Collection接口和Map接口,单列代表一个索引对应一个对象,如姓名.双列代表一个索引对应一组两个对象,如姓名＋学号,这两个都需要存储并带有一定联系的数据.List接口中应许重复数据,Set接口不存在重复数据,录入的时候录不进去的**

## 2.Collection集合的概述和使用

**Collection集合的概述**

**是单列集合的顶层接口,他表示一组对象.这些对象也称为Collection的元素**

**JDK不提供该接口的直接子实现,它提供了更具体的子接口(如List和Set)实现**

**创建Collection对象**

**多态创建,使用具体实现类ArrayList**

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

**创建了一个字符串集合**

**成员方法**

**c.add(): 添加元素**

**c.remove(): 从集合中移除指定元素**

**clear​(): 从此集合中删除所有元素**

**c.contains():判断集合中是否存在指定元素**

**c.isEmpty(): 判断集合是否为空**

**c.size(): 集合长度,元素个数**

**c.equals​(): 将指定的对象与此集合进行比较以获得相等性**

**等查看API帮助文档**

## 3.Collection集合的遍历

**Iterator:迭代器,集合的专用遍历方式**

**c.iterator(); 返回此集合中元素的迭代器,通过集合的iterator()方法得到**

**迭代器是通过集合的iterator()方法得到的,所以说它是依赖于集合而存在的**

**Iterator中的常用方法**

**ite.next(): 返回迭代器中的下一个元素;**

**ite.hasNext(): 如果迭代具有更多元素,则返回true;**

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

**while (it.hasNext()){**

**System.out.println(it.next());**

**}**

## 4.集合使用步骤图解

## 5.案例Collection集合存储学生对象并遍历

## 6.List集合概述和特点

**List集合概述**

**有序集合(也称为序列),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;**

**与Set集合不同,列表通常应许重复的元素;**

**List集合的特点**

**有序: 存储和取出的元素顺序一致;**

**可重复: 运出的元素应许重复;**

## 7.List集合的特有方法

**list.add(int index, E element); 在集合的指定位置插入指定元素**

**list.remove(int index); 删除指定索引处元素并返回被删除元素**

**list.set(int index, E element); 修改指定索引处元素并返回被修改元素**

**list.get(int index); 返回指定索引处的元素**

**这几个方法中的索引需要在集合中存在,不能传入索引中不存在的索引,否则会导致索引越界异常.**

## 8.案例:List集合存储学生对象并遍历

## 9.并发修改异常

**并发修改异常**

**ConcurrentModificationException**

**产生的原因**

**迭代器遍历的过程中,通过集合对象修改了集合中的元素个数,造成了迭代器获取元素中判断预期修改值和实际修改值不一致**

**解决方案**

**使用for循环集合的get方法遍历,然后用用集合对象做想要的操作即可**

**分析:**

**首先,在查看源码得知,Iterator迭代器的next()方法在调用时,会调用一个checkForComodification()方法判断实际修改集合的次数和预期修改集合的次数是否相等,如果不相等,throw new ConcurrentModificationException**

**就会抛出并发修改异常,这是起因.这里的实际修改集合的次数在源码中是**

**modCount; 预期修改集合的次数是expectedModCount;因为通常做集合遍历大部分情况是为了对集合中的对象进行增删改查等操作,所以在使用迭代器遍历集合时,就会有极大可能发生此异常,而for循环使用集合get索引方法得到集合中每一个元素,从源码中得知,调用list的get方法并不会判断实际修改集合的次数和预期是否相等,所以不会有此异常,故推荐使用get for循环遍历,那这里有一个问题,在使用Collection集合时,怎么样避免,我记得Collection好像没有get方法,只说了迭代器遍历! 实践后发现Collection有get方法,想想也对,创建Collection对象时使用ArrayList多态创建的,是应该有get方法.**

## 10.列表迭代器

**Listiterator: 列表迭代器**

**通过List集合的listiterator()方法得到的,所以说他是list的特有迭代器**

**用于应许程序员沿任意方向遍历列表的列表迭代器,在迭代期间修改列表,并获取列表中迭代器的当前位置.**

**Listiterator中的常用方法:**

**lit.next(): 返回迭代中的下一个元素;**

**lit.hasNext(): 如果迭代具有更多元素,则返回true;**

**lit.previous(): 返回列表的上一个元素;**

**lit.hasPrevious(): 如果迭代器在反方向上遍历具有更多元素,返回true;**

**lit.add(): 将指定元素插入列表,注意这是迭代器的add方法.**

**代码(code):**

**ListIterator<String> lit = list.listIterator();**

**while (lit.hasNext()) {**

**String s = lit.next();**

**if ("hello".equals(s)) {**

**//注意这是使用迭代器的add方法不是集合的add方法**

**lit.add("javaee");**

**}**

**}**

**特别的:**

**首先:Iterator和Listiterator的区别**

**Listiterator extends Iterator**

**Listiterator有正向和反向两种遍历集合的方式,而Iterator只能正向;**

**Listiterator具有集合的增删改查方法;**

**然后:上面代码已经表现出了使用Listiterator迭代器的next()方法遍历集合,再使用Listiterator迭代器的增删改查方法对集合进行修改,并不会发生像Iterator迭代器一样出现的并发修改异常;**

**从源码可以得知, Listiterator迭代器的add方法中,有一个赋值过程,即: expectedModCount = modCount;所以在add方法中其实也调用了checkForComodification();但是这次不会报错.**

## 11.增强for循环

**//Collection体系的集合都可以成为增强for语句的目标;**

**实现Iterator接口的类允许其对象成为增强型for语句的目标;**

**他是JDK5以后出现的,其内部原理是一个Iterator迭代器;**

**格式:**

**for(元素的数据类型 变量名 : 数组名或者Collection集合名){**

**//在此处使用变量即可,该变量就是元素**

**}**

**int[] arr = {1, 2, 3};**

**for(int i : arr){**

**i就是数组中每个元素**

**}**

## 12.集合遍历的三种方式

**迭代器,普通for,增强for**

**一般来说,数组和集合的遍历都是用增强for循环的,因为最简便**

**当遍历的过程中需要使用到索引的话,就是用普通for循环**

**迭代器使用的最少**

## 13.数据结构

**数据结构时计算机存储,组织数据的方式.是指互相之间存在一种或者多种特定关系的数据元素的集合.**

**通常情况下,精心选择的数据结构可以带来更高的数据运行和存储效率.**

**常见数据结构之栈**

**数据进入栈模型的过程称为:压/进栈**

**数据离开栈模型的过程称为:弹/出栈**

**栈是一种数据先进后出的模型,一端开口,栈顶,栈底.**

**常见数据结构之队列**

**数据从后端进入队列模型的过程称为:入队列**

**数据从前端离开队列模型的过程称为:出队列**

**队列是一种数据先进先出的模型,两端开口,前端,后端.**

**常见数据结构之数组**

**数组有一个地址,在地址中每一个索引对应一个数据.**

**查询数据通过索引定位,查询任意数据耗时相同,查询效率高;**

**删除数据时,要将原始数据删除,同时后面每个数据前移,删除效率低;**

**添加元素时,添加位置后面的每个数据都后移,在添加元素,添加效率极低**

**所以数组是一种查询快,增删慢的模型.**

**常见数据接口之链表**

**链表有一个地址,在链表中,数据也具有一个小地址,一个地址和一个数据一一对应,链表中的数据称为节点,节点与节点之间的连接方式是:空小地址+数据1+小地址1+数据二+小地址二...**

**链表是一种增删快,查询慢的模型(对比数组)**

**所以在使用的时候根据需求,合理选择数据结构.**

## 14.List集合子类特点

**ArrayList和LinkedList**

**ArrayList:底层数据结构时数组,查询快,增删慢;**

**LinkedList:底层数据结构时链表,查询慢,增删快;**

**用法:**

**当需求时要用到查询的,那就用ArrayList,当需求要用到增删等对集合操作的时候用LinkedList,当不清楚的时候就用ArrayList.LinkedList和ArrayList拥有相同的成员方法,即增删改查**

## 15.案例练习, ArrayList集合存储学生对象三种方式遍历

**...**

## 16. LinkedList集合的特有功能

**linkedlist.addFirst(); 在该列表开头插入指定元素**

**linkedlist.addLast(); 将指定的元素追加带此列表的末尾**

**linkedlist.getFirst(); 返回列表中的第一个元素**

**linkedlist.getFirst(); 返回列表中的最后一个元素**

**linkedlist.removeFirst(); 从此列表中产出并返回第一个元素**

**linkedlist.removeFirst(); 从此列表中产出并返回最后一个元素**

## 17.Set集合的概述和特点

**public interface Set<E> extends Collection{}**

**Set集合不包含重复元素**

**Set集合没有带索引的方法,所以不能使用普通for循环遍历**

**Set集合创建对象也是使用多态,一般使用其实现类HashSet来创建.**

**特别的发现:**

**虽然Set集合存储数据时无序的,但是好像只要存入了顺序就不会改变,增删也不会改变原有顺序,删就是删除,增会增加在随机位置,但是只要增加了位置也就固定了**

## 18.哈希值

**哈希值:是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**

**Object类中有一个方法可以获取对象的哈希值**

**public int hashCode(): 返回对象的哈希码值**

**对象的哈希值特点:**

**同一个对象多次调用hashCode()方法返回的哈希码值是相同的;**

**默认情况下,不同对象的哈希码值是不同的.而重写hashCode()方法,可以实现让不同对象的哈希值相同.**