一、List接口的介绍

是单列集合的一个重要分支，习惯性的会将实现了List接口的对象成为List集合，在List集合中允许出现重复的元素，所有的元素是以一种线性方式进行存储的，在程序中可以通过索引来访问集合中的指定元素，另外List集合还有一个特点就是元素有序，即元素的存入顺序和取出顺序一致

List接口的特点；

1.有序的集合，存储元素和取出去元素的顺序是一致的（存储123 取出123）

2.有索引，包含了一些带索引的方法

3.允许存储重复元素

List接口带索引的方法（索引）

public void add（int index，E element）；将指定的元素，添加到该集合中的指定位置上。

public E get（int index）；返回集合中指定位置的元素

public E remove（int index）；移除列表中指定的位置的元素，返回的是被移除的元素

public E set （int index，E element）；用指定的元素替换集合中指定位置的元素，返回值的更新前元素。

注意

操作索引的时候，一定防止索引越界异常

IndexOutOfBoundsException；索引越界异常，集合报错

ArrayIndexOutOfBoundsException；数组索引越界异常

StringIndexOutOfBoundsException；字符串索引越界异常

public void add（int index，E element）；将指定的元素，添加到该集合中的指定位置上。

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");
list.add("c");
list.add("d");
list.add("a");
//在C和D之间添加一个itheima
//add.(index:,element)
list.add(3,"itheima");//[a, b, c, itheima, d, a]
 System.out.println(list);//[a, b, c, itheima, d, a]
```

public E remove（int index）；移除列表中指定的位置的元素，返回的是被移除的元素

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");
list.add("c");
list.add("d");
list.add("a");
//移除元素
String removeE = list.remove(2);
System.out.println("被移除的元素"+removeE);//被移除的元素是C
System.out.println(list);//[a, b, itheima, d, a]
```

public E set（int index，E element）；用指定的元素替换集合中指定位置的元素，返回值的更新前元素。

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");
list.add("c");
list.add("d");
list.add("a");
//把最后一个a，替换为A
String setE=list.set(4,"A");
System.out.println("被替换的元素"+setE);//被替换的元素a
System.out.println(list);//[a, b, itheima, d, A]
```

List集合遍历有3种方式

public E get（int index）；返回集合中指定位置的元素

```
                        //List集合遍历有3种方式
//使用普通的for循环
for (int i = 0; i < list.size(); i++) {
    //public E get（int index）；返回集合中指定位置的元素
      String s = list.get(i);
    System.out.println(s);//a b itheima d A
}
System.out.println("==============");
//使用迭代器
Iterator<String> iterator = list.iterator();
while (iterator.hasNext()){
    String s = iterator.next();
    System.out.println(s);
}
System.out.println("==============");
```

```
//使用增强for循环
Iterator<String> iterator1 = list.iterator();
for (String s : list){
    System.out.println(s);
}
// 操作索引的时候，一定防止索引越界异常
String r = list.get(5);
System.out.println(r);
```

二、List集合的实现类

ArrayList集合，元素不同步，增删慢，查找快，由于日常开发中使用最多的功能为查询数据、遍历数据，所以ArrayList是最常用的集合

ArrayList集合特点

增删慢，查找快

LinkedList集合

LinkendList集合数据存储的接口是链表结构，方便元素添加，删除集合。

LinkedList的特点

1.底层是一个链表结构；查询慢，增删快

2.里面包含了大量操作首尾元素的方法

3.线程不安全，效率高

注意；使用Linked集合特有的方法，不能使用多态

LinkedList适用场景

存储数据“查询少，增删多”的场景，如用LinkedList实现栈或者队列。

LinkedList集合；

public void addFirst（E e）；将指定元素插入到此列表的开头。

public void addLast（E e）；将指定元素添加到次列表的结尾

public void push（E e）；将元素推入此列表所表示的堆栈

public E getFirst（ ）；返回此列表的第一个元素。

public E getLast（ ）；返回此列表的最后一个元素。

public boolean isEmpty（ ）；如果列表不包含元素，则返回true

public E removeFirst（ ）；移除并返回次列表的第一个元素

public E removeLast（ ）；移除并返回次列表的最后一个元素

public E pop（ ）；将此列表所表示的堆栈处弹出一个元素

public void addFirst（E e）；将指定元素插入到此列表的开头。

```
private static void show01() {
    //创建LinkedList集合对象
```

```
    LinkedList<String> linkedList = new LinkedList<>();
  //使用add方法往集合中添加元素
    linkedList.add("a");
  linkedList.add("b");
  linkedList.add("c");
  linkedList.add("d");
//public void addFirst（E e）；将指定元素插入到此列表的开头。
linkedList.addFirst("as");
System.out.println(linkedList);//[as,a,b,c]
```

public void push（E e）；将元素推入此列表所表示的堆栈

```
//public void push（E e）；将元素推入此列表所表示的堆栈
//等效于 addFirst
linkedList.push("ass");
System.out.println(linkedList);//[ass, as, a, b, c]
```

public void addLast（E e）；将指定元素添加到次列表的结尾

```
private static void show01() {
  //创建LinkedList集合对象
    LinkedList<String> linkedList = new LinkedList<>();
  //使用add方法往集合中添加元素
    linkedList.add("a");
  linkedList.add("b");
  linkedList.add("c");    linkedList.add("d");
//public void addLast（E e）；将指定元素添加到次列表的结尾
//此方法等效于add方法
linkedList.addLast("cs");
```

```
System.out.println(linkedList);//[ass, as, a, b, c, cs]
```

public E removeFirst（ ）；移除并返回次列表的第一个元素

```
//public E removeFirst（）； 移除并返回次列表的第一个元素
String s1 = linkedList.removeFirst();
System.out.println("被移除的第一个元素"+s1);//a
```

public E removeLast（ ）；移除并返回次列表的最后一个元素

```
//public E removeLast（）； 移除并返回次列表的最后一个元素
String s = linkedList.removeLast();
System.out.println("被移除的第二个元素"+s);//c
```

public E pop（ ）；将此列表所表示的堆栈处弹出一个元素

```
//public E pop（）； 将此列表所表示的堆栈处弹出一个元素
String pop = linkedList.pop();
System.out.println("被移除的第一个元素"+pop);
```

public E getFirst（ ）；返回此列表的第一个元素。

```
//public E getFirst（）； 返回此列表的第一个元素。
String first = linkedListDemo1.getFirst();
   System.out.println(first);//a
```

public E getLast（ ）；返回此列表的最后一个元素。

```
//public E getLast（）； 返回此列表的最后一个元素。
String last = linkedListDemo1.getLast();
System.out.println(last);//d
```

public boolean isEmpty（ ）；如果列表不包含元素，则返回true

```
// public boolean isEmpty（）； 如果列表不包含元素，则返回true
if ( ! linkedListDemo1.isEmpty()){ //! 取反
    String first = linkedListDemo1.getFirst();
   System.out.println(first);
   String last = linkedListDemo1.getLast();
   System.out.println(last);
```

三、Vector类

Vector类可以实现可增长的对象数组，与数组一样，它包含可以使用整数索引进行访问组件，但是 Vector的大小可以根据需要增大或者缩小，以适应创建Vector 后进行添加或移除项的操作

Vector于Collection实现不同，Vector是同步的。

四、Set接口

Set接口和List接口一样，通用继承自Collection接口，它与Collection接口中的方法基本一致，并没有对Collection接口进行功能上的扩充，只是比Collection接口更加严格了。与List接口不同的是，Set接口中元素无序，并且都会以某种规则保证存入的元素不会重复。

Set接口的特点；

1.不允许存储重复元素

2.没有牵引，没有带索引的方法，也不能使用普通的for循环遍历

HashSet集合特点

1.不允许存储重复元素

2.没有牵引，没有带索引的方法，也不能使用普通的for循环遍

3.自身特点 HashSet是一个无序集合，存储的元素和取出元素的顺序有可能不一致，

4.底层是一个哈希表结构（查询速度非常的快）

哈希表的特点 速度快。

二种遍历方式

```
Set<Integer> set = new HashSet<>();
//使用add方法添加元素到集合
set.add(1);
set.add(11);
set.add(12);
set.add(13);
set.add(11);
System.out.println(set);//[1, 11, 12, 13]
//创建迭代器
Iterator<Integer> integerIterator =set.iterator();
while (integerIterator.hasNext()){
    Integer n = integerIterator.next();
    System.out.println(n);//1.11.12.13
//使用增强for变了set集合
for (Integer i:set){
    System.out.println(i);
```

哈希值

是一个十进制的整数，由系统随机给出，重写多少就输出多少（就是对象的地址值，是一个逻辑地址，

是模拟出来的地址，不是数据实际存储的物理地址）

在Object类有一个方法，可以获取对象的哈希值。

五.int hashCode（ ）

返回该对象的哈希值

hashCode方法的源码

public native int hashCode（ ）；

native；代表该方法调用的是本地操作系统的方法

```
//Person类继承了Object类，所以可以使用Object类的hashCode方法
Person person = new Person();
//用int1 获取哈希值
int int1 = person.hashCode();
System.out.println(int1);//42121758 随机获取的十进制哈希值/可以重写变成100
Person person2 = new Person();
Integer integer1 = person2.hashCode();
System.out.println(integer1);//20671747 随机获取的十进制哈希值/可以重写变成100
//重写toString 里的 hashCode
@Override
public int hashCode() {
    return 100;
}
toString方法的源码
*            return getClass（）.getName（）
+"@"+Integer.toHexString(hascCode);*/
```

```
System.out.println(person);//com.itheima.Person@282ba1e
person地址值的十六进制写法
System.out.println(person2);//com.itheima.Person@13b6d03
person2地址值的十六进制写法
```

String类的哈希值

String类重写了Object类的hashCode方法

```
String s1 = new String("aaa");
String s2 = new String("aaa");
System.out.println(s1.hashCode());//96321
System.out.println(s2.hashCode());//96321
System.out.println("重地".hashCode());//1179395
System.out.println("通话".hashCode());//1179395
```

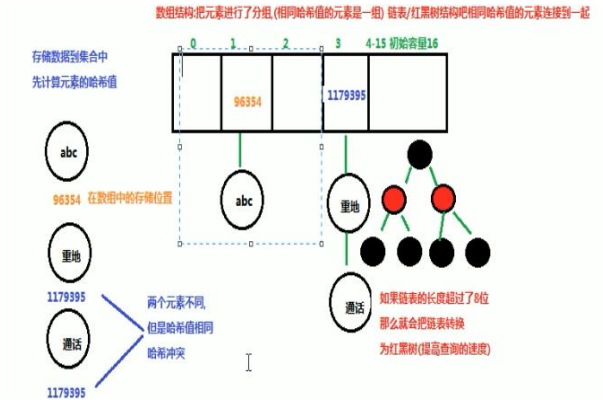
HashSet集合存储数据的结构（哈希表）

idk1.8版本前；哈希表=数组+链表

jdk1.8版本后；哈希值=数组+链表

哈希值=数组+红黑树（提高查询的速度）

哈希表的特点；速度快



HashSet存储自定义类型元素（哈希表）

Set集合不允许重复元素的原理

set集合存储元素不重复的元素

前提；存储的元素必须重写hashCode方法和equals方法


```
public class HashSetDemo {
    public static void main(String[] args) {
        //创建HashSet集合对象
            HashSet<String> set = new HashSet<>();
        String s1 = new String("abc");
        String s2 = new String("abc");
        set.add(s1);
        set.add(s2);
        set.add("通话");
        set.add("重地");
        set.add("abc");
        System.out.println(set);//[通话, 重地, abc]
```

Set集合在调用add方法的时候，add方法会调用元素的hashCode方法和equals方法判断是否重复

set.add（s1）

add方法会调用s1的hashCode方法，计算字符串"ABC"的哈希值，哈希值是96354 在集合中找有没有96354这个哈希值的元素，没有发现 就会把s1存储到集合中

set.add（s2）

add方法会调用s2的hashCode方法计算字符串"abc"的哈希值，哈希值是96354 在集合中找有没有96354这个哈希值的元素，发现有（哈希冲突）

s2会调用equals方法和哈希值相同的元素进行比较

s2.equals（s1）返回true 两个元素的哈希值相同，equals方法返回true认定两个元素相同就不会吧s2存储到集合中

set.add（"重地"）

add方法会调用"重地'的hashCode方法，计算字符串"重地"的哈希值，哈希值是1179395在集合中有没有1179395这个哈希值的元素，发现没有就会把重地存储到集合中

set.add（"通话"）

add方法会调用"重地'的hashCode方法，计算字符串"重地"的哈希值，哈希值是1179395在集合中有没有1179395这个哈希值的元素，发现有（哈希冲突）

“通话”会调用equals方法和哈希值相同的元素进行比较“通话”，equals（“重地”）返回false两个元素的哈希值相同，equals方法返回false，认定两个元素不同，就会把“通话”存储到集合中

LinkedHashSet集合

LinkedHashSet集合 extends HashSet集合

LinkedHashSet特点

底层是一个哈希表（数组+链表/红黑树）+链表；多了一条链表（记录元素的存储顺序），保证元素有序

```
public static void main(String[] args) {
    LinkedHashSet<String> set = new LinkedHashSet<>();
```

```
set.add("www");
set.add("aaa");
set.add("abc");
set.add("itcast");
System.out.println(set);//有序不允许重复
```